W. Stallings, "New Cryptography and Network Security Book", Jun. 8, 1998, 3 pages.
Fasbender,Kesdogan, and Kubitz: "Variable and Scalable Security: Protection of Location Information in Mobile IP", IEEE publication, 1996, pp. 963-967.
Linux FreeS/WAN Index File, printed from http://liberty.freeswan.org/freeswan__ trees/freeswan-1.3/doc/ on Feb. 21, 2002, 3 Pages.
J. Gilmore, "Swan: Securing the Internet against Wiretapping", printed from http://liberty.freeswan.org/freeswan__ trees/freeswan-1.3/doc/rationale.html on Feb. 21, 2002, 4 pages.
Glossary for the Linux FreeS/WAN project, printed from http://liberty.freeswan.org/freeswan__ trees/freeswan-1.3/doc/glossary.html on Feb. 21, 2002, 25 pages.
Alan O. Frier et al., "The SSL Protocol Version 3.0", Nov. 18, 1996, printed from http://www.netscape.com/eng/ss13/draft302.txt on Feb. 4, 2002, 56 pages.
Search Report (dated Aug. 20, 2002), International Application No. PCT/US01/04340.
Search Report (dated Aug. 23, 2002), International Application No. PCT/US01/13260.
Shree Murthy et al., "Congestion-Oriented Shortest Multipath Routing", Proceedings of IEEE INFOCOM, 1996, pp. 1028-1036.
Jim Jones et al., "Distributed Denial of Service Attacks: Defenses", Global Integrity Corporation, 2000, pp. 1-14.
James E. Bellaire, "New Statement of Rules—Naming Internet Domains", Internet Newsgroup, Jul. 30, 1995, 1 page.
D. Clark, "US Calls for Private Domain-Name System", Computer, IEEE Computer Society, Aug. 1, 1998, pp. 22-25.
August Bequai, "Balancing Legal Concerns Over Crime and Security in Cyberspace", Computer & Security, vol. 17, No. 4, 1998, pp. 293-298.
Rich Winkel, "CAQ: Networkinig With Spooks: The NET & The Control Of Information", Internet Newsgroup, Jun. 21, 1997, 4 pages.
Search Report (dated Oct. 7, 2002), International Application No. PCT/US01/13261.
F. Halsall, "Data Communications, Computer Networks And Open Systems", Chapter 4, Protocol Basics, 1996, pp. 198-203.
Reiter, Michael K. and Rubin, Aviel D. (AT&T Labs—Research), "Crowds: Anonymity for Web Transmissoins", pp. 1-23.
Dolev, Shlomi and Ostrovsky, Rafil, "Efficient Anonymous Multicast and Reception"(Extended Abstract), 16 pages.
Rubin, Aviel D., Greer, Daniel, and Ranum, Marcus J. (Wiley Computer Publishing), "Web Security Sourcebook", pp. 82-94.
Fasbender, Kesdogan, and Kubitz: "Variable and Scalable Security" Protection of Location Information in Mobile IP, IEEE publication, 1996, pp. 963-967.
Eastlake, D. E., "Domain Name System Security Extensions", Internet Draft, Apr. 1998, XP002199931, Sections 1, 2.3 and 2.4.
RFC 2401 (dated Nov. 1998) Security Architecture for the Internet Protocol (RTP).
RFC 2543-SIP (dated Mar. 1999): Session Initiation Protocol (SIP or SIPS).
Search Report, IPER (dataed Nov. 13, 2002), International Application No. PCT/US01/04340.
Search Report, IPER (dated Feb. 6, 2002), International Application No. PCT/US01/13261.
Search Report, IPER (dated Jan. 14, 2003), International Application No. PCT/US01/13260.
Shankur, A.U. "A verified sliding window protocol with variable flow control". Proceedings of ACM SIGCOMM conference on Communications architectures & protocols. pp. 84-91, ACM Press, NY, NY 1986.
W. Stallings, "Crytography and Network Security", 2nd, Edition, Chapter 13, IP Security, Jun. 8, 1998, pp. 399-440.

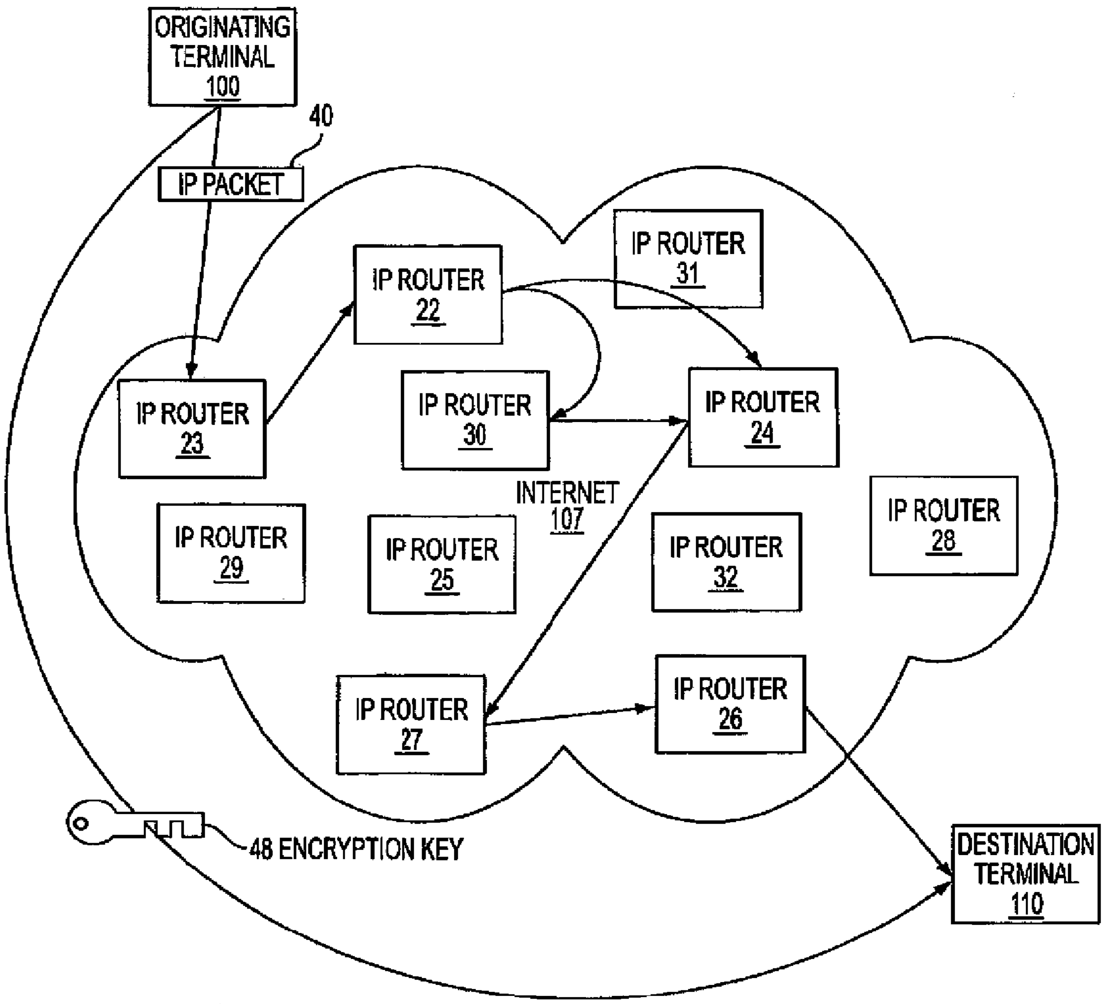

ORIGINATING TERMINAL 100
40
IP PACKET
IP ROUTER 22
IP ROUTER 31
IP ROUTER 23
IP ROUTER 30
IP ROUTER 24
INTERNET 107
IP ROUTER 29
IP ROUTER 25
IP ROUTER 32
IP ROUTER 28
IP ROUTER 27
IP ROUTER 26
48 ENCRYPTION KEY
DESTINATION TERMINAL 110

FIG. 1

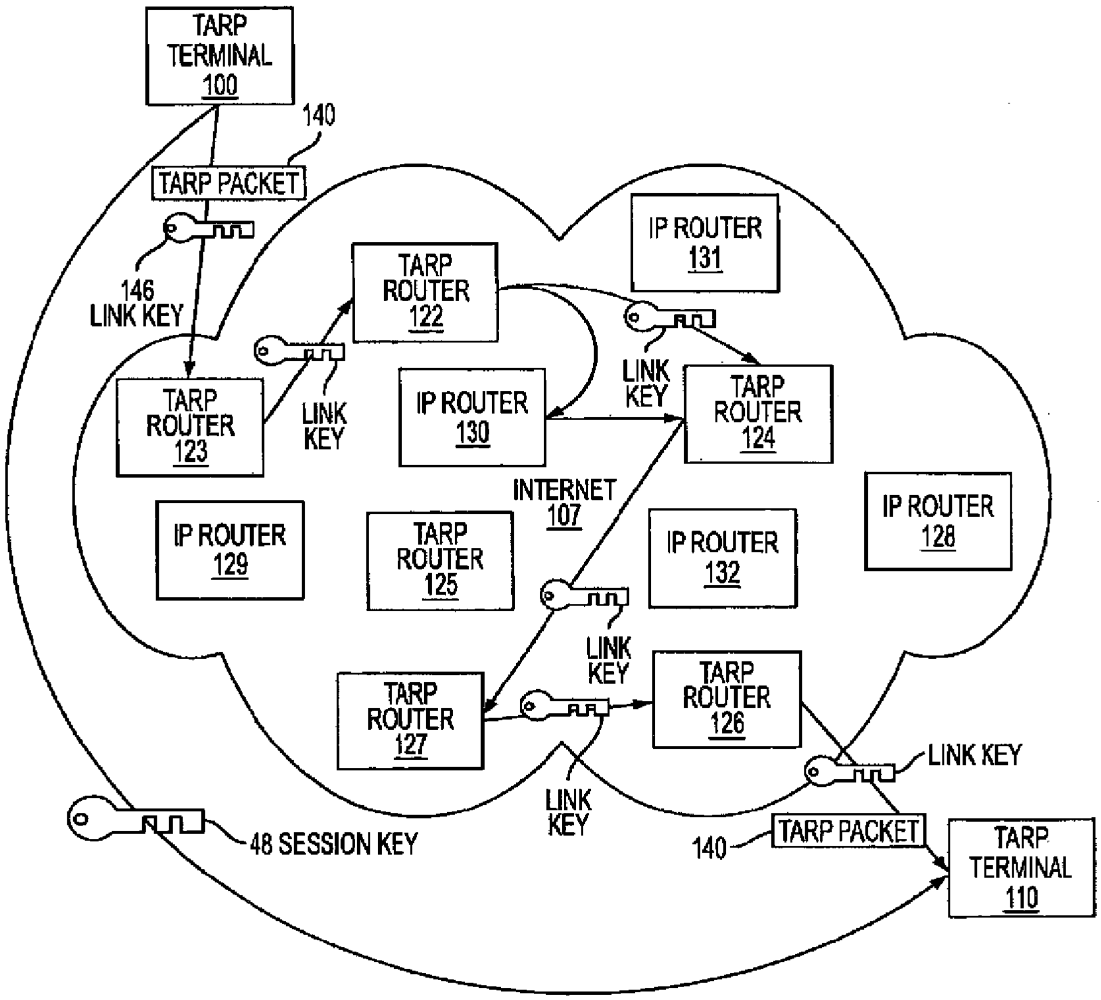

TARP TERMINAL 100
140
TARP PACKET
146 LINK KEY
TARP ROUTER 122
IP ROUTER 131
LINK KEY
TARP ROUTER 123
LINK KEY
IP ROUTER 130
LINK KEY
TARP ROUTER 124
INTERNET 107
IP ROUTER 129
TARP ROUTER 125
IP ROUTER 132
IP ROUTER 128
LINK KEY
TARP ROUTER 126
TARP ROUTER 127
LINK KEY
LINK KEY
48 SESSION KEY
140
TARP PACKET
TARP TERMINAL 110

FIG. 2

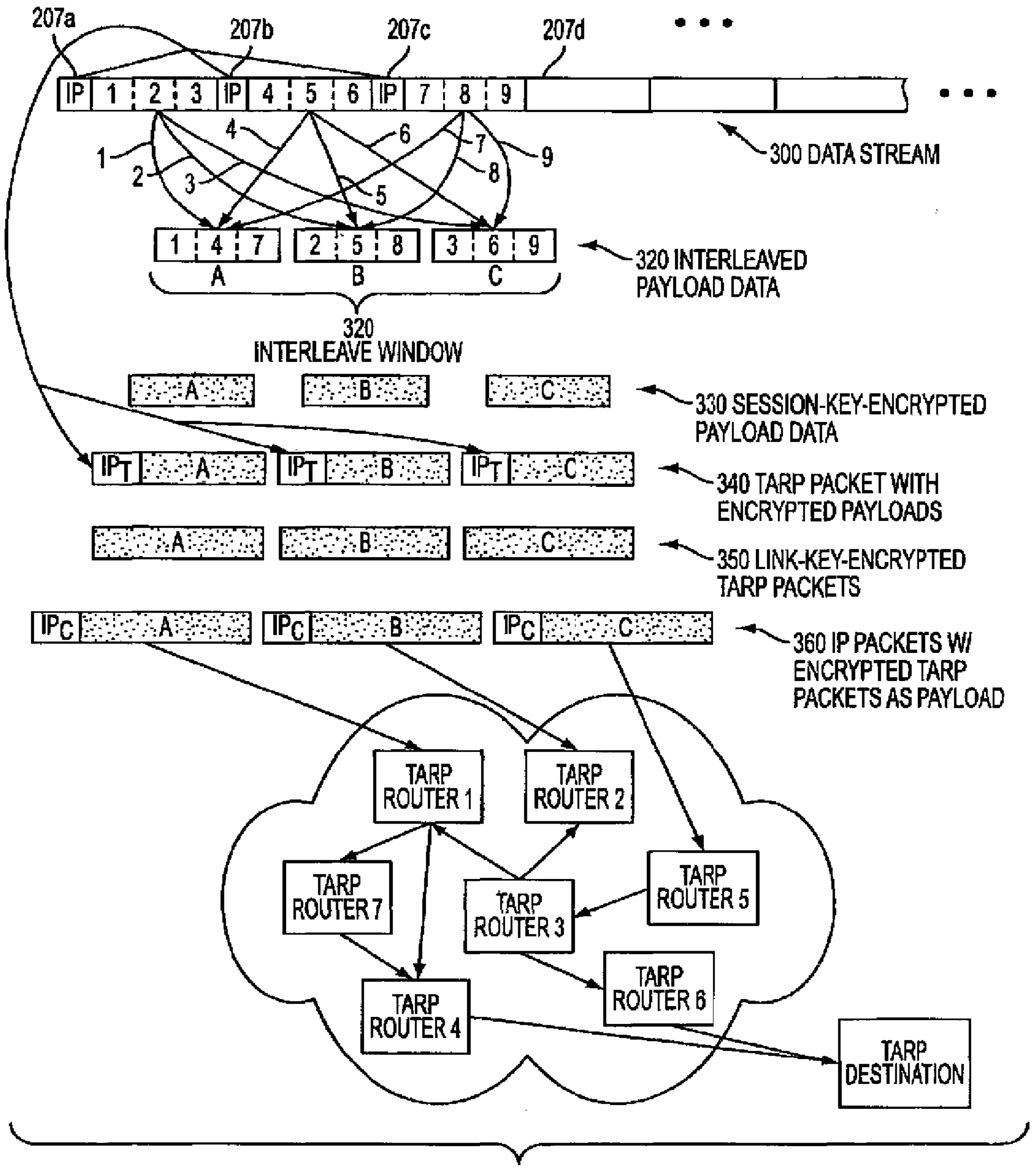

207a
207b
207c
207d
IP 1 2 3 IP 4 5 6 IP 7 8 9
300 DATA STREAM
1 4 7
2 5 8
3 6 9
A
B
C
320 INTERLEAVED PAYLOAD DATA
320 INTERLEAVE WINDOW
330 SESSION-KEY-ENCRYPTED PAYLOAD DATA
IPT A IPT B IPT C
340 TARP PACKET WITH ENCRYPTED PAYLOADS
350 LINK-KEY-ENCRYPTED TARP PACKETS
IPC A IPC B IPC C
360 IP PACKETS W/ ENCRYPTED TARP PACKETS AS PAYLOAD
TARP ROUTER 1
TARP ROUTER 2
TARP ROUTER 7
TARP ROUTER 3
TARP ROUTER 5
TARP ROUTER 6
TARP ROUTER 4
TARP DESTINATION

FIG. 3A

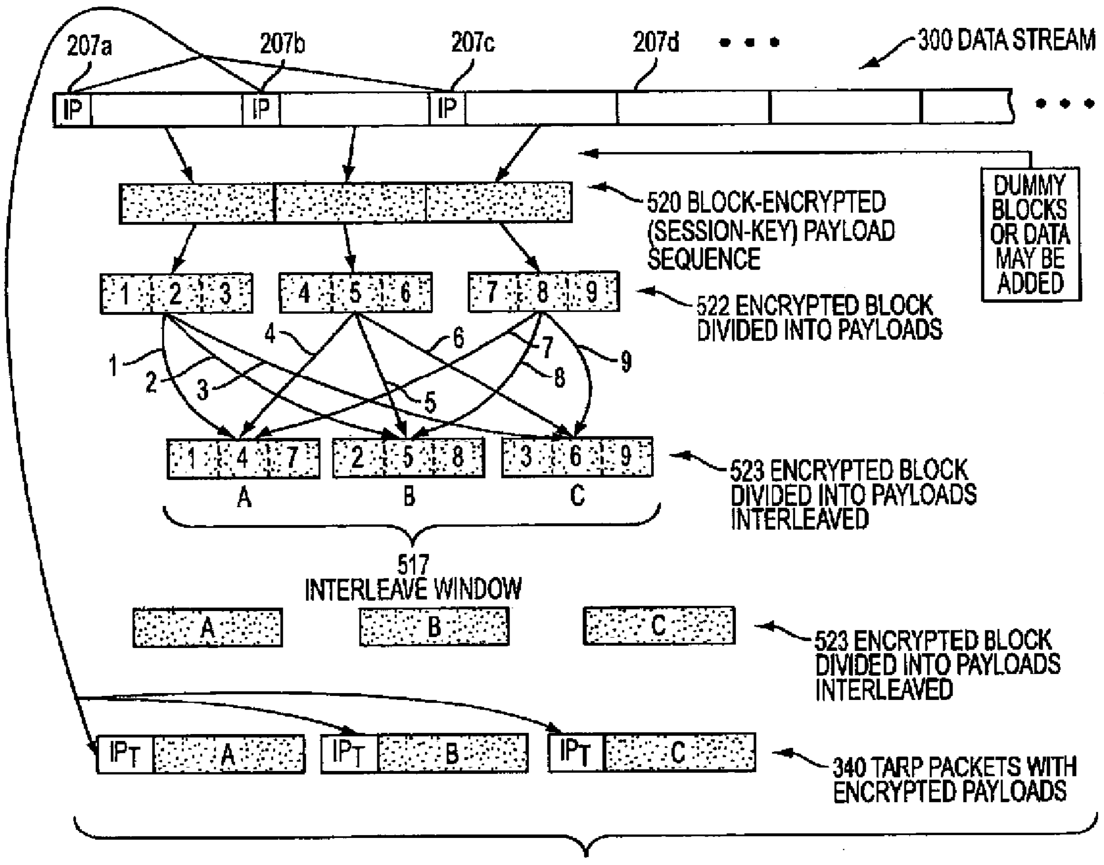

207a
207b
207c
207d
300 DATA STREAM
IP
IP
IP
DUMMY BLOCKS OR DATA MAY BE ADDED
520 BLOCK-ENCRYPTED (SESSION-KEY) PAYLOAD SEQUENCE
1 2 3
4 5 6
7 8 9
522 ENCRYPTED BLOCK DIVIDED INTO PAYLOADS
1 4 7
2 5 8
3 6 9
A
B
C
523 ENCRYPTED BLOCK DIVIDED INTO PAYLOADS INTERLEAVED
517 INTERLEAVE WINDOW
A
B
C
523 ENCRYPTED BLOCK DIVIDED INTO PAYLOADS INTERLEAVED
IPT A
IPT B
IPT C
340 TARP PACKETS WITH ENCRYPTED PAYLOADS

FIG. 3B

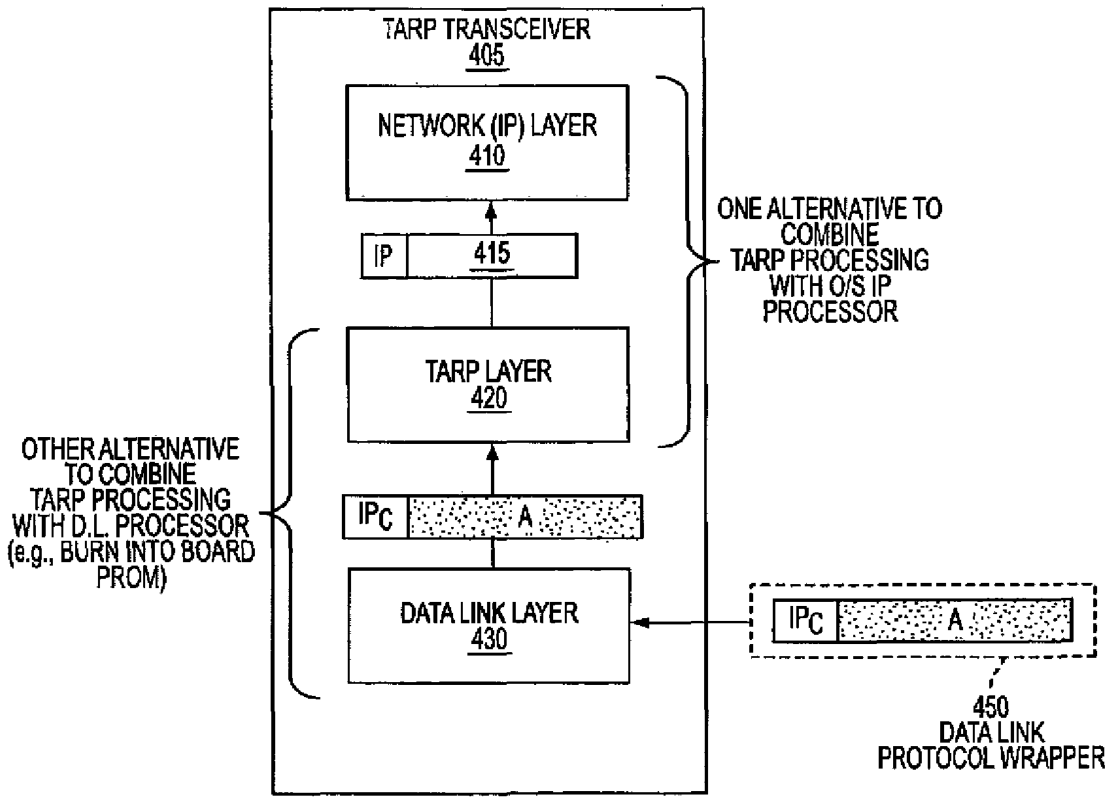

TARP TRANSCEIVER
405
NETWORK (IP) LAYER
410
IP
415
TARP LAYER
420
IPC
A
DATA LINK LAYER
430
ONE ALTERNATIVE TO COMBINE TARP PROCESSING WITH O/S IP PROCESSOR
OTHER ALTERNATIVE TO COMBINE TARP PROCESSING WITH D.L. PROCESSOR (e.g., BURN INTO BOARD PROM)
IPC
A
450
DATA LINK PROTOCOL WRAPPER

FIG. 4

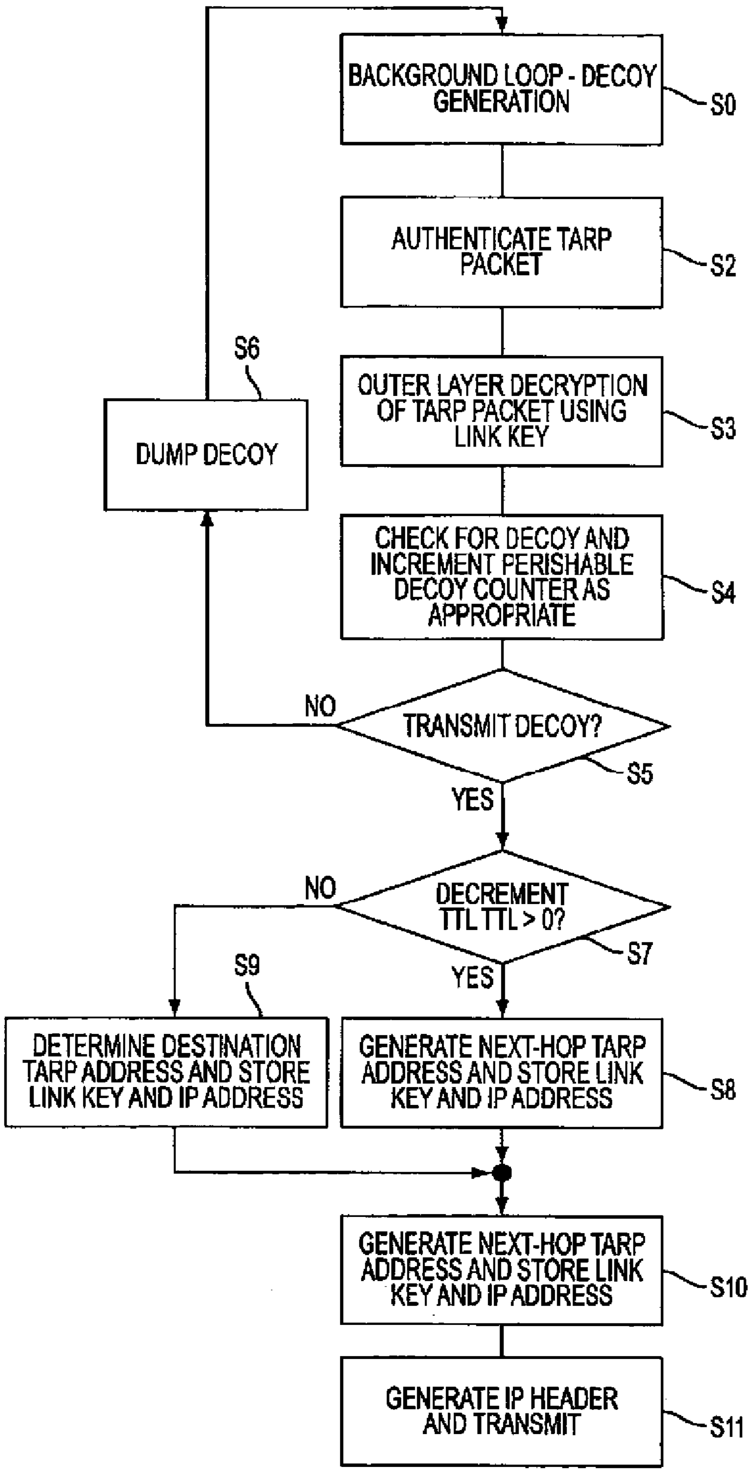

BACKGROUND LOOP - DECOY GENERATION
S0
AUTHENTICATE TARP PACKET
S2
OUTER LAYER DECRYPTION OF TARP PACKET USING LINK KEY
S3
CHECK FOR DECOY AND INCREMENT PERISHABLE DECOY COUNTER AS APPROPRIATE
S4
S6
DUMP DECOY
NO
TRANSMIT DECOY?
S5
YES
NO
DECREMENT TTL TTL > 0?
S7
YES
S9
DETERMINE DESTINATION TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
GENERATE NEXT-HOP TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
S8
GENERATE NEXT-HOP TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
S10
GENERATE IP HEADER AND TRANSMIT
S11

FIG. 5

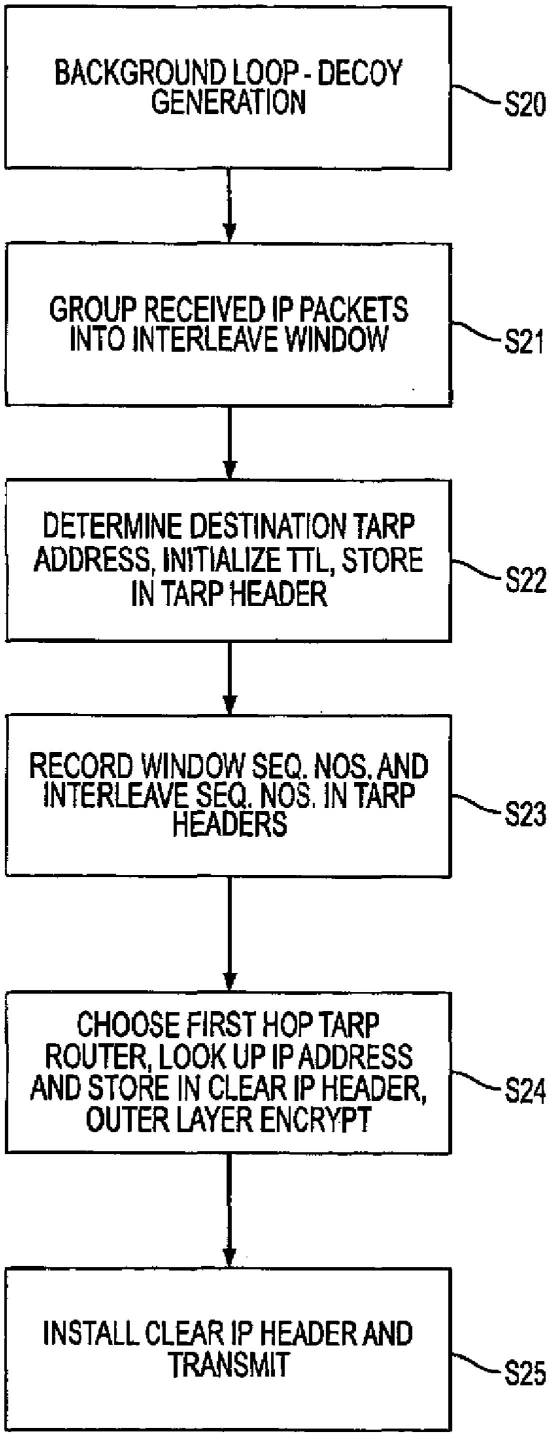

BACKGROUND LOOP - DECOY GENERATION
S20
GROUP RECEIVED IP PACKETS INTO INTERLEAVE WINDOW
S21
DETERMINE DESTINATION TARP ADDRESS, INITIALIZE TTL, STORE IN TARP HEADER
S22
RECORD WINDOW SEQ. NOS. AND INTERLEAVE SEQ. NOS. IN TARP HEADERS
S23
CHOOSE FIRST HOP TARP ROUTER, LOOK UP IP ADDRESS AND STORE IN CLEAR IP HEADER, OUTER LAYER ENCRYPT
S24
INSTALL CLEAR IP HEADER AND TRANSMIT
S25

FIG. 6

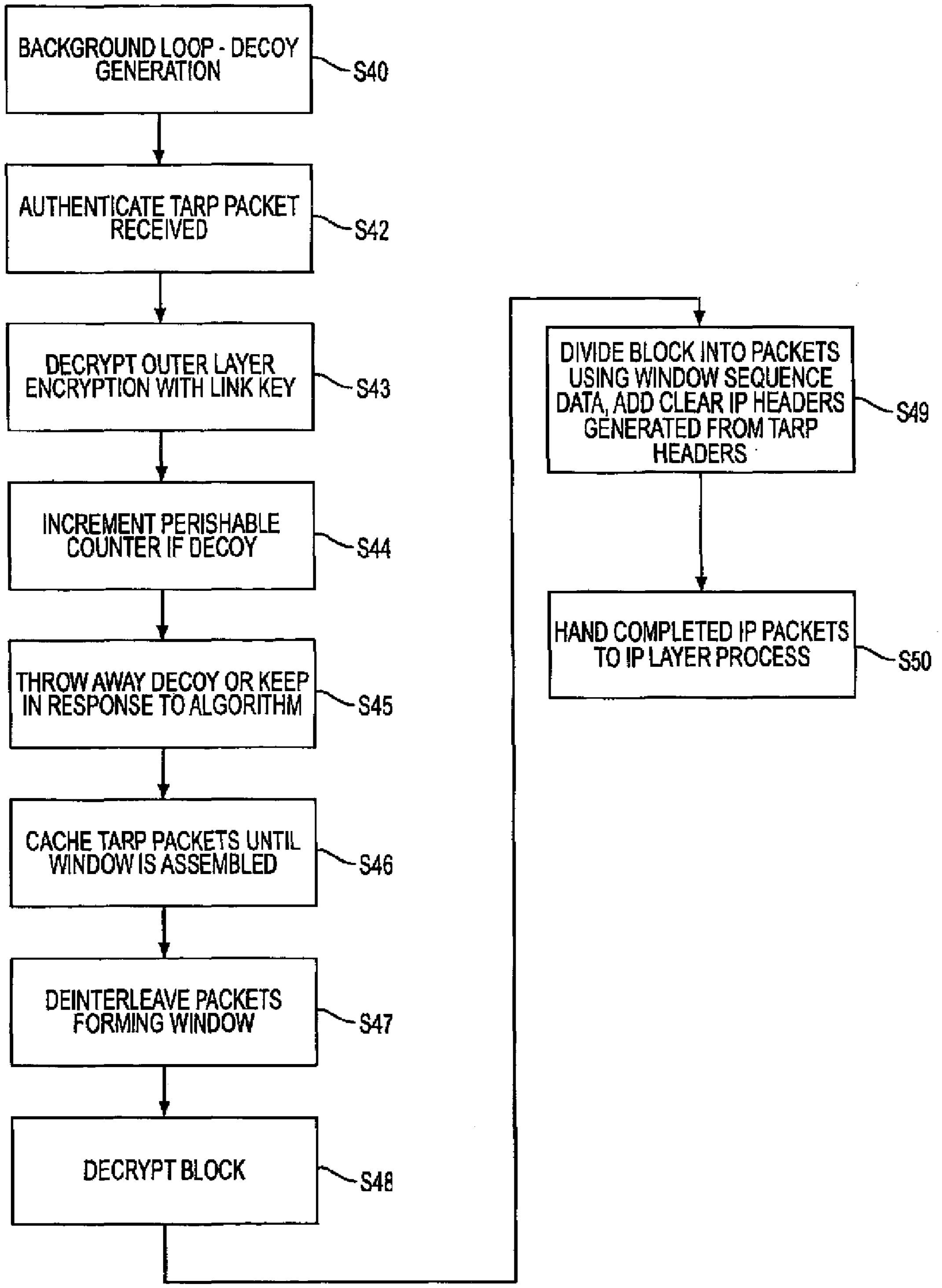

BACKGROUND LOOP - DECOY GENERATION
S40
AUTHENTICATE TARP PACKET RECEIVED
S42
DECRYPT OUTER LAYER ENCRYPTION WITH LINK KEY
S43
INCREMENT PERISHABLE COUNTER IF DECOY
S44
THROW AWAY DECOY OR KEEP IN RESPONSE TO ALGORITHM
S45
CACHE TARP PACKETS UNTIL WINDOW IS ASSEMBLED
S46
DEINTERLEAVE PACKETS FORMING WINDOW
S47
DECRYPT BLOCK
S48
DIVIDE BLOCK INTO PACKETS USING WINDOW SEQUENCE DATA, ADD CLEAR IP HEADERS GENERATED FROM TARP HEADERS
S49
HAND COMPLETED IP PACKETS TO IP LAYER PROCESS
S50

FIG. 7

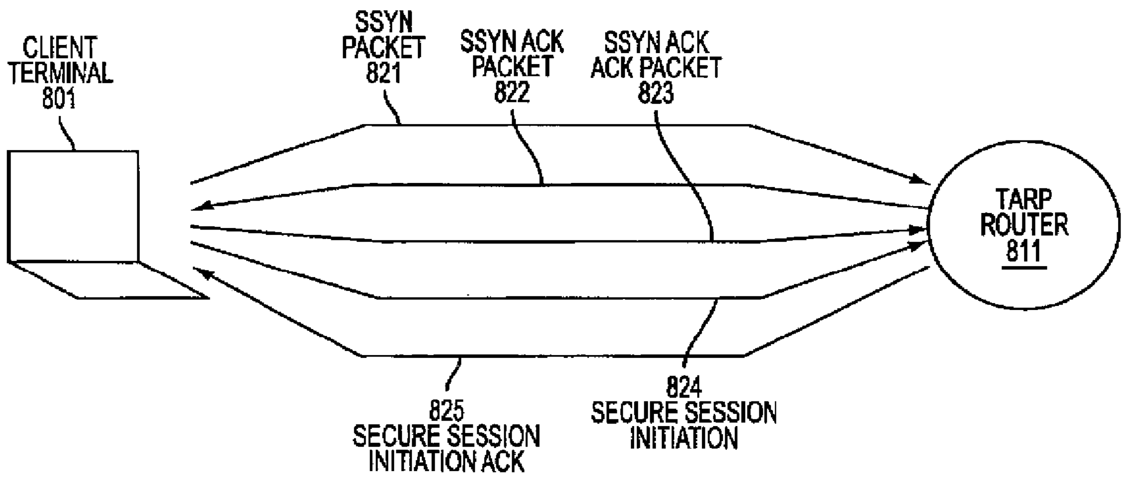

CLIENT TERMINAL 801
SSYN PACKET 821
SSYN ACK PACKET 822
SSYN ACK ACK PACKET 823
TARP ROUTER 811
825 SECURE SESSION INITIATION ACK
824 SECURE SESSION INITIATION

FIG. 8

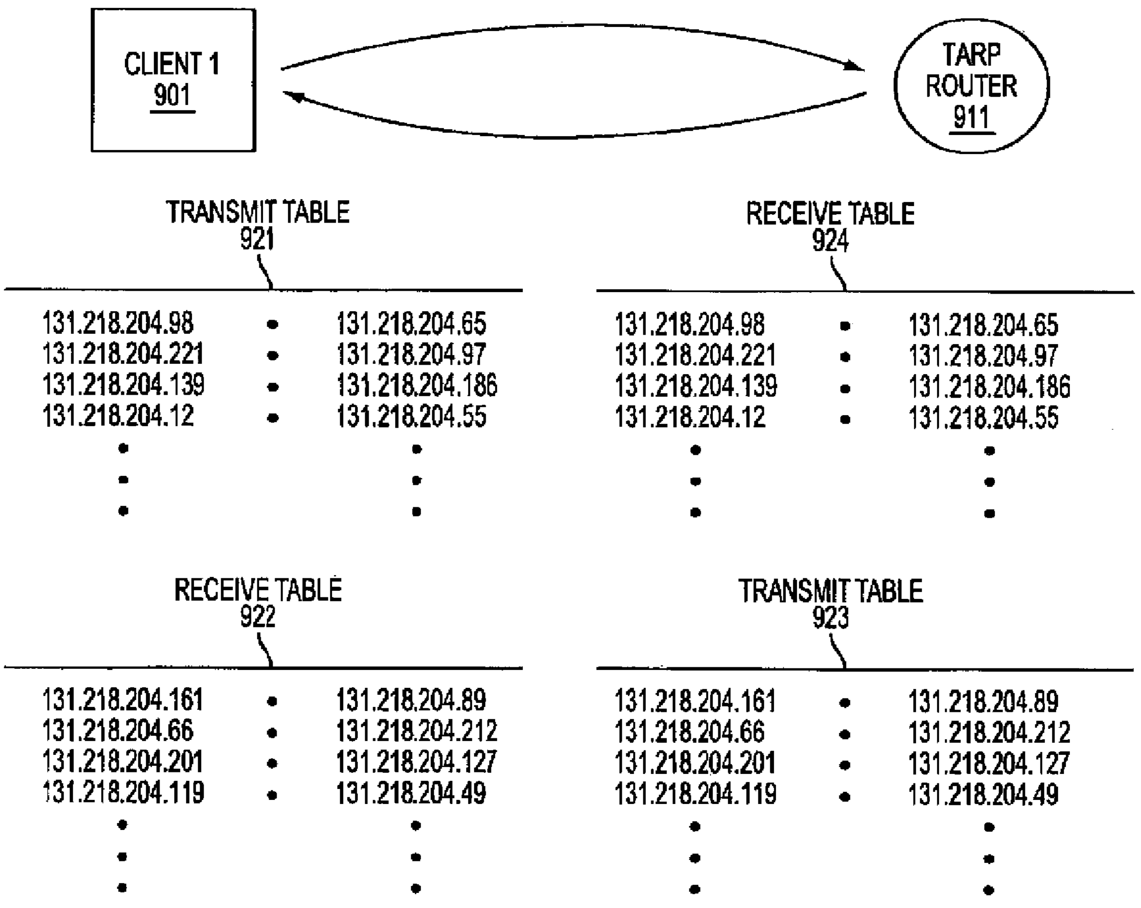

CLIENT 1
901
TARP
ROUTER
911
TRANSMIT TABLE
921
131.218.204.98 • 131.218.204.65
131.218.204.221 • 131.218.204.97
131.218.204.139 • 131.218.204.186
131.218.204.12 • 131.218.204.55
RECEIVE TABLE
924
131.218.204.98 • 131.218.204.65
131.218.204.221 • 131.218.204.97
131.218.204.139 • 131.218.204.186
131.218.204.12 • 131.218.204.55
RECEIVE TABLE
922
131.218.204.161 • 131.218.204.89
131.218.204.66 • 131.218.204.212
131.218.204.201 • 131.218.204.127
131.218.204.119 • 131.218.204.49
TRANSMIT TABLE
923
131.218.204.161 • 131.218.204.89
131.218.204.66 • 131.218.204.212
131.218.204.201 • 131.218.204.127
131.218.204.119 • 131.218.204.49

FIG. 9

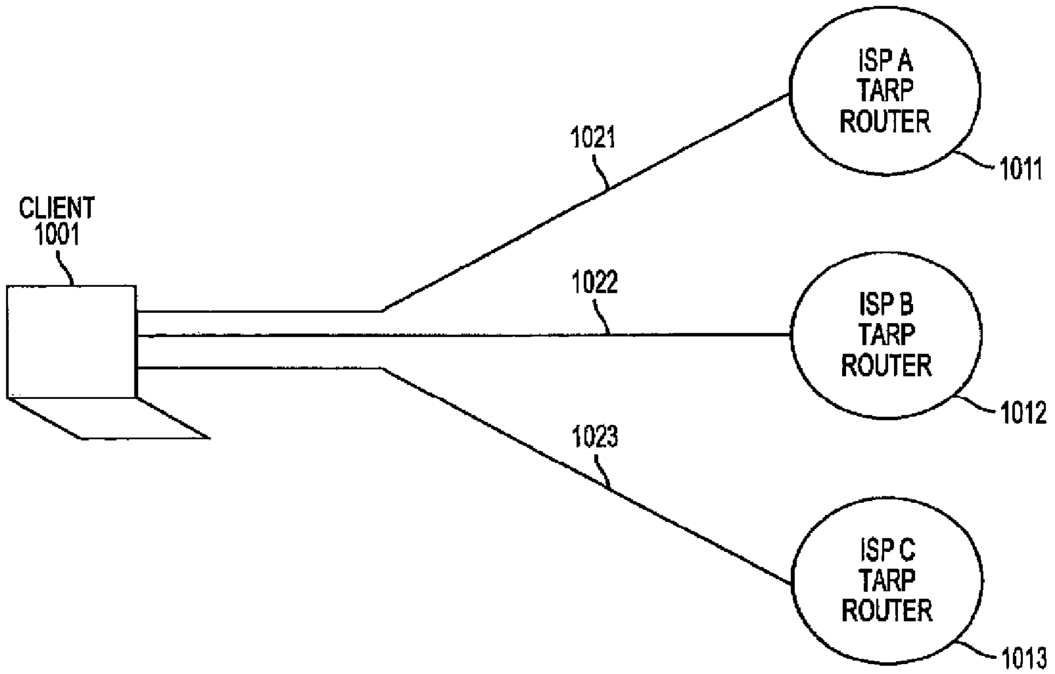

CLIENT
1001
1021
1022
1023
ISP A
TARP
ROUTER
1011
ISP B
TARP
ROUTER
1012
ISP C
TARP
ROUTER
1013

FIG. 10

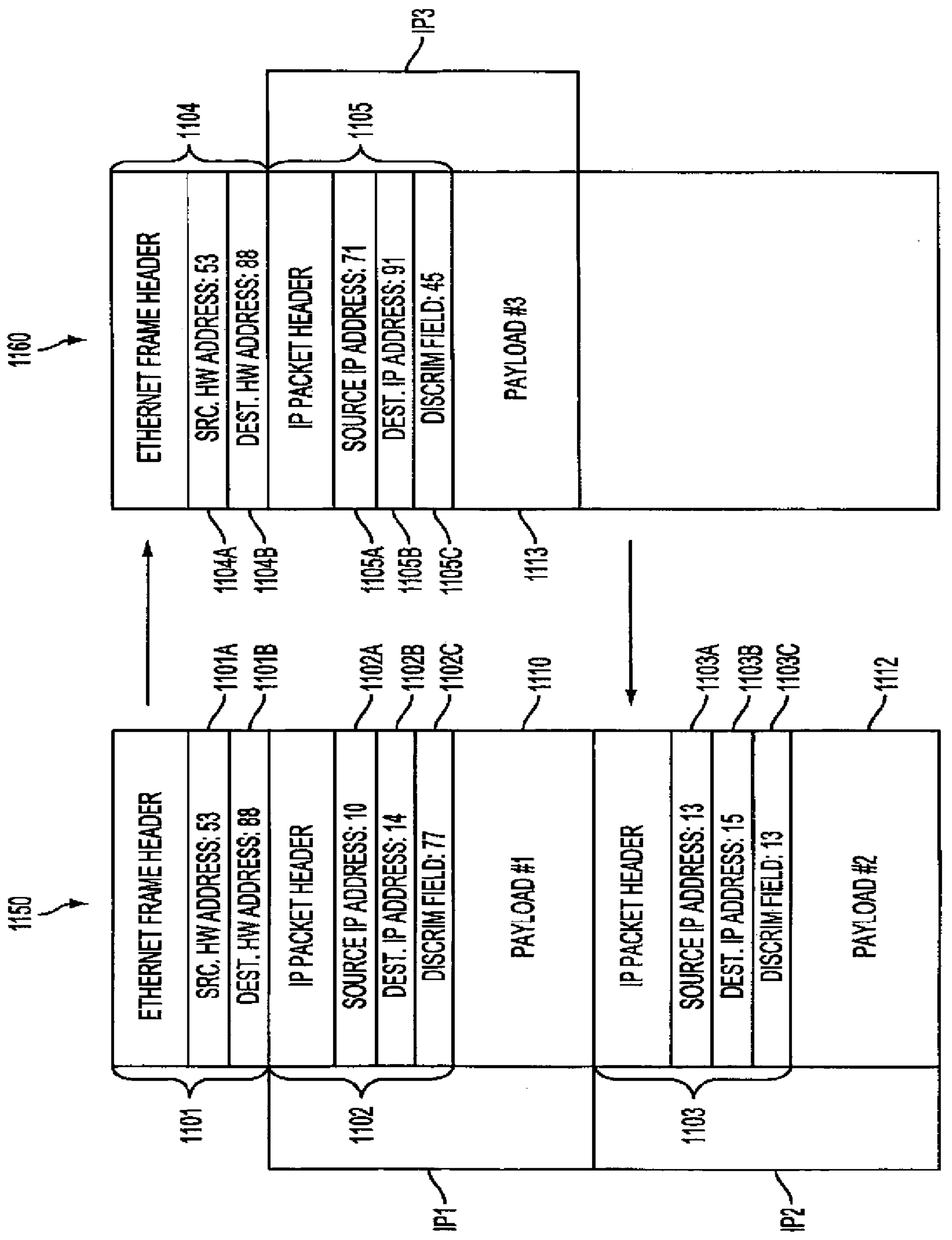

1150
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
IP PACKET HEADER
SOURCE IP ADDRESS: 10
DEST. IP ADDRESS: 14
DISCRIM FIELD: 77
PAYLOAD #1
IP PACKET HEADER
SOURCE IP ADDRESS: 13
DEST. IP ADDRESS: 15
DISCRIM FIELD: 13
PAYLOAD #2
1101
1102
1103
IP1
IP2
1101A
1101B
1102A
1102B
1102C
1110
1103A
1103B
1103C
1112
1160
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
IP PACKET HEADER
SOURCE IP ADDRESS: 71
DEST. IP ADDRESS: 91
DISCRIM FIELD: 45
PAYLOAD #3
1104
1105
IP3
1104A
1104B
1105A
1105B
1105C
1113

FIG. 11

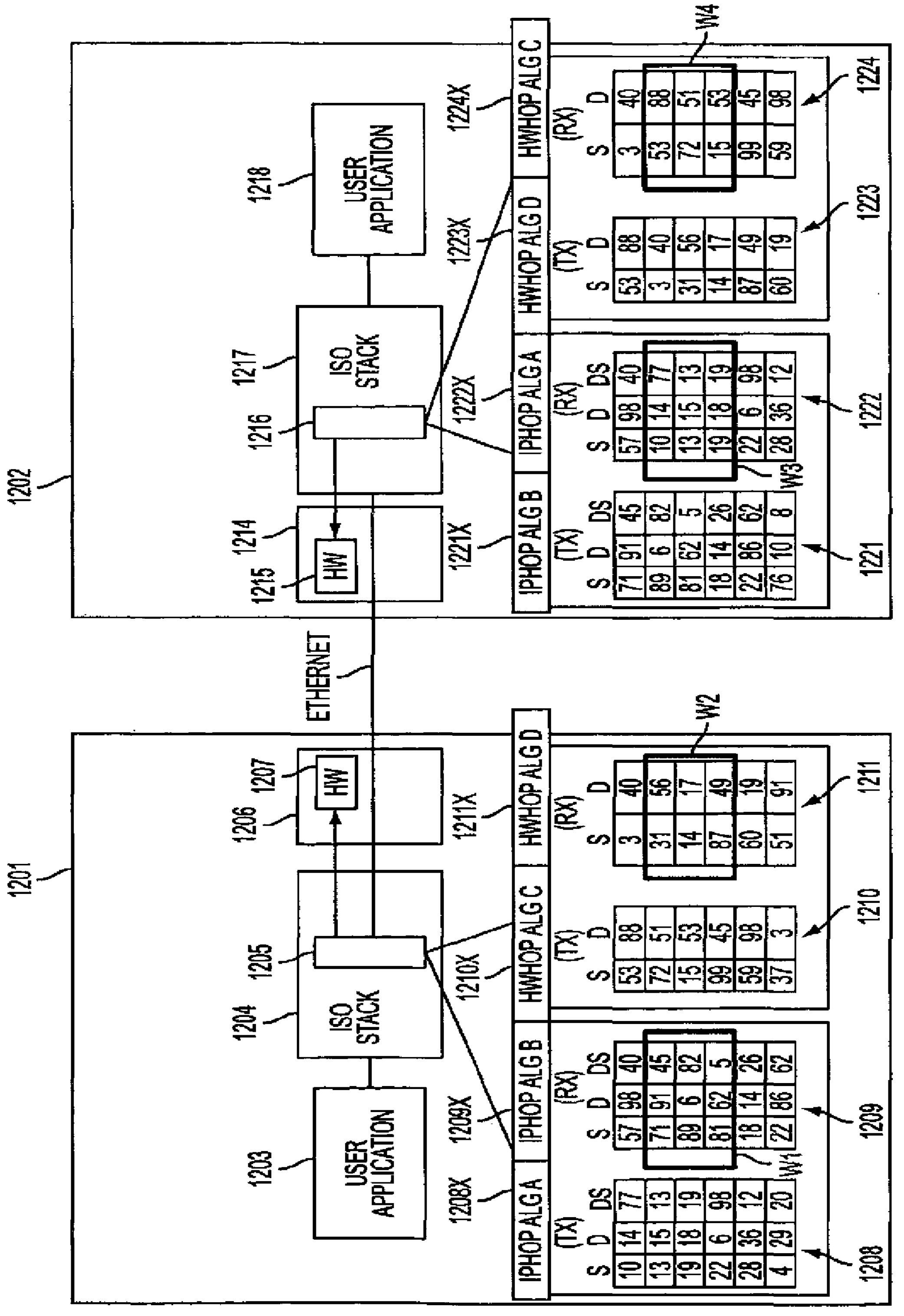

1201
1202
1203
USER APPLICATION
1204
ISO STACK
1205
1206
1207
HW
ETHERNET
1214
1215
HW
1216
1217
ISO STACK
1218
USER APPLICATION
1208X
IPHOP ALG A
(TX) S D DS
10 14 77
13 15 13
19 18 19
22 6 98
28 36 12
4 29 20
1208
1209X
IPHOP ALG B
(RX) S D DS
57 98 40
71 91 45
89 6 82
81 62 5
18 14 26
22 86 62
W1
1209
1210X
HWHOP ALG C
(TX) S D
53 88
72 51
15 53
99 45
59 98
37 3
1210
1211X
HWHOP ALG D
(RX) S D
3 40
31 56
14 17
87 49
60 19
51 91
W2
1211
1221X
IPHOP ALG B
(TX) S D DS
71 91 45
89 6 82
81 62 5
18 14 26
22 86 62
76 10 8
1221
1222X
IPHOP ALG A
(RX) S D DS
57 98 40
10 14 77
13 15 13
19 18 19
22 6 98
28 36 12
W3
1222
1223X
HWHOP ALG D
(TX) S D
53 88
3 40
31 56
14 17
87 49
60 19
1223
1224X
HWHOP ALG C
(RX) S D
3 40
53 88
72 51
15 53
99 45
59 98
W4
1224

FIG. 12A

MODE OR EMBODIMENT	HARDWARE ADDRESSES	IP ADDRESSES	DISCRIMINATOR FIELD VALUES
1. PROMISCUOUS	SAME FOR ALL NODES OR COMPLETELY RANDOM	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
2. PROMISCUOUS PER VPN	FIXED FOR EACH VPN	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
3. HARDWARE HOPPING	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC

FIG. 12B

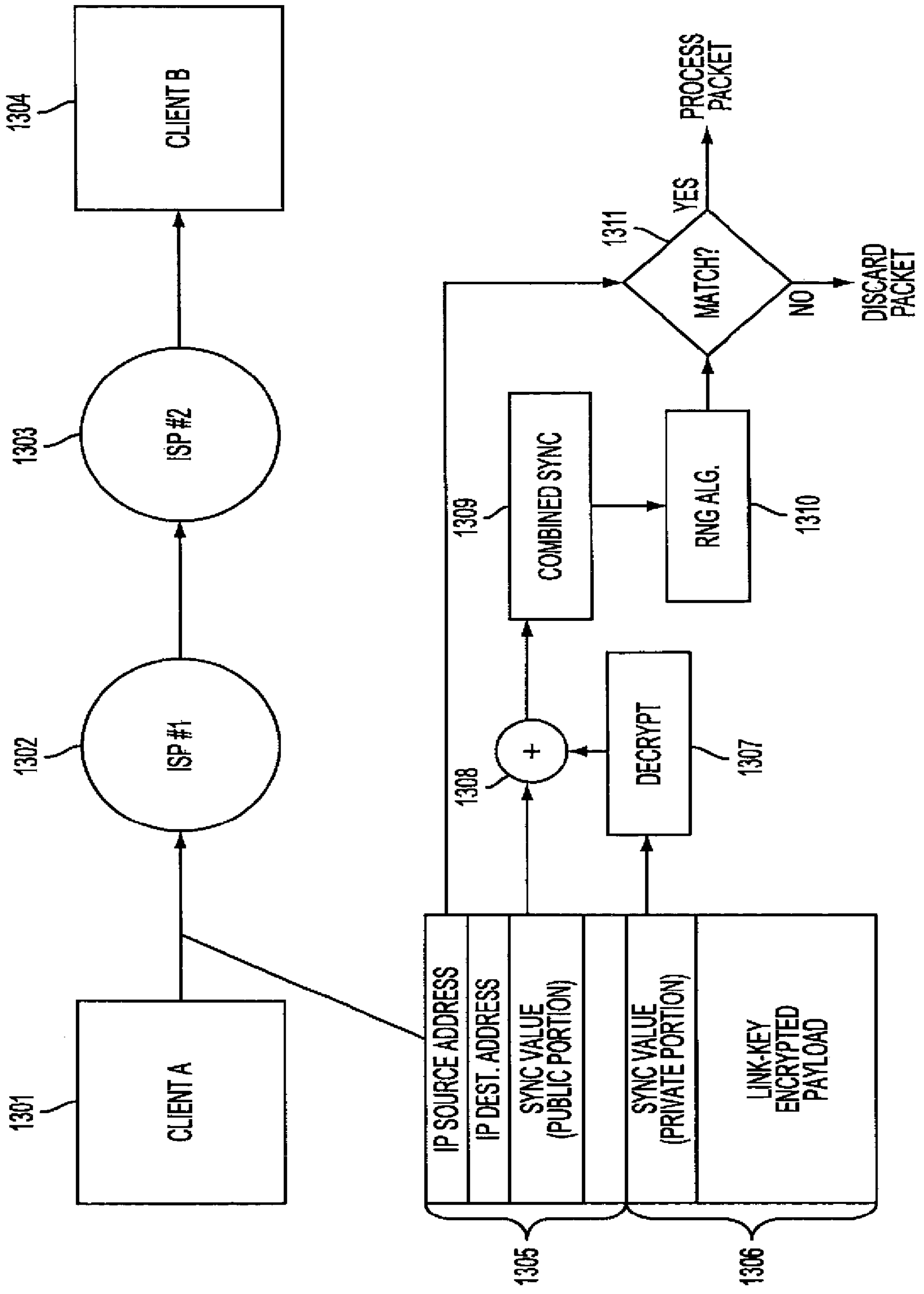
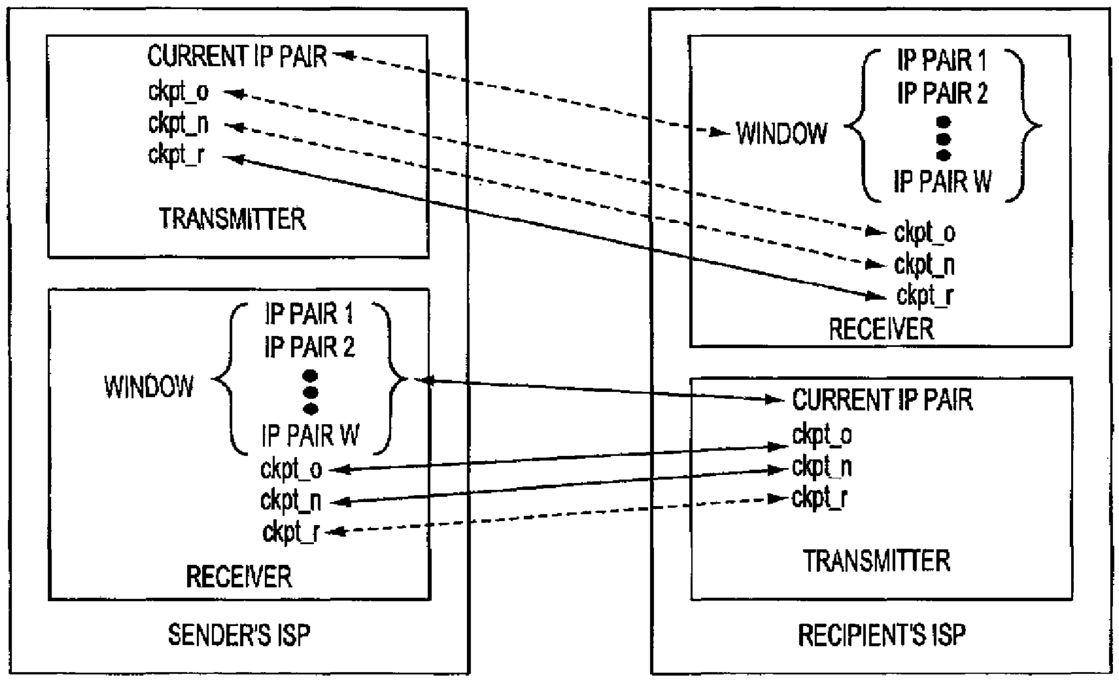

1301
CLIENT A
1302
ISP #1
1303
ISP #2
1304
CLIENT B
IP SOURCE ADDRESS
IP DEST. ADDRESS
SYNC VALUE (PUBLIC PORTION)
SYNC VALUE (PRIVATE PORTION)
LINK-KEY ENCRYPTED PAYLOAD
1305
1306
1307
DECRYPT
1308
+
1309
COMBINED SYNC
1310
RNG ALG.
1311
MATCH?
YES
PROCESS PACKET
NO
DISCARD PACKET

FIG. 13



KEPT IN SYNC FOR SENDER TO RECIPIENT SYNCHRONIZER	◄- - - - - - - - - - ►
KEPT IN SYNC FOR RECIPIENT TO SENDER SYNCHRONIZER	◄──────────►

FIG. 14

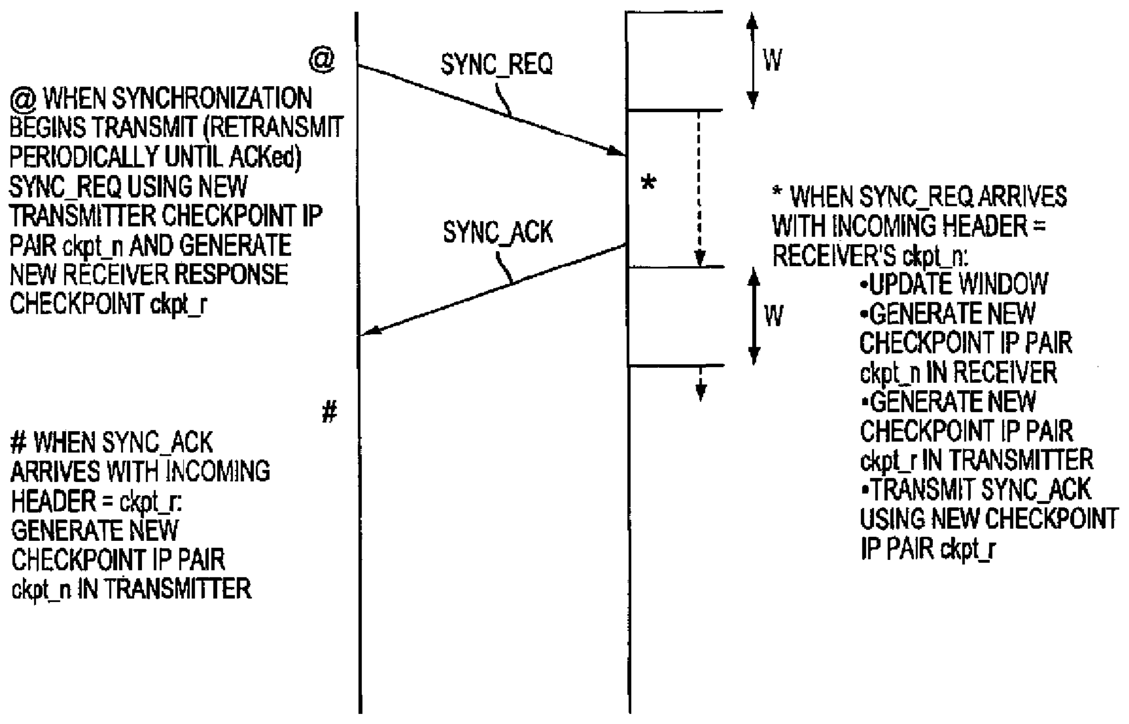

@
SYNC_REQ
W
@ WHEN SYNCHRONIZATION BEGINS TRANSMIT (RETRANSMIT PERIODICALLY UNTIL ACKed) SYNC_REQ USING NEW TRANSMITTER CHECKPOINT IP PAIR ckpt_n AND GENERATE NEW RECEIVER RESPONSE CHECKPOINT ckpt_r
*
SYNC_ACK
* WHEN SYNC_REQ ARRIVES WITH INCOMING HEADER = RECEIVER'S ckpt_n:
•UPDATE WINDOW
W
•GENERATE NEW CHECKPOINT IP PAIR ckpt_n IN RECEIVER
•GENERATE NEW CHECKPOINT IP PAIR ckpt_r IN TRANSMITTER
•TRANSMIT SYNC_ACK USING NEW CHECKPOINT IP PAIR ckpt_r
#
# WHEN SYNC_ACK ARRIVES WITH INCOMING HEADER = ckpt_r: GENERATE NEW CHECKPOINT IP PAIR ckpt_n IN TRANSMITTER

FIG. 15

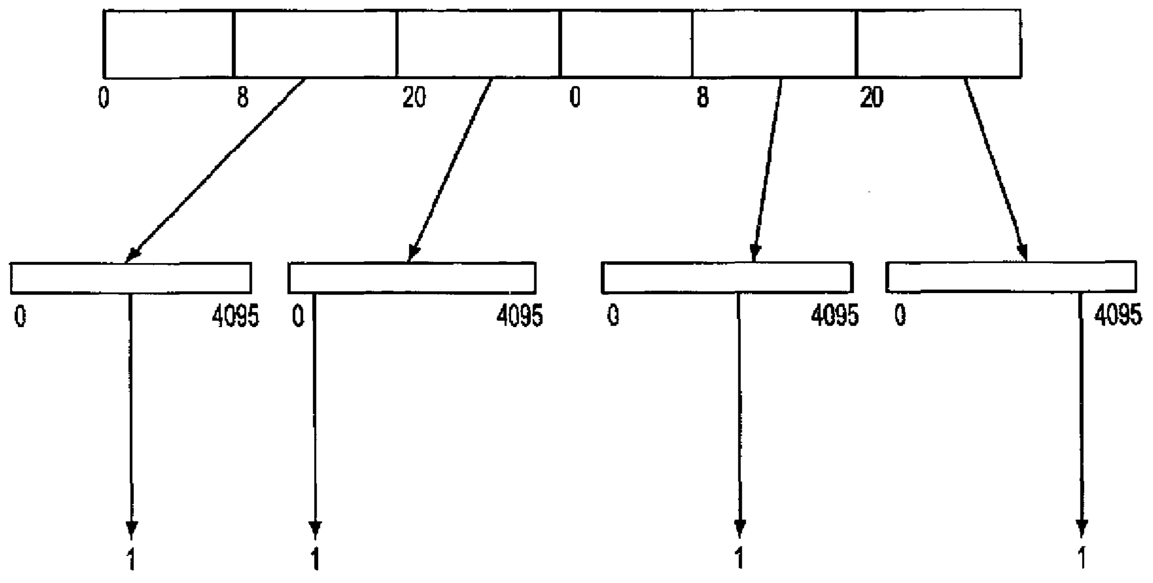


FIG. 16

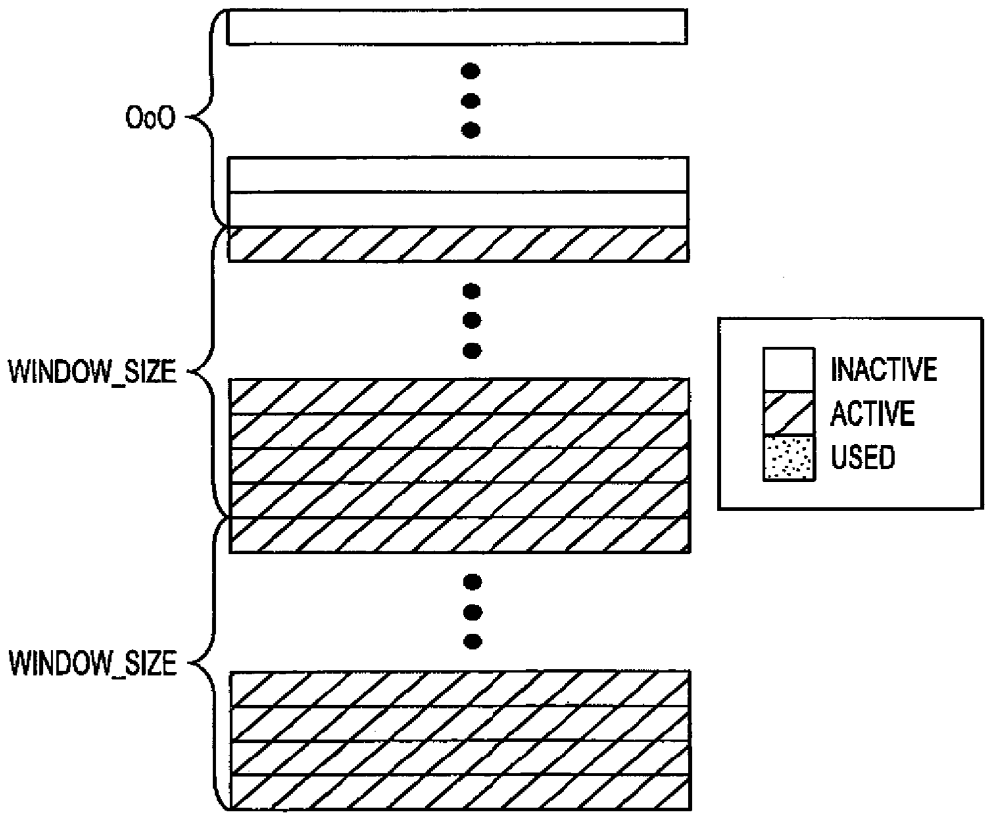


FIG. 17

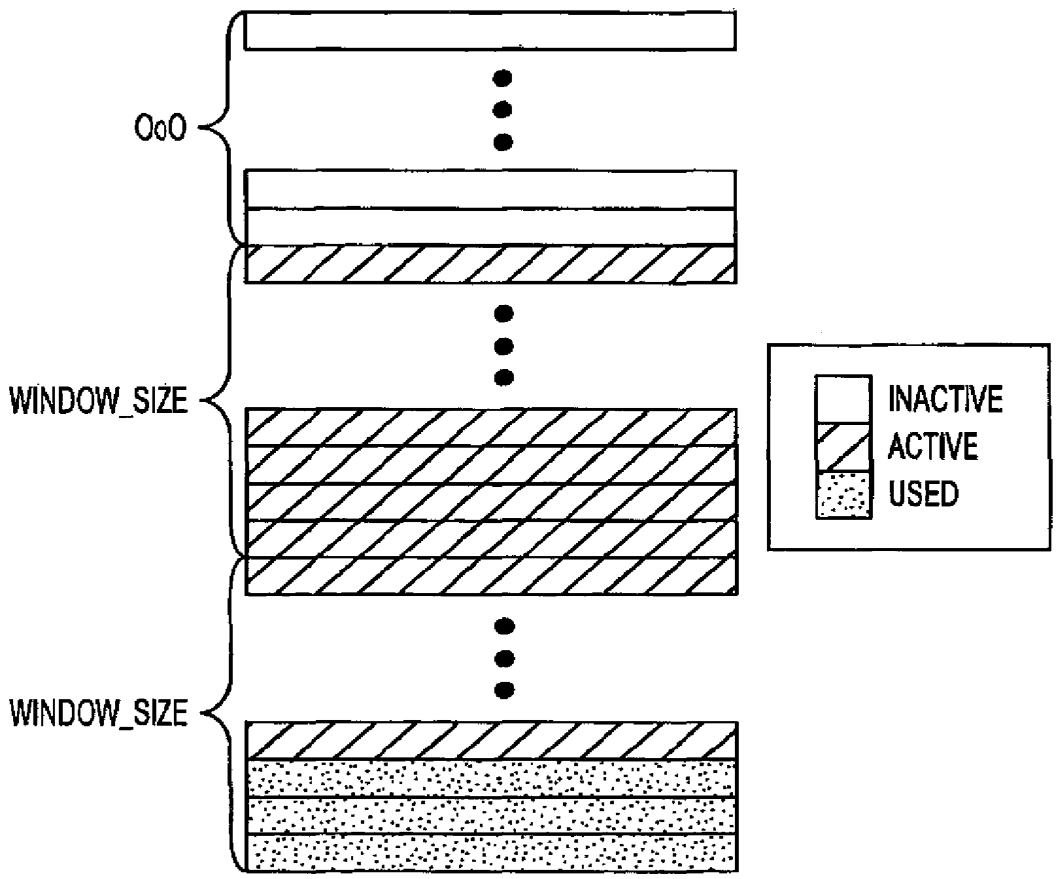


FIG. 18

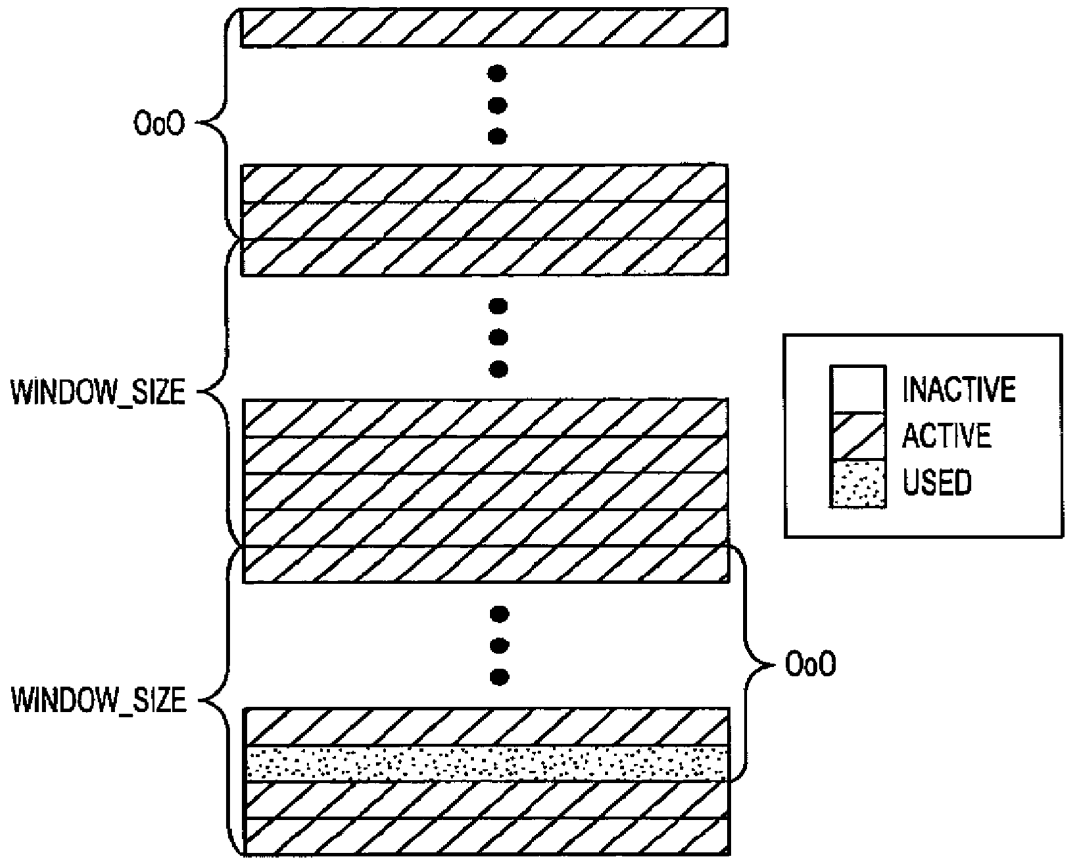


FIG. 19

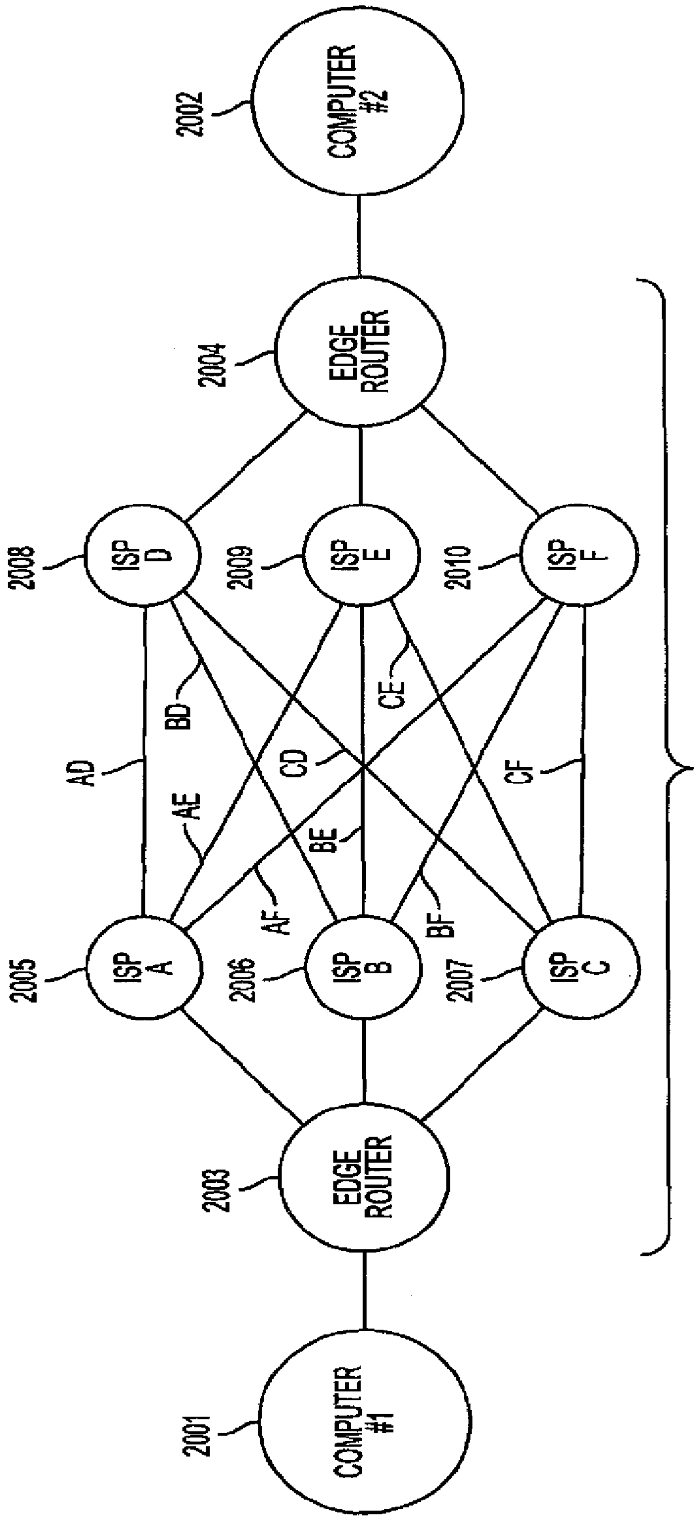

COMPUTER #2
2002
EDGE ROUTER
2004
ISP D
2008
ISP E
2009
ISP F
2010
AD
BD
CD
CE
CF
AE
BE
AF
BF
ISP A
2005
ISP B
2006
ISP C
2007
2011
EDGE ROUTER
2003
COMPUTER #1
2001

FIG. 20

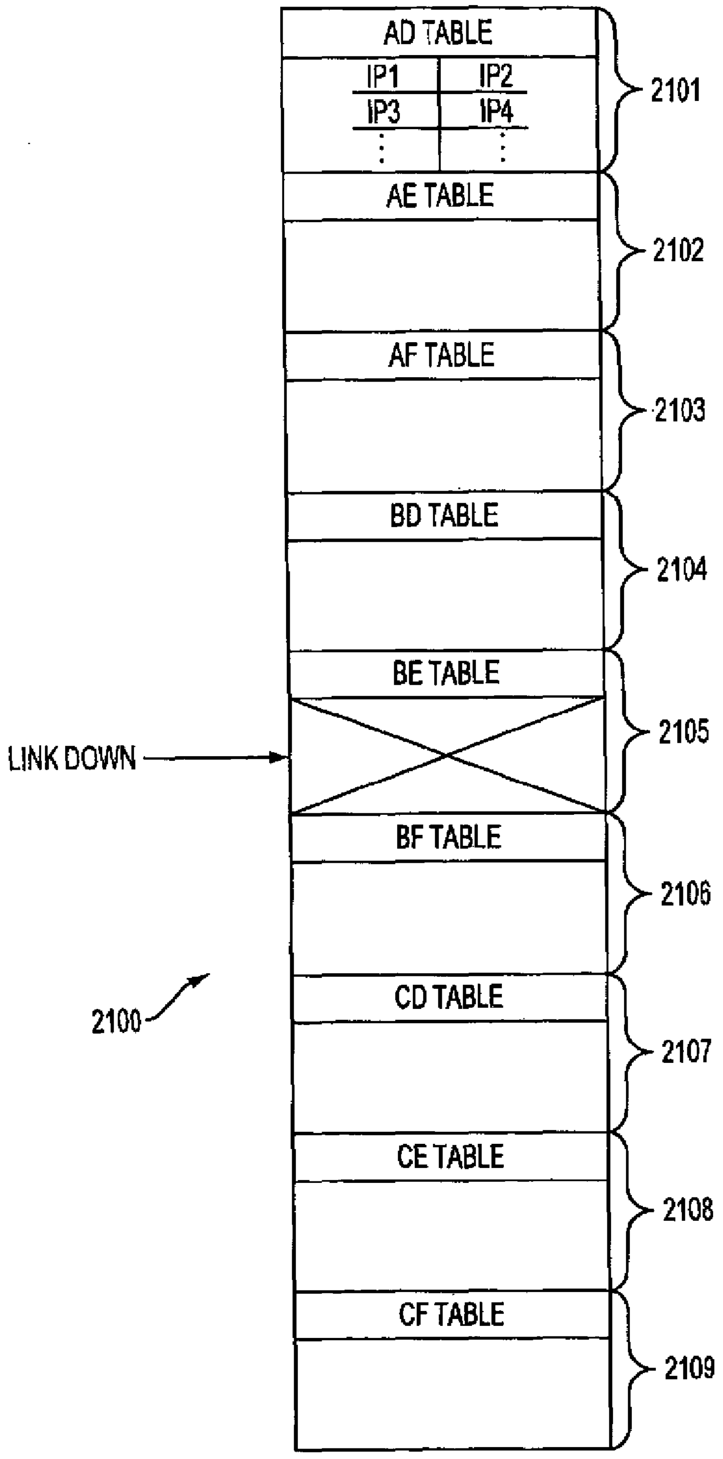

AD TABLE
IP1
IP2
IP3
IP4
2101
AE TABLE
2102
AF TABLE
2103
BD TABLE
2104
BE TABLE
2105
LINK DOWN
BF TABLE
2106
2100
CD TABLE
2107
CE TABLE
2108
CF TABLE
2109

FIG. 21

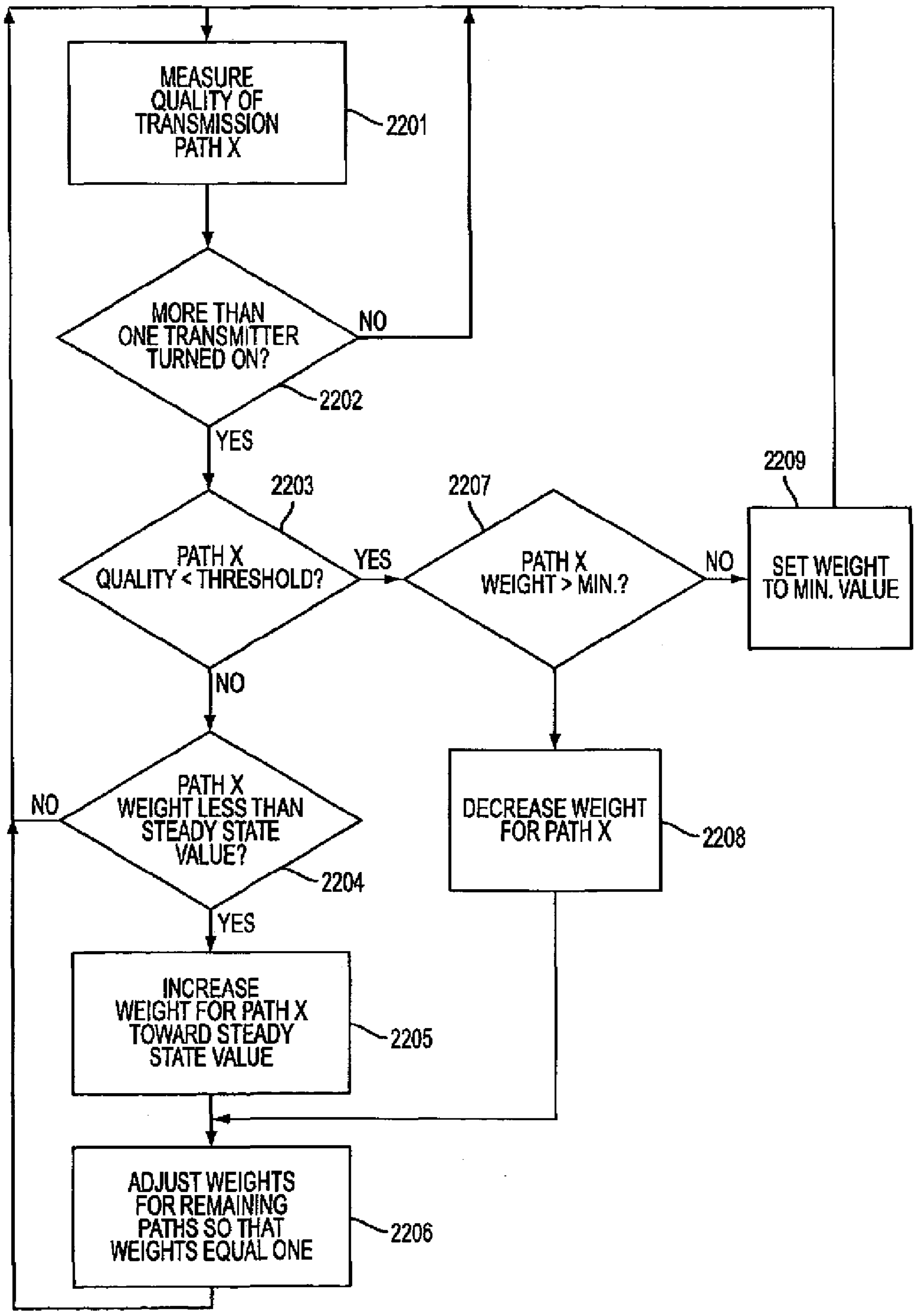

MEASURE QUALITY OF TRANSMISSION PATH X
2201
MORE THAN ONE TRANSMITTER TURNED ON?
NO
2202
YES
2203
PATH X QUALITY < THRESHOLD?
YES
2207
PATH X WEIGHT > MIN.?
NO
2209
SET WEIGHT TO MIN. VALUE
NO
PATH X WEIGHT LESS THAN STEADY STATE VALUE?
NO
2204
YES
DECREASE WEIGHT FOR PATH X
2208
INCREASE WEIGHT FOR PATH X TOWARD STEADY STATE VALUE
2205
ADJUST WEIGHTS FOR REMAINING PATHS SO THAT WEIGHTS EQUAL ONE
2206

FIG. 22A

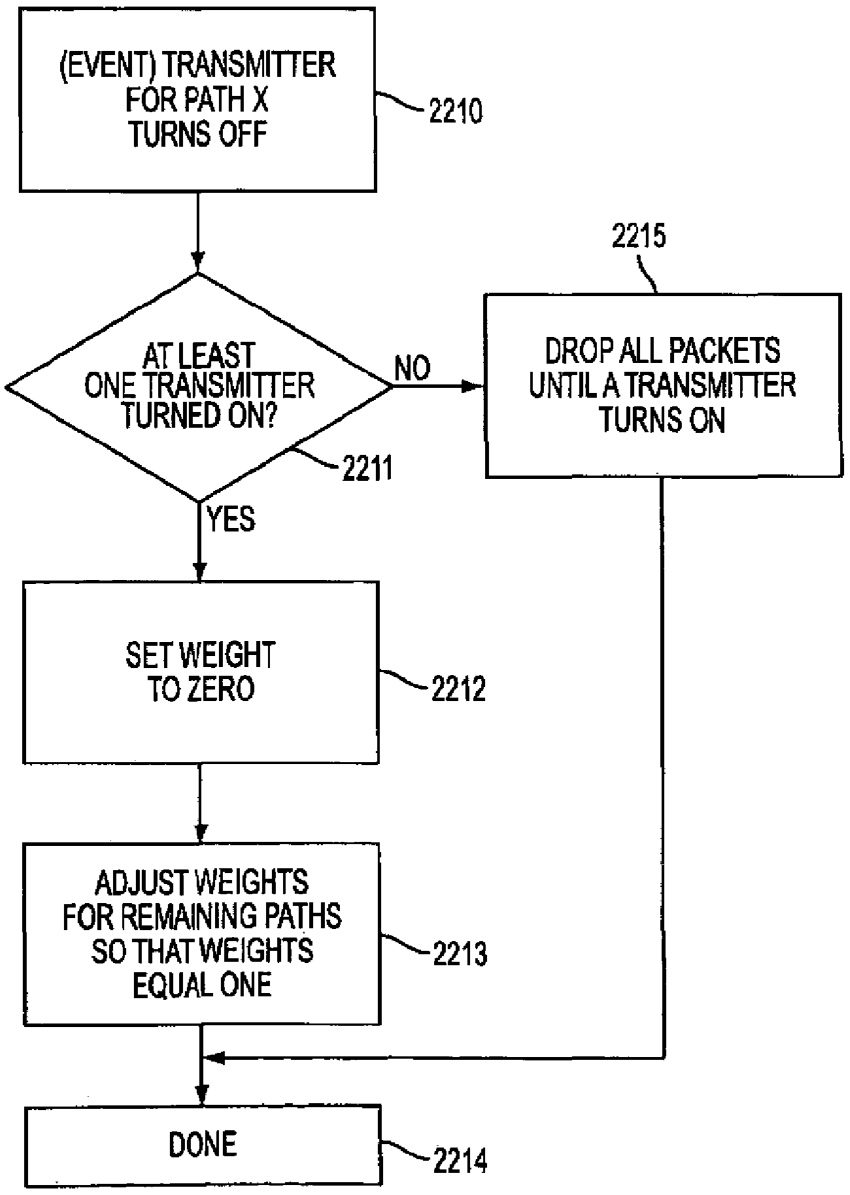

(EVENT) TRANSMITTER FOR PATH X TURNS OFF
2210
2215
AT LEAST ONE TRANSMITTER TURNED ON?
NO
DROP ALL PACKETS UNTIL A TRANSMITTER TURNS ON
2211
YES
SET WEIGHT TO ZERO
2212
ADJUST WEIGHTS FOR REMAINING PATHS SO THAT WEIGHTS EQUAL ONE
2213
DONE
2214

FIG. 22B

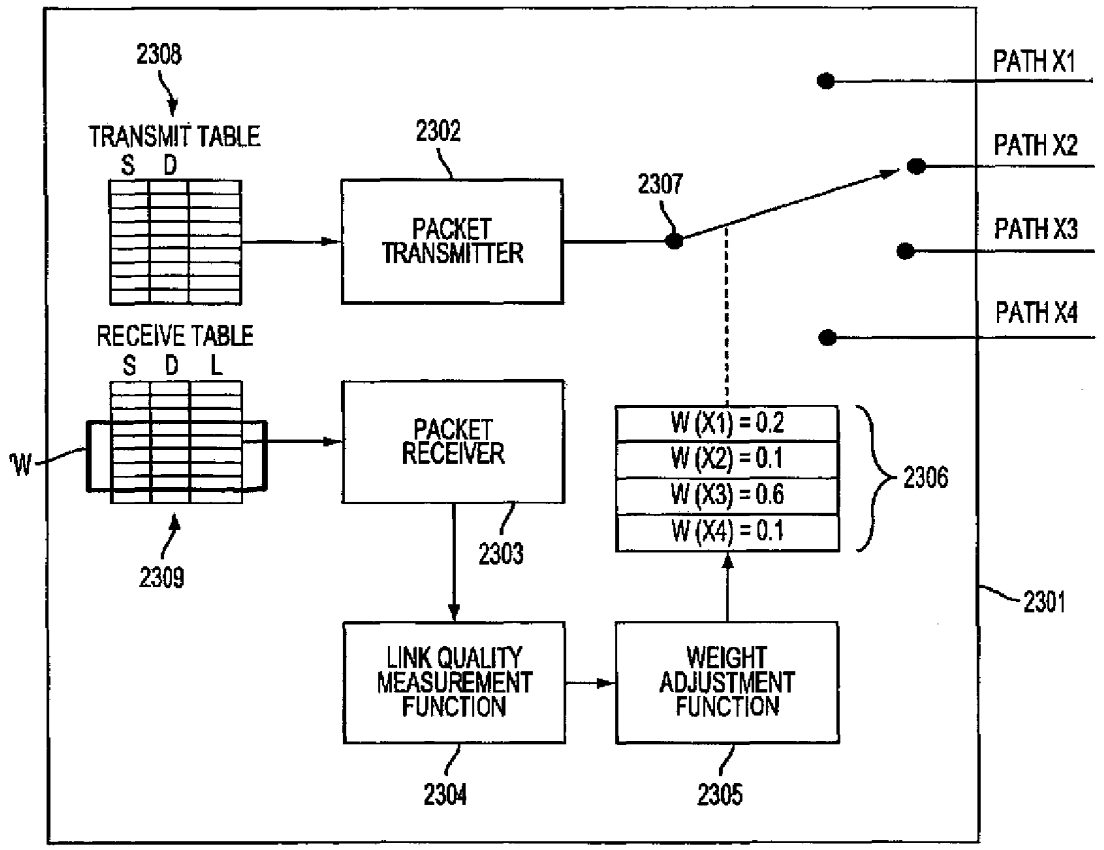

2308
TRANSMIT TABLE
S D
2302
PACKET TRANSMITTER
2307
PATH X1
PATH X2
PATH X3
PATH X4
RECEIVE TABLE
S D L
W
PACKET RECEIVER
W (X1) = 0.2
W (X2) = 0.1
W (X3) = 0.6
W (X4) = 0.1
2306
2309
2303
2301
LINK QUALITY MEASUREMENT FUNCTION
WEIGHT ADJUSTMENT FUNCTION
2304
2305

FIG. 23

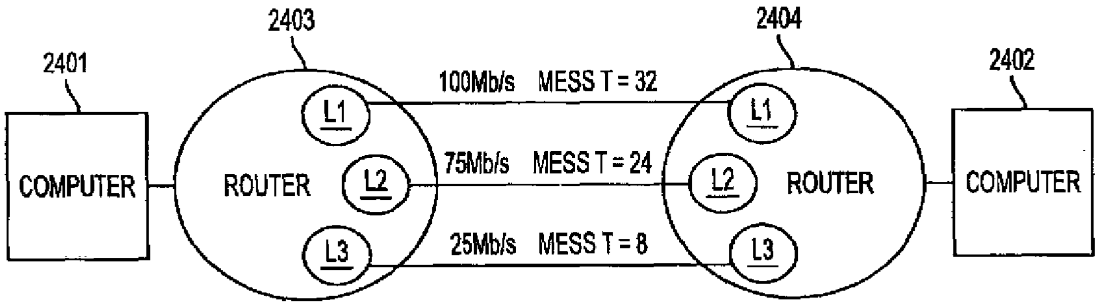

2401
COMPUTER
2403
ROUTER
L1
L2
L3
100Mb/s MESS T = 32
75Mb/s MESS T = 24
25Mb/s MESS T = 8
2404
L1
L2
L3
ROUTER
2402
COMPUTER

FIG. 24

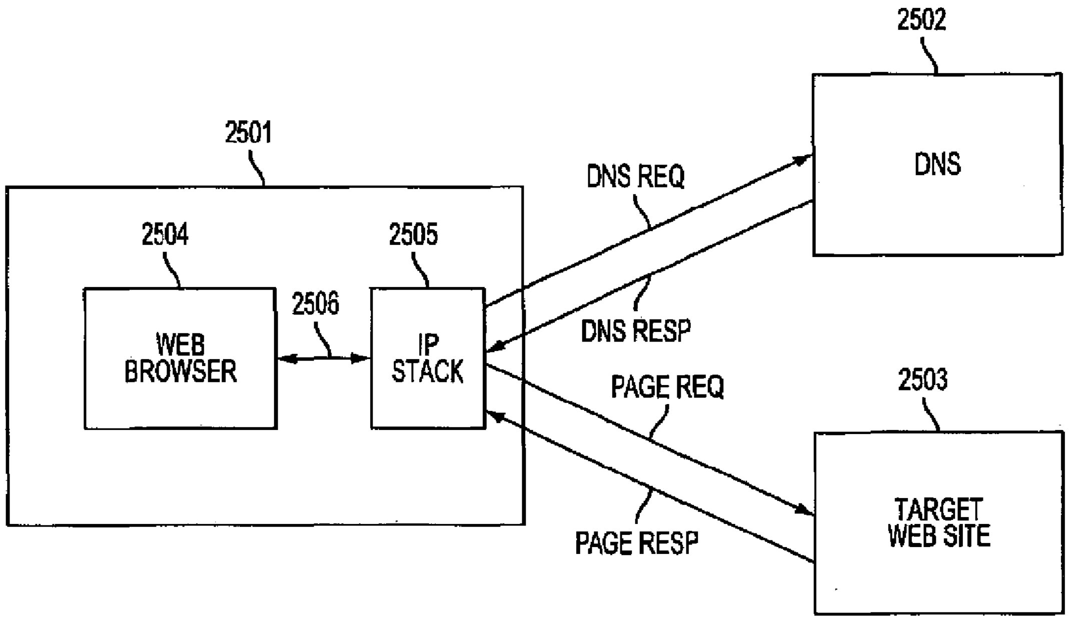


FIG. 25
(PRIOR ART)

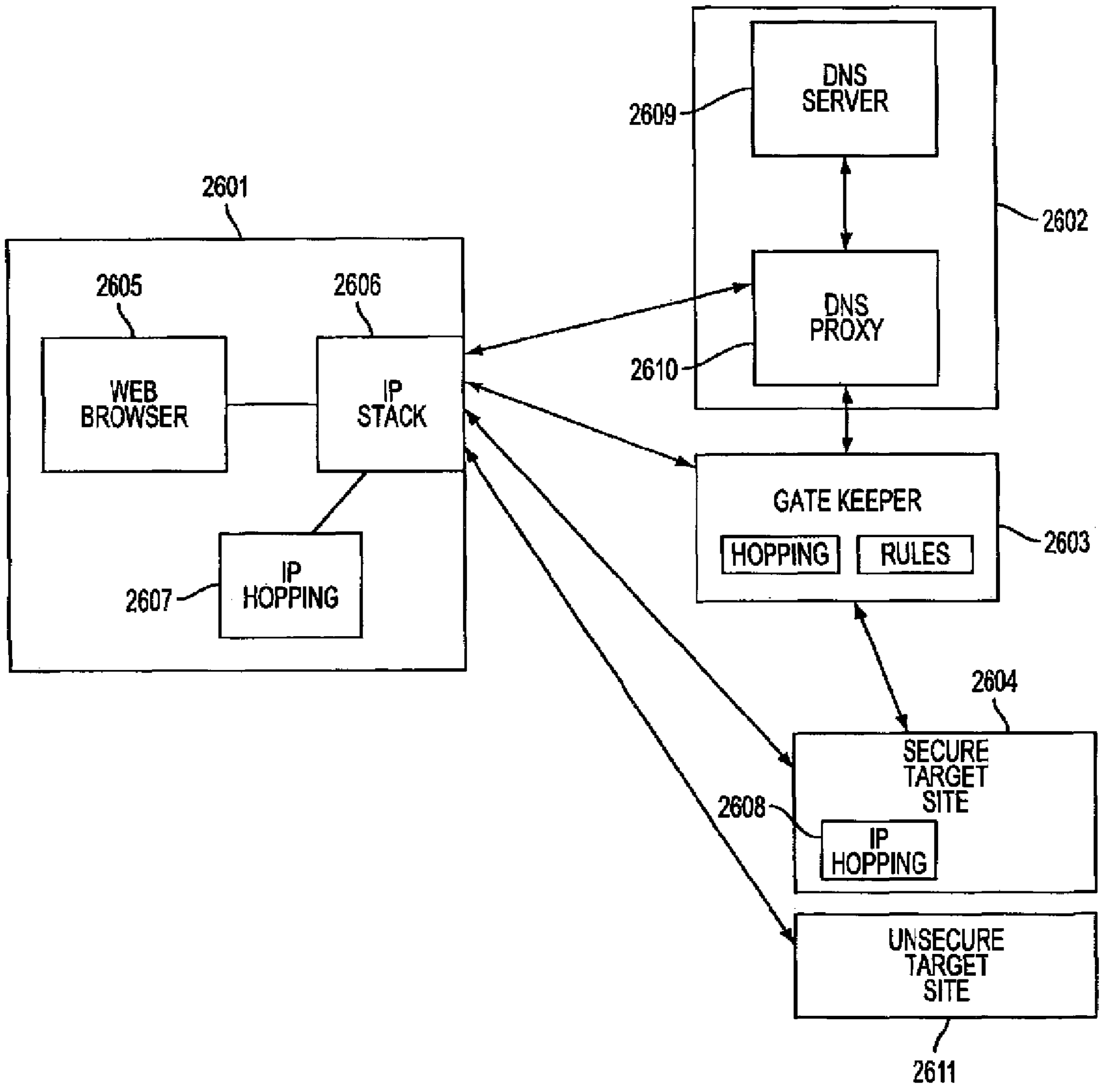

DNS SERVER
2609
2601
2602
2605
2606
DNS PROXY
WEB BROWSER
IP STACK
2610
GATE KEEPER
HOPPING
RULES
2603
IP HOPPING
2607
2604
SECURE TARGET SITE
2608
IP HOPPING
UNSECURE TARGET SITE
2611

FIG. 26

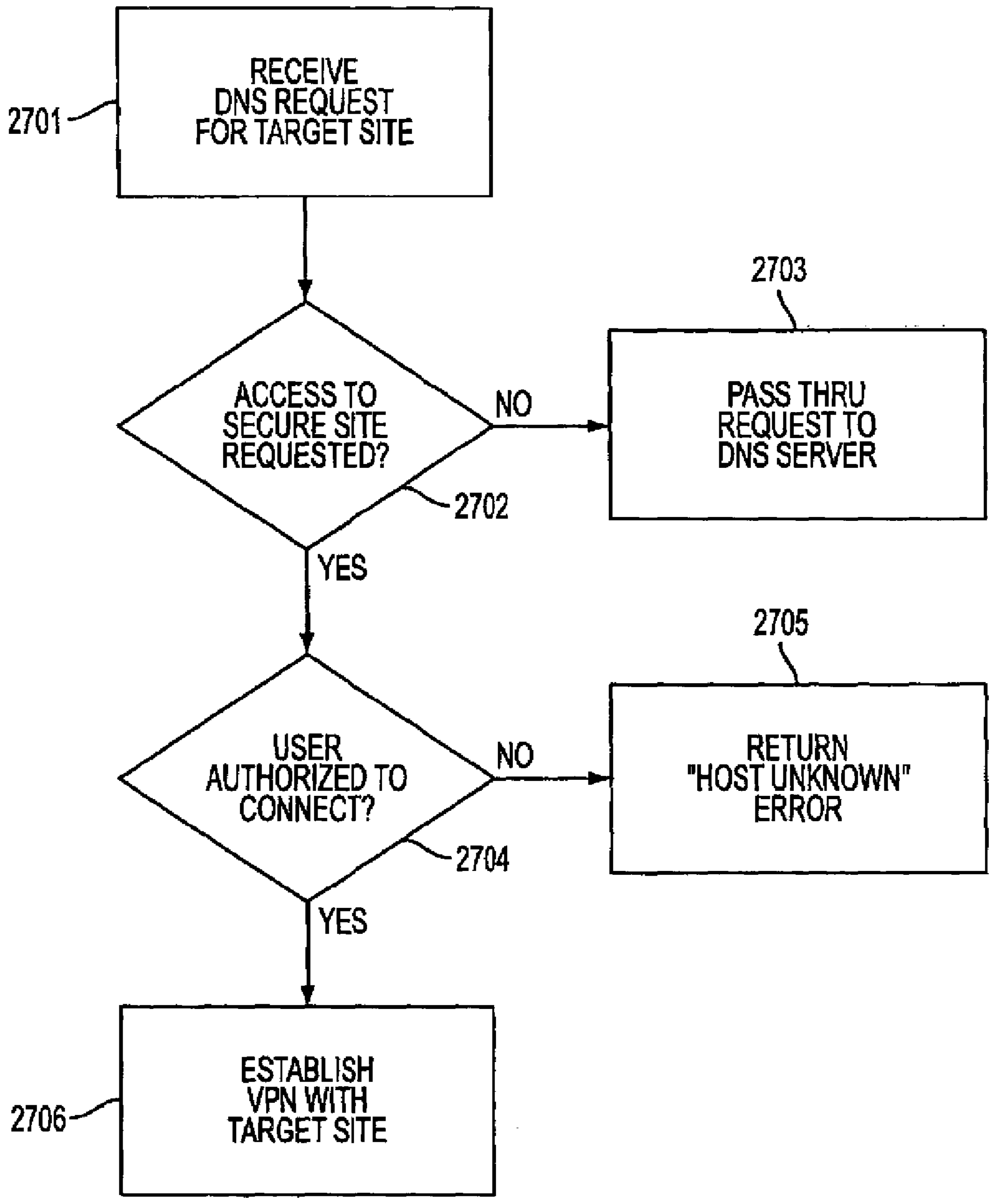

RECEIVE DNS REQUEST FOR TARGET SITE
2701
ACCESS TO SECURE SITE REQUESTED?
2702
NO
PASS THRU REQUEST TO DNS SERVER
2703
YES
USER AUTHORIZED TO CONNECT?
2704
NO
RETURN "HOST UNKNOWN" ERROR
2705
YES
ESTABLISH VPN WITH TARGET SITE
2706

FIG. 27

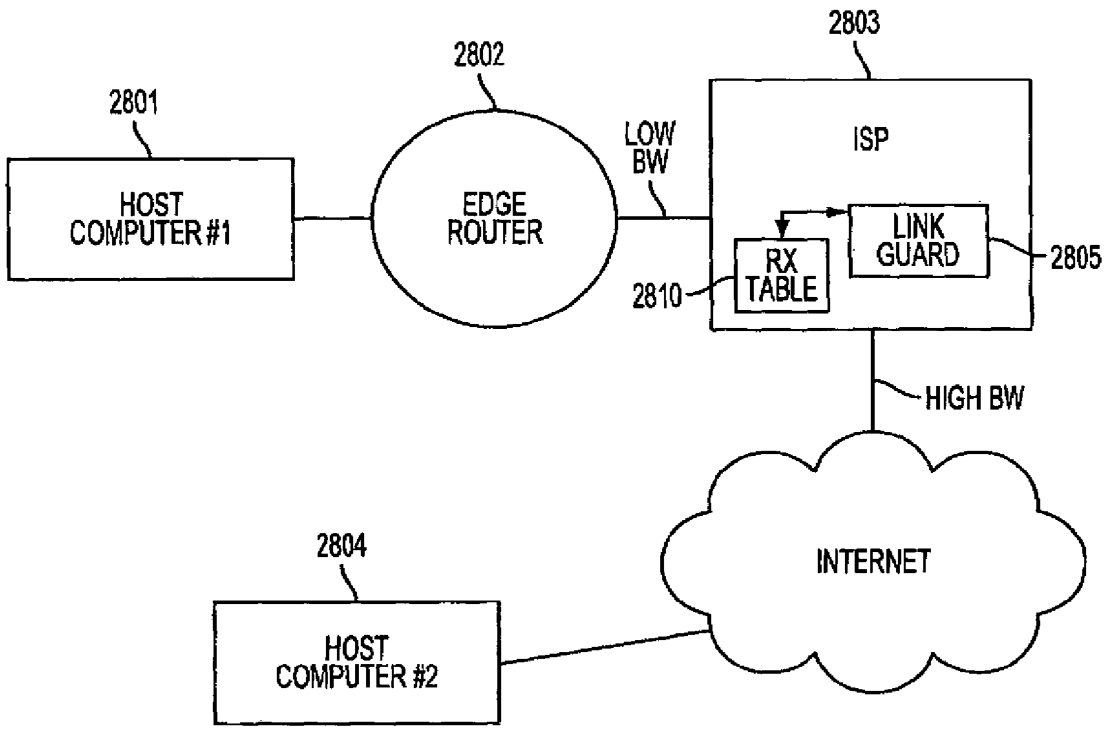

2801
HOST COMPUTER #1
2802
EDGE ROUTER
LOW BW
2803
ISP
LINK GUARD
2805
RX TABLE
2810
HIGH BW
INTERNET
2804
HOST COMPUTER #2

FIG. 28

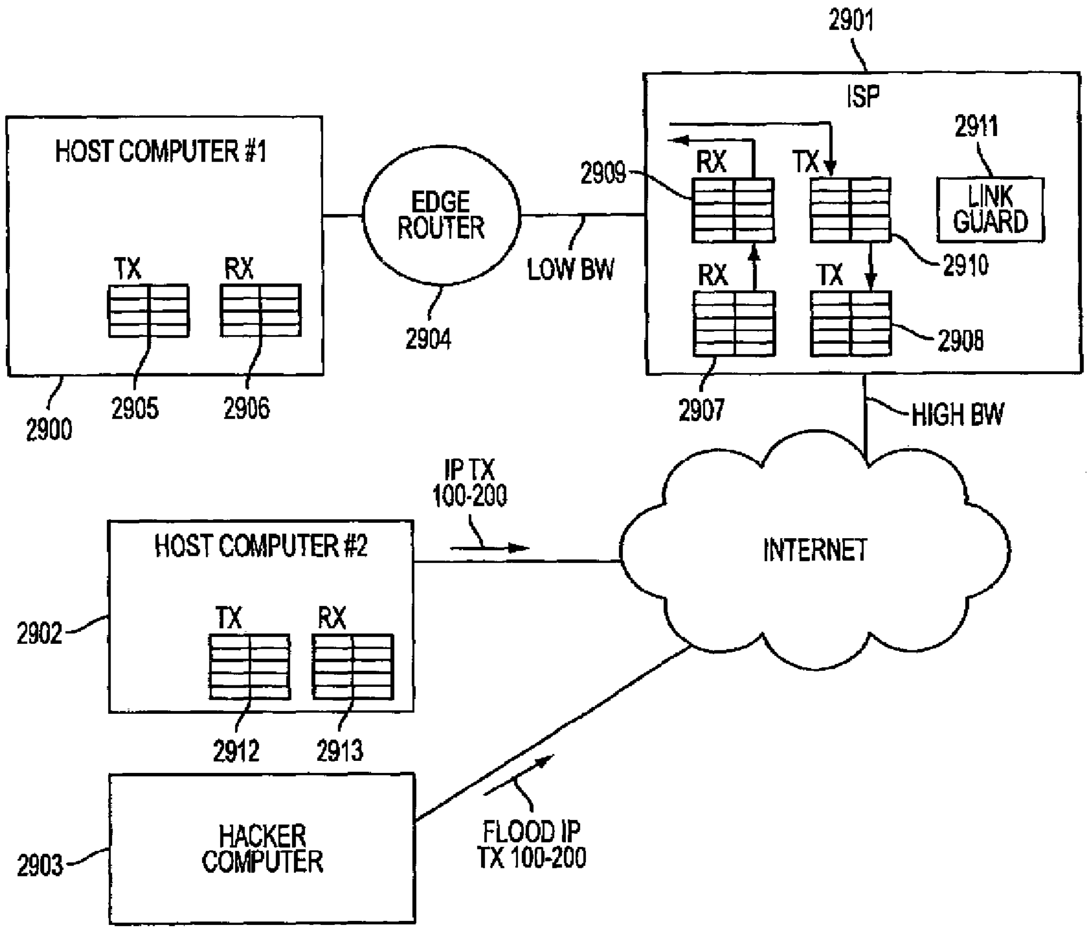

2901
ISP
2911
LINK GUARD
HOST COMPUTER #1
2909
RX
TX
EDGE ROUTER
LOW BW
2910
RX
TX
TX
RX
2908
2904
2900
2905
2906
2907
HIGH BW
IP TX 100-200
HOST COMPUTER #2
INTERNET
2902
TX
RX
2912
2913
HACKER COMPUTER
2903
FLOOD IP TX 100-200

FIG. 29

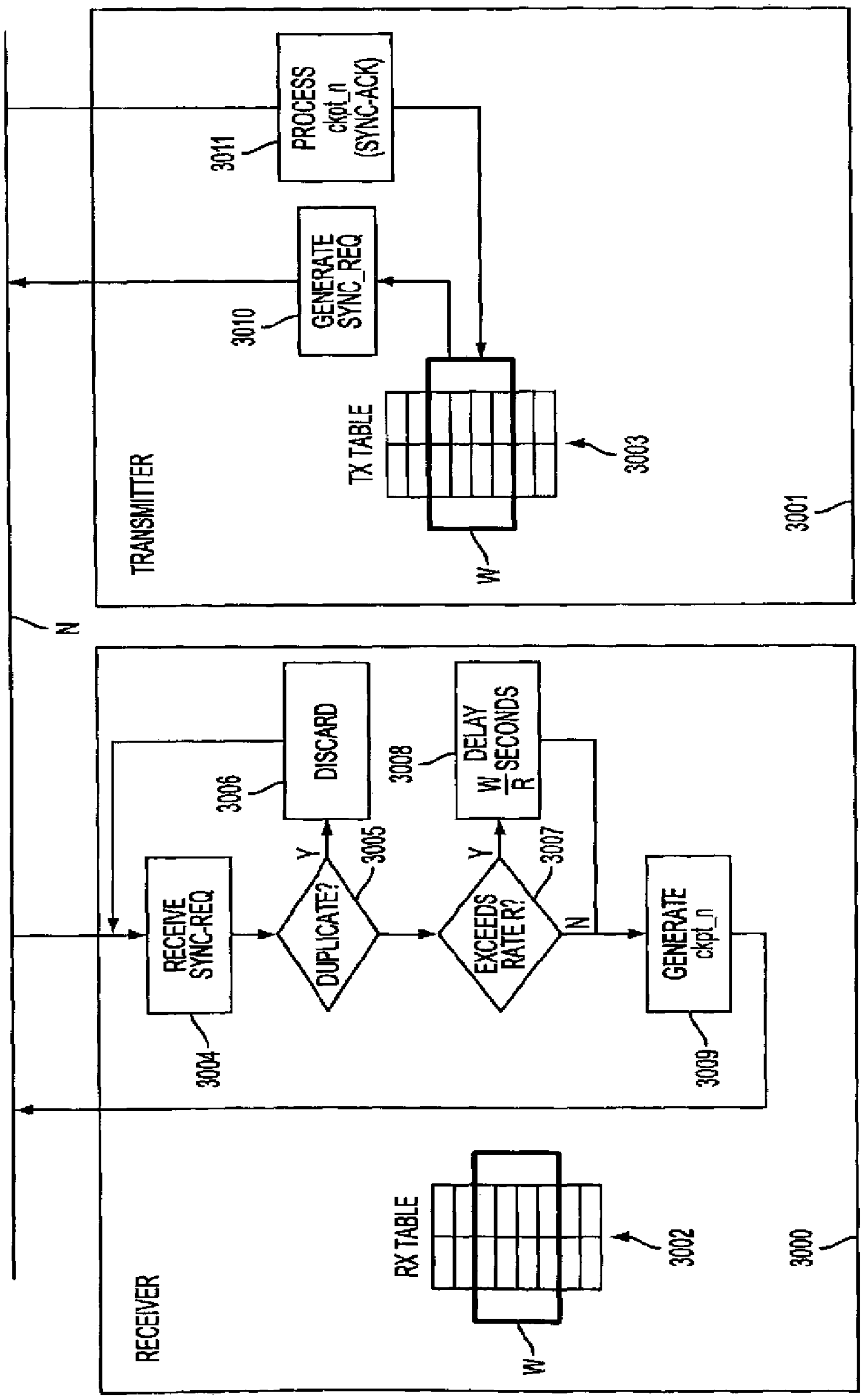

TRANSMITTER
PROCESS ckpt_n (SYNC-ACK)
3011
GENERATE SYNC_REQ
3010
TX TABLE
3003
W
3001
N
RECEIVER
RECEIVE SYNC-REQ
3004
DUPLICATE?
3005
Y
DISCARD
3006
EXCEEDS RATE R?
3007
Y
N
DELAY $\frac{W}{R}$ SECONDS
3008
GENERATE ckpt_n
3009
RX TABLE
3002
W
3000

FIG. 30

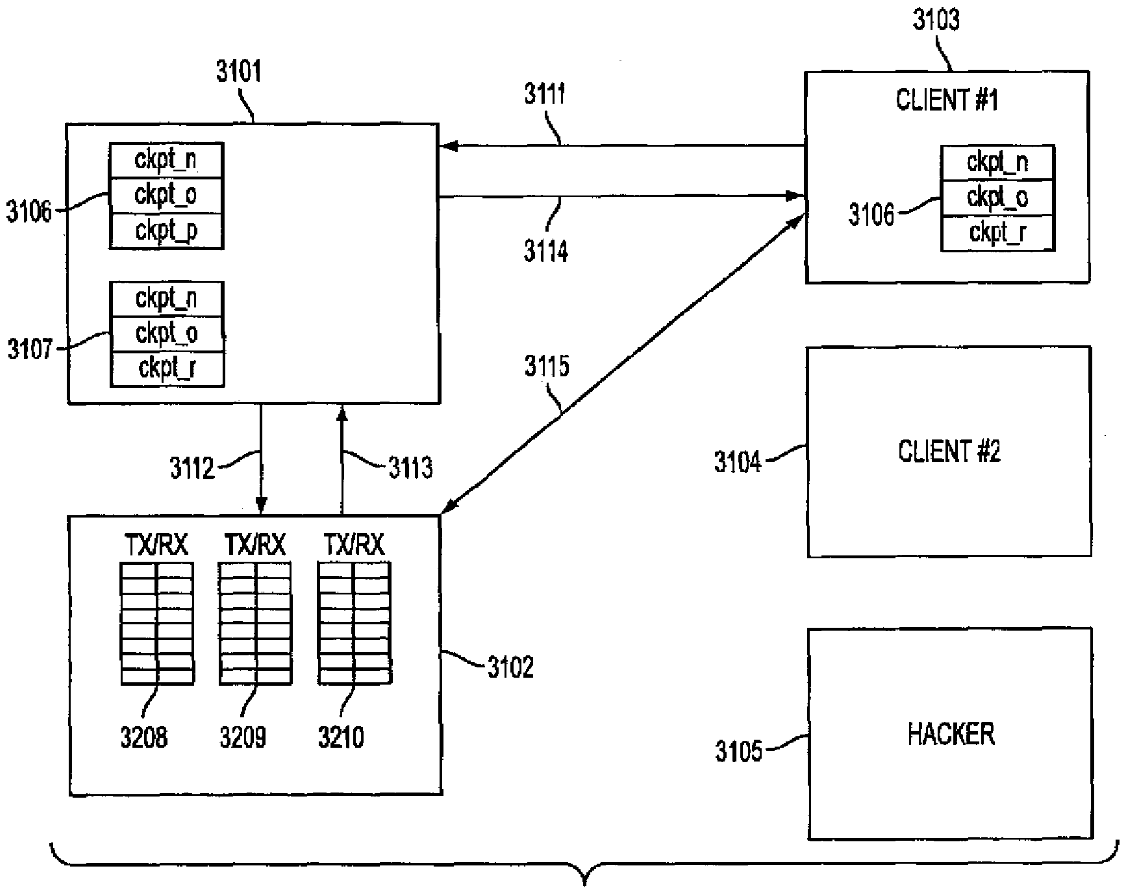

3101
3103
3111
3114
3115
CLIENT #1
ckpt_n
ckpt_o
ckpt_p
ckpt_r
3106
3107
3112
3113
TX/RX
3208
3209
3210
3102
CLIENT #2
3104
HACKER
3105

FIG. 31

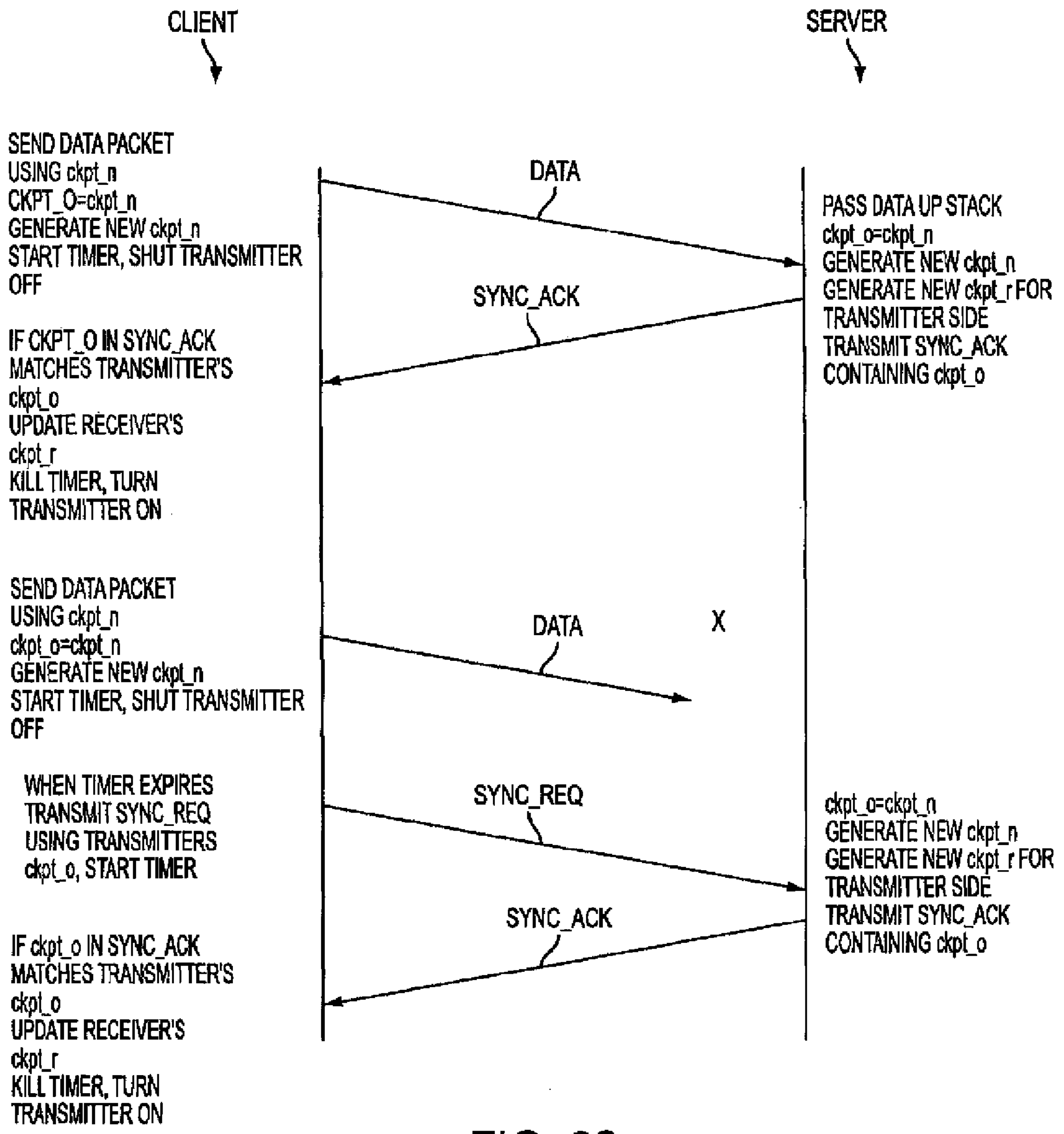

CLIENT
SERVER
SEND DATA PACKET
USING ckpt_n
CKPT_O=ckpt_n
GENERATE NEW ckpt_n
START TIMER, SHUT TRANSMITTER
OFF
DATA
PASS DATA UP STACK
ckpt_o=ckpt_n
GENERATE NEW ckpt_n
GENERATE NEW ckpt_r FOR
TRANSMITTER SIDE
TRANSMIT SYNC_ACK
CONTAINING ckpt_o
SYNC_ACK
IF CKPT_O IN SYNC_ACK
MATCHES TRANSMITTER'S
ckpt_o
UPDATE RECEIVER'S
ckpt_r
KILL TIMER, TURN
TRANSMITTER ON
SEND DATA PACKET
USING ckpt_n
ckpt_o=ckpt_n
GENERATE NEW ckpt_n
START TIMER, SHUT TRANSMITTER
OFF
DATA
X
WHEN TIMER EXPIRES
TRANSMIT SYNC_REQ
USING TRANSMITTERS
ckpt_o, START TIMER
SYNC_REQ
ckpt_o=ckpt_n
GENERATE NEW ckpt_n
GENERATE NEW ckpt_r FOR
TRANSMITTER SIDE
TRANSMIT SYNC_ACK
CONTAINING ckpt_o
SYNC_ACK
IF ckpt_o IN SYNC_ACK
MATCHES TRANSMITTER'S
ckpt_o
UPDATE RECEIVER'S
ckpt_r
KILL TIMER, TURN
TRANSMITTER ON

FIG. 32

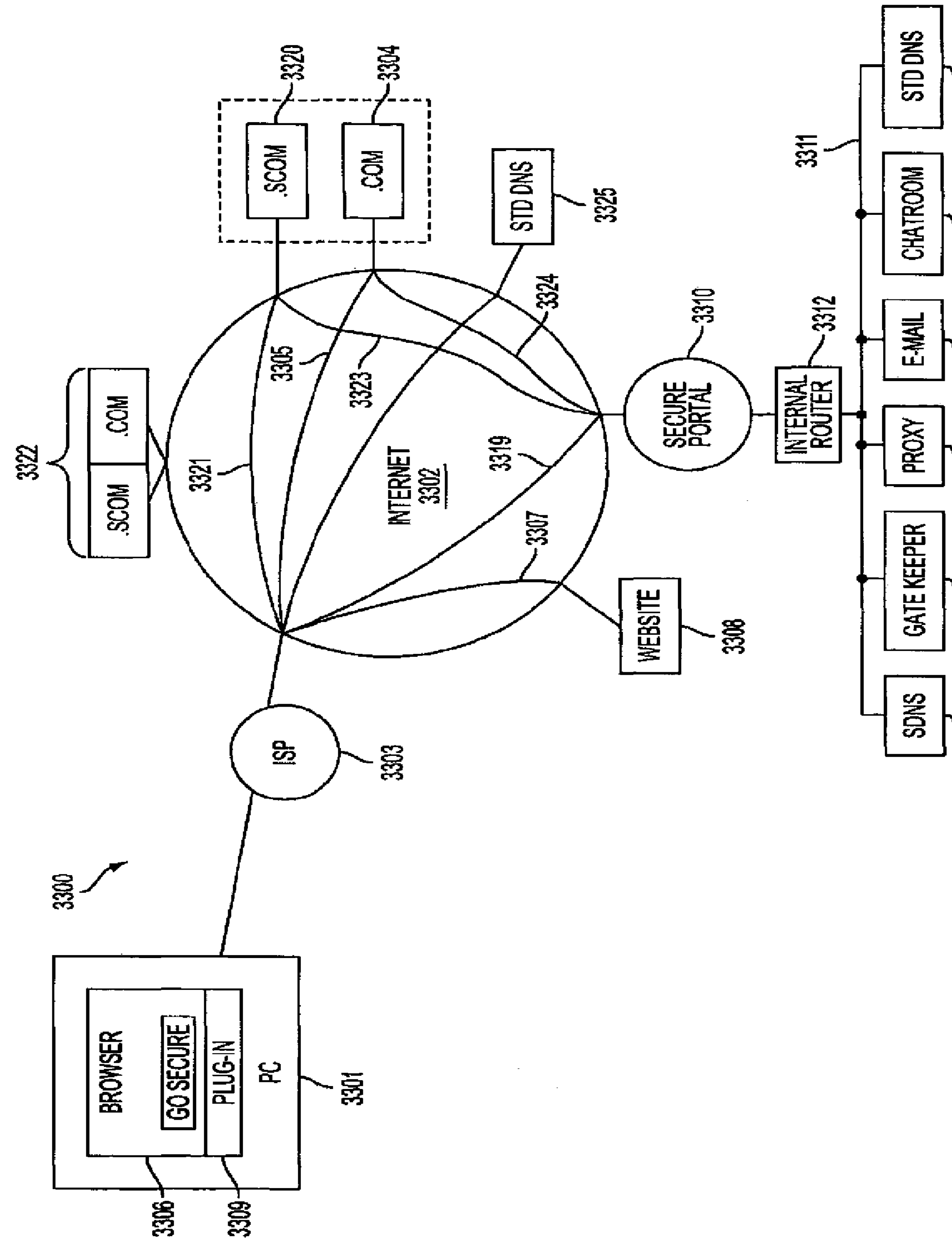

3300
BROWSER
GO SECURE
PLUG-IN
PC
3301
3306
3309
ISP
3303
3322
.SCOM
.COM
INTERNET
3302
3321
3305
3323
3319
3307
3324
3320
3304
.SCOM
.COM
STD DNS
3325
WEBSITE
3308
SECURE PORTAL
3310
INTERNAL ROUTER
3312
3311
SDNS
3313
GATE KEEPER
3314
PROXY
3315
E-MAIL
3316
CHATROOM
3317
STD DNS
3318

FIG. 33

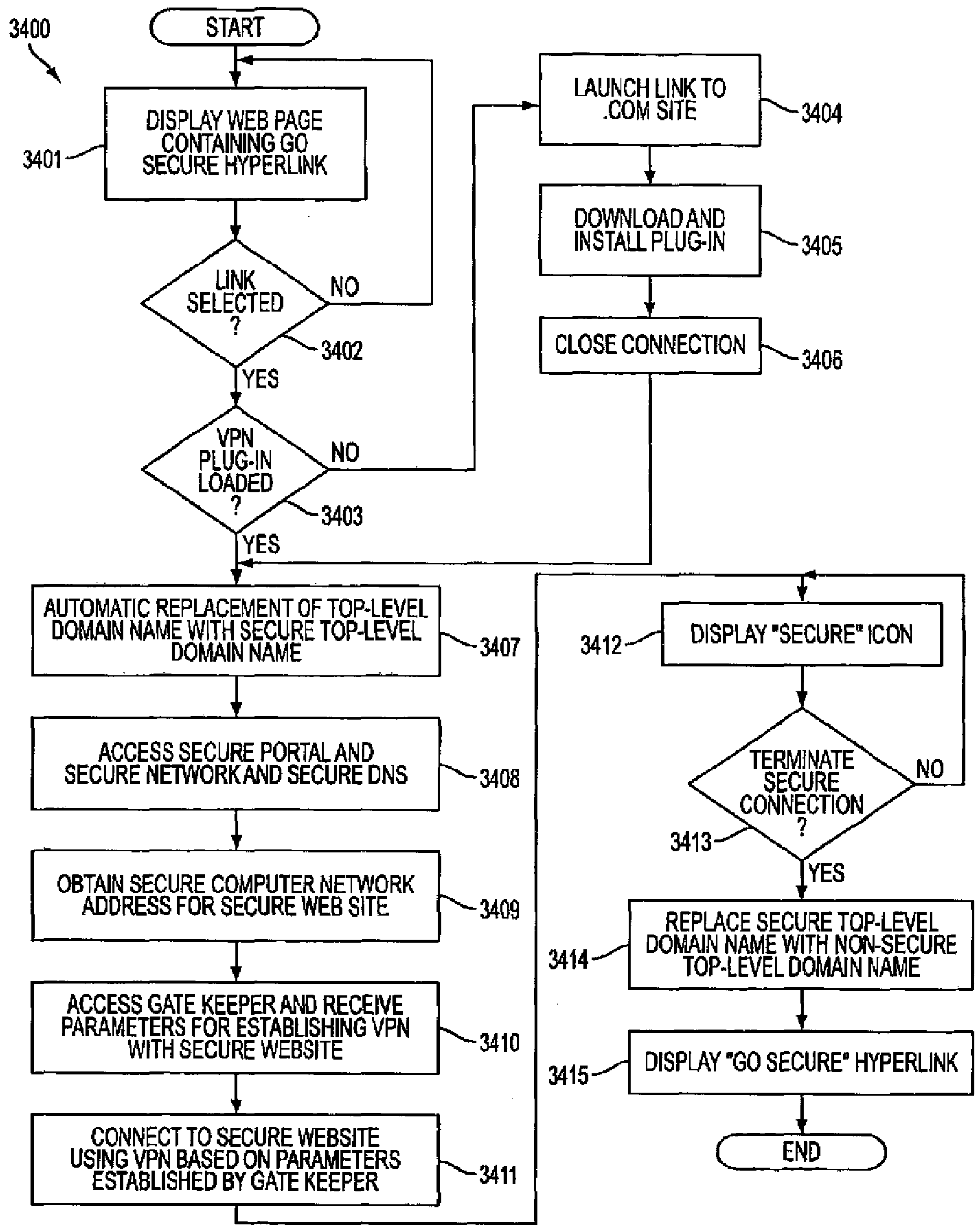

3400
START
DISPLAY WEB PAGE CONTAINING GO SECURE HYPERLINK
3401
LINK SELECTED ?
NO
3402
YES
VPN PLUG-IN LOADED ?
NO
3403
YES
LAUNCH LINK TO .COM SITE
3404
DOWNLOAD AND INSTALL PLUG-IN
3405
CLOSE CONNECTION
3406
AUTOMATIC REPLACEMENT OF TOP-LEVEL DOMAIN NAME WITH SECURE TOP-LEVEL DOMAIN NAME
3407
ACCESS SECURE PORTAL AND SECURE NETWORK AND SECURE DNS
3408
OBTAIN SECURE COMPUTER NETWORK ADDRESS FOR SECURE WEB SITE
3409
ACCESS GATE KEEPER AND RECEIVE PARAMETERS FOR ESTABLISHING VPN WITH SECURE WEBSITE
3410
CONNECT TO SECURE WEBSITE USING VPN BASED ON PARAMETERS ESTABLISHED BY GATE KEEPER
3411
3412
DISPLAY "SECURE" ICON
TERMINATE SECURE CONNECTION ?
NO
3413
YES
REPLACE SECURE TOP-LEVEL DOMAIN NAME WITH NON-SECURE TOP-LEVEL DOMAIN NAME
3414
DISPLAY "GO SECURE" HYPERLINK
3415
END

FIG. 34

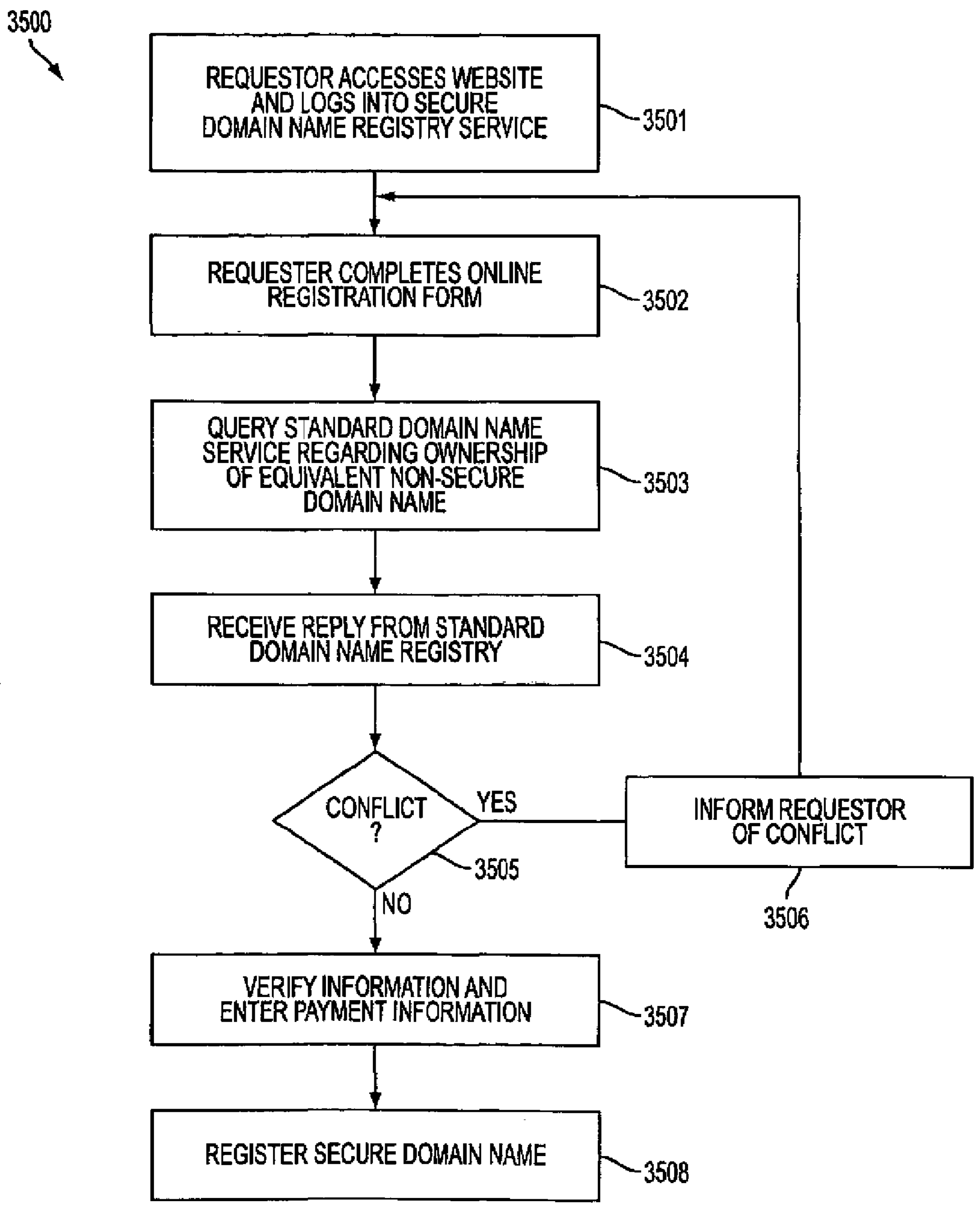

3500
REQUESTOR ACCESSES WEBSITE AND LOGS INTO SECURE DOMAIN NAME REGISTRY SERVICE
3501
REQUESTER COMPLETES ONLINE REGISTRATION FORM
3502
QUERY STANDARD DOMAIN NAME SERVICE REGARDING OWNERSHIP OF EQUIVALENT NON-SECURE DOMAIN NAME
3503
RECEIVE REPLY FROM STANDARD DOMAIN NAME REGISTRY
3504
CONFLICT ?
YES
3505
INFORM REQUESTOR OF CONFLICT
3506
NO
VERIFY INFORMATION AND ENTER PAYMENT INFORMATION
3507
REGISTER SECURE DOMAIN NAME
3508

FIG. 35

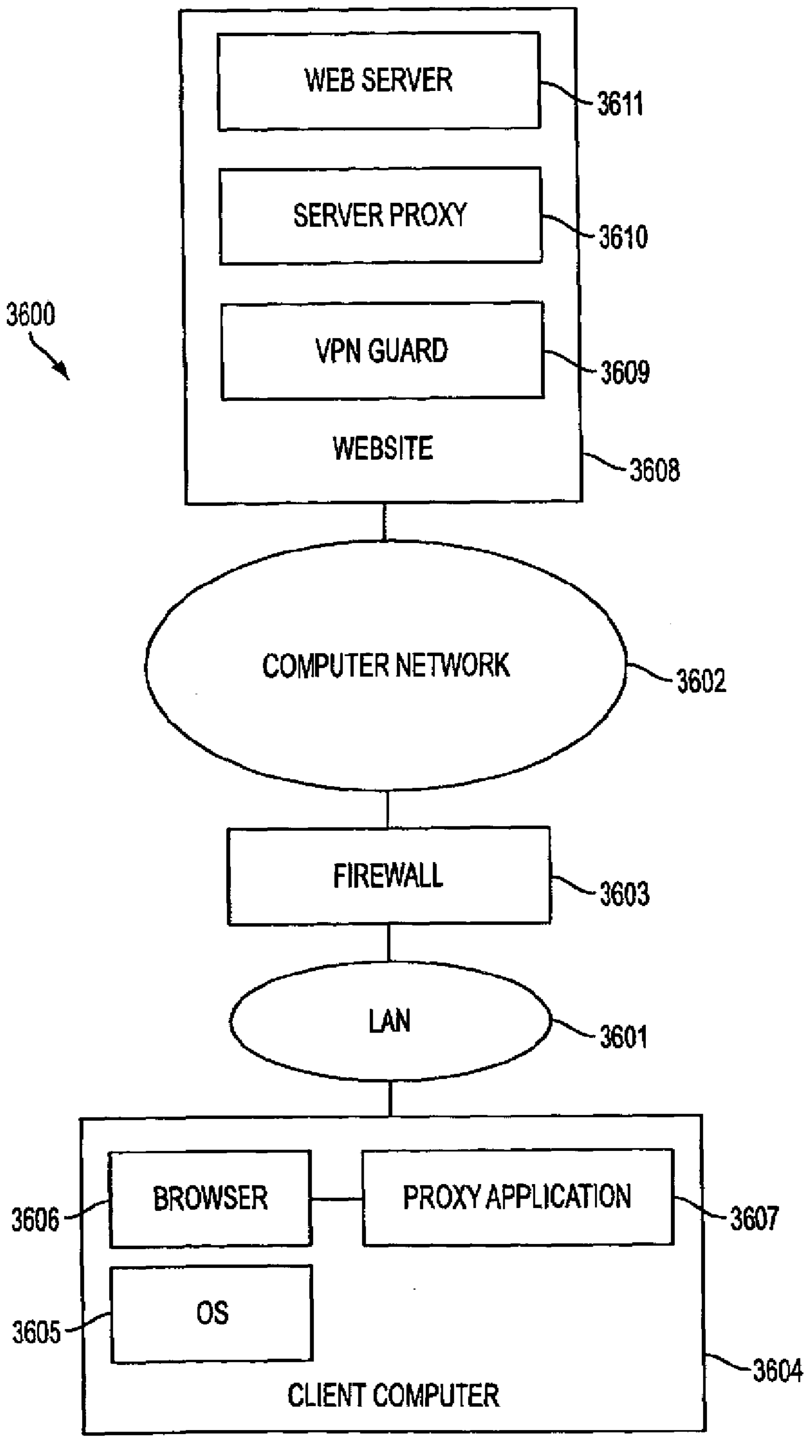

WEB SERVER
3611
SERVER PROXY
3610
3600
VPN GUARD
3609
WEBSITE
3608
COMPUTER NETWORK
3602
FIREWALL
3603
LAN
3601
BROWSER
3606
PROXY APPLICATION
3607
OS
3605
CLIENT COMPUTER
3604

FIG. 36

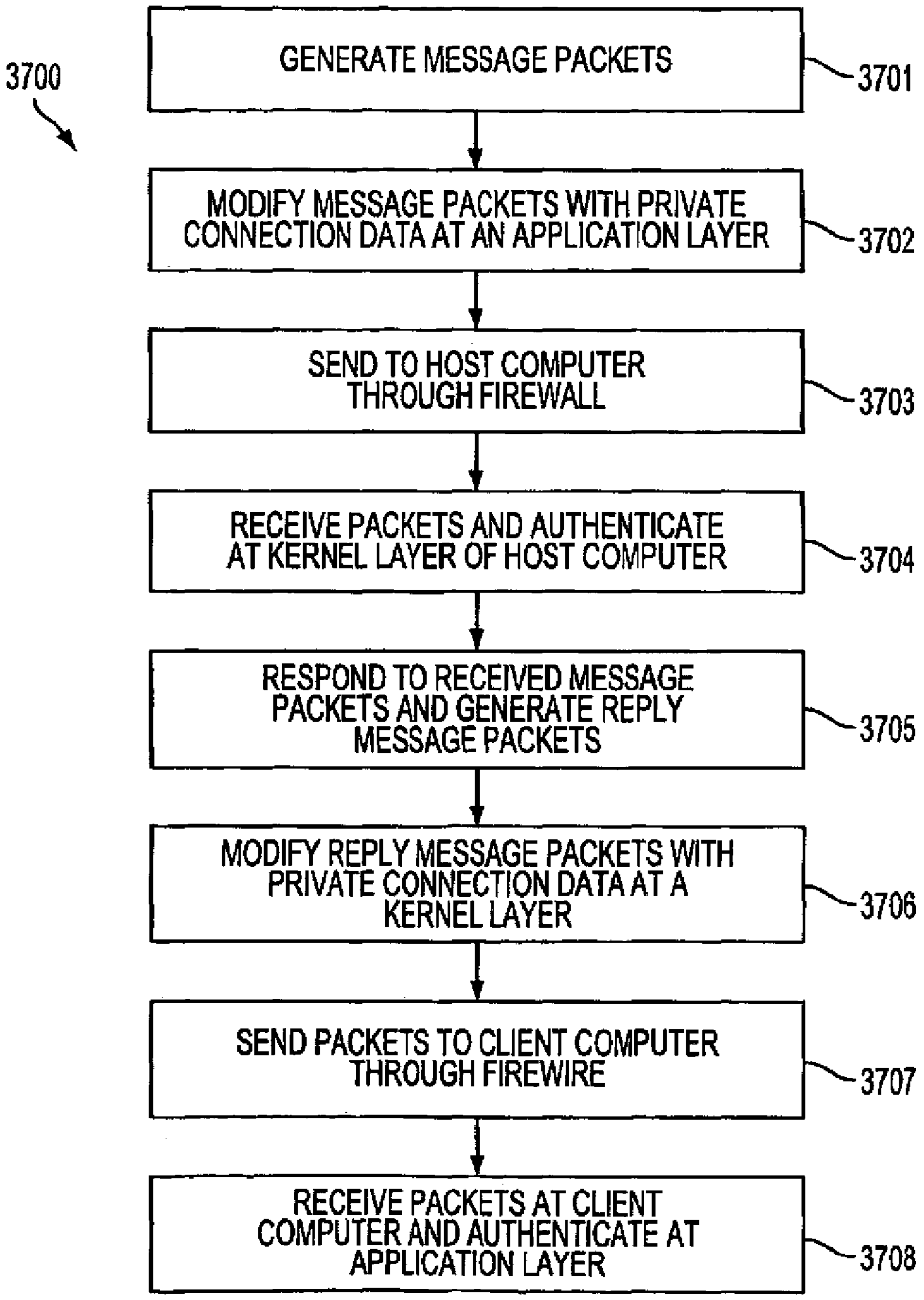

3700
GENERATE MESSAGE PACKETS
3701
MODIFY MESSAGE PACKETS WITH PRIVATE CONNECTION DATA AT AN APPLICATION LAYER
3702
SEND TO HOST COMPUTER THROUGH FIREWALL
3703
RECEIVE PACKETS AND AUTHENTICATE AT KERNEL LAYER OF HOST COMPUTER
3704
RESPOND TO RECEIVED MESSAGE PACKETS AND GENERATE REPLY MESSAGE PACKETS
3705
MODIFY REPLY MESSAGE PACKETS WITH PRIVATE CONNECTION DATA AT A KERNEL LAYER
3706
SEND PACKETS TO CLIENT COMPUTER THROUGH FIREWIRE
3707
RECEIVE PACKETS AT CLIENT COMPUTER AND AUTHENTICATE AT APPLICATION LAYER
3708

FIG. 37

AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority from and is a continuation patent application of U.S. application Ser. No. 09/558,210, filed Apr. 26, 2000 now abandoneed, which is a continuation-in-part patent application of previously-filed U.S. application Ser. No. 09/504,783, filed on Feb. 15, 2000, now U.S. Pat. No. 6,502,135, issued Dec. 31, 2002, which claims priority from and is a continuation-in-part patent application of previously-filed U.S. application Ser. No. 09/429,643, filed on Oct. 29, 1999 now U.S. Pat. No. 7,010,604. The subject matter of U.S. application Ser. No. 09/429,643, which is bodily incorporated herein, derives from provisional U.S. application Nos. 60/106,261 (filed Oct. 30, 1998) and 60/137,704 (filed Jun. 7, 1999). The present application is also related to U.S. application Ser. No. 09/558,209, filed Apr. 26, 2000, and which is incorporated by reference herein.

GOVERNMENT CONTRACT RIGHTS

This invention was made with Government support under Contract No. 360000-1999-000000-QC-000-000 awarded by the Central Intelligence Agency. The Government has certain rights in the invention.

BACKGROUND OF THE INVENTION

A tremendous variety of methods have been proposed and implemented to provide security and anonymity for communications over the Internet. The variety stems, in part, from the different needs of different Internet users. A basic heuristic framework to aid in discussing these different security techniques is illustrated in FIG. **1**. Two terminals, an originating terminal **100** and a destination terminal **110** are in communication over the Internet. It is desired for the communications to be secure, that is, immune to eavesdropping. For example, terminal **100** may transmit secret information to terminal **110** over the Internet **107**. Also, it may be desired to prevent an eavesdropper from discovering that terminal **100** is in communication with terminal **110**. For example, if terminal **100** is a user and terminal **110** hosts a web site, terminal **100**'s user may not want anyone in the intervening networks to know what web sites he is "visiting." Anonymity would thus be an issue, for example, for companies that want to keep their market research interests private and thus would prefer to prevent outsiders from knowing which web-sites or other Internet resources they are "visiting." These two security issues may be called data security and anonymity, respectively.

Data security is usually tackled using some form of data encryption. An encryption key **48** is known at both the originating and terminating terminals **100** and **110**. The keys may be private and public at the originating and destination terminals **100** and **110**, respectively or they may be symmetrical keys (the same key is used by both parties to encrypt and decrypt). Many encryption methods are known and usable in this context.

To hide traffic from a local administrator or ISP, a user can employ a local proxy server in communicating over an encrypted channel with an outside proxy such that the local administrator or ISP only sees the encrypted traffic. Proxy servers prevent destination servers from determining the

identities of the originating clients. This system employs an intermediate server interposed between client and destination server. The destination server sees only the Internet Protocol (IP) address of the proxy server and not the originating client. The target server only sees the address of the outside proxy. This scheme relies on a trusted outside proxy server. Also, proxy schemes are vulnerable to traffic analysis methods of determining identities of transmitters and receivers. Another important limitation of proxy servers is that the server knows the identities of both calling and called parties. In many instances, an originating terminal, such as terminal A, would prefer to keep its identity concealed from the proxy, for example, if the proxy server is provided by an Internet service provider (ISP).

To defeat traffic analysis, a scheme called Chaum's mixes employs a proxy server that transmits and receives fixed length messages, including dummy messages. Multiple originating terminals are connected through a mix (a server) to multiple target servers. It is difficult to tell which of the originating terminals are communicating to which of the connected target servers, and the dummy messages confuse eavesdroppers' efforts to detect communicating pairs by analyzing traffic. A drawback is that there is a risk that the mix server could be compromised. One way to deal with this risk is to spread the trust among multiple mixes. If one mix is compromised, the identities of the originating and target terminals may remain concealed. This strategy requires a number of alternative mixes so that the intermediate servers interposed between the originating and target terminals are not determinable except by compromising more than one mix. The strategy wraps the message with multiple layers of encrypted addresses. The first mix in a sequence can decrypt only the outer layer of the message to reveal the next destination mix in sequence. The second mix can decrypt the message to reveal the next mix and so on. The target server receives the message and, optionally, a multi-layer encrypted payload containing return information to send data back in the same fashion. The only way to defeat such a mix scheme is to collude among mixes. If the packets are all fixed-length and intermixed with dummy packets, there is no way to do any kind of traffic analysis.

Still another anonymity technique, called 'crowds,' protects the identity of the originating terminal from the intermediate proxies by providing that originating terminals belong to groups of proxies called crowds. The crowd proxies are interposed between originating and target terminals. Each proxy through which the message is sent is randomly chosen by an upstream proxy. Each intermediate proxy can send the message either to another randomly chosen proxy in the "crowd" or to the destination. Thus, even crowd members cannot determine if a preceding proxy is the originator of the message or if it was simply passed from another proxy.

ZKS (Zero-Knowledge Systems) Anonymous IP Protocol allows users to select up to any of five different pseudonyms, while desktop software encrypts outgoing traffic and wraps it in User Datagram Protocol (UDP) packets. The first server in a 2+-hop system gets the UDP packets, strips off one layer of encryption to add another, then sends the traffic to the next server, which strips off yet another layer of encryption and adds a new one. The user is permitted to control the number of hops. At the final server, traffic is decrypted with an untraceable IP address. The technique is called onion-routing. This method can be defeated using traffic analysis. For a simple example, bursts of packets from a user during low-duty periods can reveal the identities of sender and receiver.

Firewalls attempt to protect LANs from unauthorized access and hostile exploitation or damage to computers connected to the LAN. Firewalls provide a server through which all access to the LAN must pass. Firewalls are centralized systems that require administrative overhead to maintain. They can be compromised by virtual-machine applications ("applets"). They instill a false sense of security that leads to security breaches for example by users sending sensitive information to servers outside the firewall or encouraging use of modems to sidestep the firewall security. Firewalls are not useful for distributed systems such as business travelers, extranets, small teams, etc.

SUMMARY OF THE INVENTION

A secure mechanism for communicating over the internet, including a protocol referred to as the Tunneled Agile Routing Protocol (TARP), uses a unique two-layer encryption format and special TARP routers. TARP routers are similar in function to regular IP routers. Each TARP router has one or more IP addresses and uses normal IP protocol to send IP packet messages ("packets" or "datagrams"). The IP packets exchanged between TARP terminals via TARP routers are actually encrypted packets whose true destination address is concealed except to TARP routers and servers. The normal or "clear" or "outside" IP header attached to TARP IP packets contains only the address of a next hop router or destination server. That is, instead of indicating a final destination in the destination field of the IP header, the TARP packet's IP header always points to a next-hop in a series of TARP router hops, or to the final destination. This means there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet since the destination could always be next-hop TARP router as well as the final destination.

Each TARP packet's true destination is concealed behind a layer of encryption generated using a link key. The link key is the encryption key used for encrypted communication between the hops intervening between an originating TARP terminal and a destination TARP terminal. Each TARP router can remove the outer layer of encryption to reveal the destination router for each TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal by the sender/receiver IP numbers in the cleartext IP header.

Once the outer layer of encryption is removed, the TARP router determines the final destination. Each TARP packet **140** undergoes a minimum number of hops to help foil traffic analysis. The hops may be chosen at random or by a fixed value. As a result, each TARP packet may make random trips among a number of geographically disparate routers before reaching its destination. Each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined. This feature is called agile routing. The fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. The associated advantages have to do with the inner layer of encryption discussed below. Agile routing is combined with another feature that furthers this purpose; a feature that ensures that any message is broken into multiple packets.

The IP address of a TARP router can be changed, a feature called IP agility. Each TARP router, independently or under direction from another TARP terminal or router, can change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs.

The message payload is hidden behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the intervening TARP routers. The session key is used to decrypt the payloads of the TARP packets permitting the data stream to be reconstructed.

Communication may be made private using link and session keys, which in turn may be shared and used according to any desired method. For example, public/private keys or symmetric keys may be used.

To transmit a data stream, a TARP originating terminal constructs a series of TARP packets from a series of IP packets generated by a network (IP) layer process. (Note that the terms "network layer," "data link layer," "application layer," etc. used in this specification correspond to the Open Systems Interconnection (OSI) network terminology.) The payloads of these packets are assembled into a block and chain-block encrypted using the session key. This assumes, of course, that all the IP packets are destined for the same TARP terminal. The block is then interleaved and the interleaved encrypted block is broken into a series of payloads, one for each TARP packet to be generated. Special TARP headers IPT are then added to each payload using the IP headers from the data stream packets. The TARP headers can be identical to normal IP headers or customized in some way. They should contain a formula or data for deinterleaving the data at the destination TARP terminal, a time-to-live (TTL) parameter to indicate the number of hops still to be executed, a data type identifier which indicates whether the payload contains, for example, TCP or UDP data, the sender's TARP address, the destination TARP address, and an indicator as to whether the packet contains real or decoy data or a formula for filtering out decoy data if decoy data is spread in some way through the TARP payload data.

Note that although chain-block encryption is discussed here with reference to the session key, any encryption method may be used. Preferably, as in chain block encryption, a method should be used that makes unauthorized decryption difficult without an entire result of the encryption process. Thus, by separating the encrypted block among multiple packets and making it difficult for an interloper to obtain access to all of such packets, the contents of the communications are provided an extra layer of security.

Decoy or dummy data can be added to a stream to help foil traffic analysis by reducing the peak-to-average network load. It may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to portion, or entirety, of a message, and that portion or entirety then interleaved into a number of separate packets. Considering the agile IP routing of the packets, and the attendant difficulty of reconstructing an entire sequence of packets to form a single block-encrypted message element, decoy packets can significantly increase the difficulty of reconstructing an entire data stream.

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Because the encryption system described above is insertable between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the Network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicating that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. In addition, it may create a subprocess that maintains the original IP address and continues interacting with the attacker in some manner.

Decoy packets may be generated by each TARP terminal on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis.

In various other embodiments of the invention, a scalable version of the system may be constructed in which a plurality of IP addresses are preassigned to each pair of communicating nodes in the network. Each pair of nodes agrees upon an algorithm for "hopping" between IP addresses (both sending and receiving), such that an eavesdropper sees apparently continuously random IP address pairs (source and destination) for packets transmitted between the pair. Overlapping or

"reusable" IP addresses may be allocated to different users on the same subnet, since each node merely verifies that a particular packet includes a valid source/destination pair from the agreed-upon algorithm. Source/destination pairs are preferably not reused between any two nodes during any given end-to-end session, though limited IP block sizes or lengthy sessions might require it.

Further improvements described in this continuation-in-part application include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities

The present invention provides key technologies for implementing a secure virtual Internet by using a new agile network protocol that is built on top of the existing Internet protocol (IP). The secure virtual Internet works over the existing Internet infrastructure, and interfaces with client applications the same way as the existing Internet. The key technologies provided by the present invention that support the secure virtual Internet include a "one-click" and "no-click" technique to become part of the secure virtual Internet, a secure domain name service (SDNS) for the secure virtual Internet, and a new approach for interfacing specific client applications onto the secure virtual Internet. According to the invention, the secure domain name service interfaces with existing applications, in addition to providing a way to register and serve domain names and addresses.

According to one aspect of the present invention, a user can conveniently establish a VPN using a "one-click" or a "no-click" technique without being required to enter user identification information, a password and/or an encryption key for establishing a VPN. The advantages of the present invention are provided by a method for establishing a secure communication link between a first computer and a second computer over a computer network, such as the Internet. In one embodiment, a secure communication mode is enabled at a first computer without a user entering any cryptographic information for establishing the secure communication mode of communication, preferably by merely selecting an icon displayed on the first computer. Alternatively, the secure communication mode of communication can be enabled by entering a command into the first computer. Then, a secure communication link is established between the first computer and a second computer over a computer network based on the enabled secure communication mode of communication. According to the invention, it is determined whether a secure communication software module is stored on the first computer in response to the step of enabling the secure communication mode of communication. A predetermined computer network address is then accessed for loading the secure communication software module when the software module is not stored on the first computer. Subsequently, the proxy software module is stored in the first computer. The secure communication link is a virtual private network communication link over the computer network. Preferably, the virtual private network can be based on inserting into each data packet one or more data values that vary according to a pseudo-random sequence. Alternatively, the virtual private network can be based on a computer network address hopping regime that is used to pseudorandomly change computer network addresses or other data values in packets transmitted between the first computer and the second computer, such that the second computer compares the data values in each data packet transmitted between the first computer and the second computer to a moving window of valid values. Yet another alternative provides that the virtual private network can be based on a comparison between a discriminator field in each data packet to a table of valid discriminator fields maintained for the first computer.

According to another aspect of the invention, a command is entered to define a setup parameter associated with the secure communication link mode of communication. Consequently, the secure communication mode is automatically established when a communication link is established over the computer network.

The present invention also provides a computer system having a communication link to a computer network, and a display showing a hyperlink for establishing a virtual private network through the computer network. When the hyperlink for establishing the virtual private network is selected, a virtual private network is established over the computer network. A non-standard top-level domain name is then sent over the virtual private network communication to a predetermined computer network address, such as a computer network address for a secure domain name service (SDNS).

The present invention provides a domain name service that provides secure computer network addresses for secure, non-standard top-level domain names. The advantages of the present invention are provided by a secure domain name service for a computer network that includes a portal connected to a computer network, such as the Internet, and a domain name database connected to the computer network through the portal. According to the invention, the portal authenticates a query for a secure computer network address, and the domain name database stores secure computer network addresses for the computer network. Each secure computer network address is based on a non-standard top-level domain name, such as .scom, .sorg, .snet, .snet, .sedu, .smil and .sint.

The present invention provides a way to encapsulate existing application network traffic at the application layer of a client computer so that the client application can securely communicate with a server protected by an agile network protocol. The advantages of the present invention are provided by a method for communicating using a private communication link between a client computer and a server computer over a computer network, such as the Internet. According to the invention, an information packet is sent from the client computer to the server computer over the computer network. The information packet contains data that is inserted into the payload portion of the packet at the application layer of the client computer and is used for forming a virtual private connection between the client computer and the server computer. The modified information packet can be sent through a firewall before being sent over the computer network to the server computer and by working on top of existing protocols (i.e., UDP, ICMP and TCP), the present invention more easily penetrates the firewall. The information packet is received at a kernel layer of an operating system on the server side. It is then determined at the kernel layer of the operating system on the host computer whether the information packet contains the data that is used for forming the virtual private connection. The server side replies by sending an information packet to the client computer that has been modified at the kernel layer to containing virtual private connection information in the payload portion of the reply information packet. Preferably, the information packet from the client computer and the reply information packet from the server side are each a UDP protocol information packet. Alternative, both information packets could be a TCP/IP protocol information packet, or an ICMP protocol information packet.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is an illustration of secure communications over the Internet according to a prior art embodiment.

FIG. **2** is an illustration of secure communications over the Internet according to a an embodiment of the invention.

FIG. **3***a* is an illustration of a process of forming a tunneled IP packet according to an embodiment of the invention.

FIG. **3***b* is an illustration of a process of forming a tunneled IP packet according to another embodiment of the invention.

FIG. **4** is an illustration of an OSI layer location of processes that may be used to implement the invention.

FIG. **5** is a flow chart illustrating a process for routing a tunneled packet according to an embodiment of the invention.

FIG. **6** is a flow chart illustrating a process for forming a tunneled packet according to an embodiment of the invention.

FIG. **7** is a flow chart illustrating a process for receiving a tunneled packet according to an embodiment of the invention.

FIG. **8** shows how a secure session is established and synchronized between a client and a TARP router.

FIG. **9** shows an IP address hopping scheme between a client computer and TARP router using transmit and receive tables in each computer.

FIG. **10** shows physical link redundancy among three Internet Service Providers (ISPs) and a client computer.

FIG. **11** shows how multiple IP packets can be embedded into a single "frame" such as an Ethernet frame, and further shows the use of a discriminator field to camouflage true packet recipients.

FIG. **12A** shows a system that employs hopped hardware addresses, hopped IP addresses, and hopped discriminator fields.

FIG. **12B** shows several different approaches for hopping hardware addresses, IP addresses, and discriminator fields in combination.

FIG. **13** shows a technique for automatically re-establishing synchronization between sender and receiver through the use of a partially public sync value.

FIG. **14** shows a "checkpoint" scheme for regaining synchronization between a sender and recipient.

FIG. **15** shows further details of the checkpoint scheme of FIG. **14**.

FIG. **16** shows how two addresses can be decomposed into a plurality of segments for comparison with presence vectors.

FIG. **17** shows a storage array for a receiver's active addresses.

FIG. **18** shows the receiver's storage array after receiving a sync request.

FIG. **19** shows the receiver's storage array after new addresses have been generated.

FIG. **20** shows a system employing distributed transmission paths.

FIG. **21** shows a plurality of link transmission tables that can be used to route packets in the system of FIG. **20**.

FIG. **22A** shows a flowchart for adjusting weight value distributions associated with a plurality of transmission links.

FIG. **22B** shows a flowchart for setting a weight value to zero if a transmitter turns off.

FIG. 23 shows a system employing distributed transmission paths with adjusted weight value distributions for each path.

FIG. 24 shows an example using the system of FIG. 23.

FIG. 25 shows a conventional domain-name look-up service.

FIG. 26 shows a system employing a DNS proxy server with transparent VPN creation.

FIG. 27 shows steps that can be carried out to implement transparent VPN creation based on a DNS look-up function.

FIG. 28 shows a system including a link guard function that prevents packet overloading on a low-bandwidth link LOW BW.

FIG. 29 shows one embodiment of a system employing the principles of FIG. 28.

FIG. 30 shows a system that regulates packet transmission rates by throttling the rate at which synchronizations are performed.

FIG. 31 shows a signaling server 3101 and a transport server 3102 used to establish a VPN with a client computer.

FIG. 32 shows message flows relating to synchronization protocols of FIG. 31.

FIG. 33 shows a system block diagram of a computer network in which the "one-click" secure communication link of the present invention is suitable for use.

FIG. 34 shows a flow diagram for installing and establishing a "one-click" secure communication link over a computer network according to the present invention.

FIG. 35 shows a flow diagram for registering a secure domain name according to the present invention.

FIG. 36 shows a system block diagram of a computer network in which a private connection according to the present invention can be configured to more easily traverse a firewall between two computer networks.

FIG. 37 shows a flow diagram for establishing a virtual private connection that is encapsulated using an existing network protocol.

DETAILED DESCRIPTION OF THE INVENTION

Referring to FIG. 2, a secure mechanism for communicating over the internet employs a number of special routers or servers, called TARP routers 122-127 that are similar to regular IP routers 128-132 in that each has one or more IP addresses and uses normal IP protocol to send normal-looking IP packet messages, called TARP packets 140. TARP packets 140 are identical to normal IP packet messages that are routed by regular IP routers 128-132 because each TARP packet 140 contains a destination address as in a normal IP packet. However, instead of indicating a final destination in the destination field of the IP header, the TARP packet's 140 IP header always points to a next-hop in a series of TARP router hops, or the final destination, TARP terminal 110. Because the header of the TARP packet contains only the next-hop destination, there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet 140 since the destination could always be the next-hop TARP router as well as the final destination, TARP terminal 110.

Each TARP packet's true destination is concealed behind an outer layer of encryption generated using a link key 146. The link key 146 is the encryption key used for encrypted communication between the end points (TARP terminals or TARP routers) of a single link in the chain of hops connecting the originating TARP terminal 100 and the destination TARP terminal 110. Each TARP router 122-127, using the link key 146 it uses to communicate with the previous hop in a chain, can use the link key to reveal the true destination of a TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal (which may indicate the link key used) by the sender field of the clear IP header. Alternatively, this identity may be hidden behind another layer of encryption in available bits in the clear IP header. Each TARP router, upon receiving a TARP message, determines if the message is a TARP message by using authentication data in the TARP packet. This could be recorded in available bytes in the TARP packet's IP header. Alternatively, TARP packets could be authenticated by attempting to decrypt using the link key 146 and determining if the results are as expected. The former may have computational advantages because it does not involve a decryption process.

Once the outer layer of decryption is completed by a TARP router 122-127, the TARP router determines the final destination. The system is preferably designed to cause each TARP packet 140 to undergo a minimum number of hops to help foil traffic analysis. The time to live counter in the IP header of the TARP message may be used to indicate a number of TARP router hops yet to be completed. Each TARP router then would decrement the counter and determine from that whether it should forward the TARP packet 140 to another TARP router 122-127 or to the destination TARP terminal 110. If the time to live counter is zero or below zero after decrementing, for an example of usage, the TARP router receiving the TARP packet 140 may forward the TARP packet 140 to the destination TARP terminal 110. If the time to live counter is above zero after decrementing, for an example of usage, the TARP router receiving the TARP packet 140 may forward the TARP packet 140 to a TARP router 122-127 that the current TARP terminal chooses at random. As a result, each TARP packet 140 is routed through some minimum number of hops of TARP routers 122-127 which are chosen at random.

Thus, each TARP packet, irrespective of the traditional factors determining traffic in the Internet, makes random trips among a number of geographically disparate routers before reaching its destination and each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined as described above. This feature is called agile routing. For reasons that will become clear shortly, the fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. Agile routing is combined with another feature that furthers this purpose, a feature that ensures that any message is broken into multiple packets.

A TARP router receives a TARP packet when an IP address used by the TARP router coincides with the IP address in the TARP packet's IP header IP_C. The IP address of a TARP router, however, may not remain constant. To avoid and manage attacks, each TARP router, independently or under direction from another TARP terminal or router, may change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUT's. In reality, whenever a TARP router looks up the address of a destination in the encrypted header, it must convert a TARP address to a real IP address using its LUT.

While every TARP router receiving a TARP packet has the ability to determine the packet's final destination, the message payload is embedded behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the TARP routers **122-127** intervening between the originating **100** and destination **110** TARP terminals. The session key is used to decrypt the payloads of the TARP packets **140** permitting an entire message to be reconstructed.

In one embodiment, communication may be made private using link and session keys, which in turn may be shared and used according any desired method. For example, a public key or symmetric keys may be communicated between link or session endpoints using a public key method. Any of a variety of other mechanisms for securing data to ensure that only authorized computers can have access to the private information in the TARP packets **140** may be used as desired.

Referring to FIG. **3***a*, to construct a series of TARP packets, a data stream **300** of IP packets **207***a*, **207***b*, **207***c*, etc., such series of packets being formed by a network (IP) layer process, is broken into a series of small sized segments. In the present example, equal-sized segments **1-9** are defined and used to construct a set of interleaved data packets A, B, and C. Here it is assumed that the number of interleaved packets A, B, and C formed is three and that the number of IP packets **207***a*-**207***c* used to form the three interleaved packets A, B, and C is exactly three. Of course, the number of IP packets spread over a group of interleaved packets may be any convenient number as may be the number of interleaved packets over which the incoming data stream is spread. The latter, the number of interleaved packets over which the data stream is spread, is called the interleave window.

To create a packet, the transmitting software interleaves the normal IP packets **207***a* et. seq. to form a new set of interleaved payload data **320**. This payload data **320** is then encrypted using a session key to form a set of session-key-encrypted payload data **330**, each of which, A, B, and C, will form the payload of a TARP packet. Using the IP header data, from the original packets **207***a*-**207***c*, new TARP headers IP_T are formed. The TARP headers IP_T can be identical to normal IP headers or customized in some way. In a preferred embodiment, the TARP headers IP_T are IP headers with added data providing the following information required for routing and reconstruction of messages, some of which data is ordinarily, or capable of being, contained in normal IP headers:

1. A window sequence number—an identifier that indicates where the packet belongs in the original message sequence.
2. An interleave sequence number—an identifier that indicates the interleaving sequence used to form the packet so that the packet can be deinterleaved along with other packets in the interleave window.
3. A time-to-live (TTL) datum—indicates the number of TARP-router-hops to be executed before the packet reaches its destination. Note that the TTL parameter may provide a datum to be used in a probabilistic formula for determining whether to route the packet to the destination or to another hop.
4. Data type identifier—indicates whether the payload contains, for example, TCP or UDP data.
5. Sender's address—indicates the sender's address in the TARP network.
6. Destination address—indicates the destination terminal's address in the TARP network.
7. Decoy/Real—an indicator of whether the packet contains real message data or dummy decoy data or a combination.

Obviously, the packets going into a single interleave window must include only packets with a common destination. Thus, it is assumed in the depicted example that the IP headers of IP packets **207***a*-**207***c* all contain the same destination address or at least will be received by the same terminal so that they can be deinterleaved. Note that dummy or decoy data or packets can be added to form a larger interleave window than would otherwise be required by the size of a given message. Decoy or dummy data can be added to a stream to help foil traffic analysis by leveling the load on the network. Thus, it may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to a portion, or the entirety, of a message, and that portion or entirety then interleaved into a number of separate packets.

Referring to FIG. **3***b*, in an alternative mode of TARP packet construction, a series of IP packets are accumulated to make up a predefined interleave window. The payloads of the packets are used to construct a single block **520** for chain block encryption using the session key. The payloads used to form the block are presumed to be destined for the same terminal. The block size may coincide with the interleave window as depicted in the example embodiment of FIG. **3***b*. After encryption, the encrypted block is broken into separate payloads and segments which are interleaved as in the embodiment of FIG. **3***a*. The resulting interleaved packets A, B, and C, are then packaged as TARP packets with TARP headers as in the Example of FIG. **3***a*. The remaining process is as shown in, and discussed with reference to, FIG. **3***a*.

Once the TARP packets **340** are formed, each entire TARP packet **340**, including the TARP header IP_T, is encrypted using the link key for communication with the first-hop-TARP router. The first hop TARP router is randomly chosen. A final unencrypted IP header IP_C is added to each encrypted TARP packet **340** to form a normal IP packet **360** that can be transmitted to a TARP router. Note that the process of constructing the TARP packet **360** does not have to be done in stages as described. The above description is just a useful heuristic for describing the final product, namely, the TARP packet.

Note that, TARP header IP_T could be a completely custom header configuration with no similarity to a normal IP header except that it contain the information identified above. This is so since this header is interpreted by only TARP routers.

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Referring to FIG. **4**, a TARP transceiver **405** can be an originating terminal **100**, a destination terminal **110**, or a TARP router **122-127**. In each TARP Transceiver **405**, a transmitting process is generated to receive normal packets from the Network (IP) layer and generate TARP packets for communication over the network. A receiving process is generated to receive normal IP packets containing TARP packets and generate from these normal IP packets which are "passed up" to the Network (IP) layer. Note that where the TARP Transceiver **405** is a router, the received TARP packets **140** are not processed into a stream of IP packets **415** because they need only be authenticated as proper TARP packets and then passed to another TARP router or a TARP destination terminal **110**. The intervening process, a "TARP Layer" **420**, could be combined with either the data link layer **430** or the Network layer **410**. In either case, it would intervene between the data link layer **430** so that the process would receive regular IP packets containing embedded TARP packets and "hand up" a series of reassembled IP packets to the Network layer **410**. As an example of combining the TARP layer **420** with the data link layer **430**, a program may augment the normal processes running a communications card, for example, an Ethernet card. Alternatively, the TARP layer processes may form part of a dynamically loadable module that is loaded and executed to support communications between the network and data link layers.

Because the encryption system described above can be inserted between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

Note that IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicates that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. To accomplish this, the TARP process will construct a TARP-formatted message, in the style of Internet Control Message Protocol (ICMP) datagrams as an example; this message will contain the machine's TARP address, its previous IP address, and its new IP address. The TARP layer will transmit this packet to at least one known TARP router; then upon receipt and validation of the message, the TARP router will update its LUT with the new IP address for the stated TARP address. The TARP router will then format a similar message, and broadcast it to the other TARP routers so that they may update their LUTs. Since the total number of TARP routers on any given subnet is expected to be relatively small, this process of updating the LUTs should be relatively fast. It may not, however, work as well when there is a relatively large number of TARP routers and/or a relatively large number of clients; this has motivated a refinement of this architecture to provide scalability; this refinement has led to a second embodiment, which is discussed below.

Upon detection of an attack, the TARP process may also create a subprocess that maintains the original IP address and continues interacting with the attacker. The latter may provide an opportunity to trace the attacker or study the attacker's methods (called "fishbowling" drawing upon the analogy of a small fish in a fish bowl that "thinks" it is in the ocean but is actually under captive observation). A history of the communication between the attacker and the abandoned (fishbowled) IP address can be recorded or transmitted for human analysis or further synthesized for purposes of responding in some way.

As mentioned above, decoy or dummy data or packets can be added to outgoing data streams by TARP terminals or routers. In addition to making it convenient to spread data over a larger number of separate packets, such decoy packets can also help to level the load on inactive portions of the Internet to help foil traffic analysis efforts.

Decoy packets may be generated by each TARP terminal **100**, **110** or each router **122-127** on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. That is, when a series of messages are received, the decoy packet generation rate may be increased. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis. The rate of reception of packets, decoy or otherwise, may be indicated to the decoy packet dropping and generating processes through perishable decoy and regular packet counters. (A perishable counter is one that resets or decrements its value in response to time so that it contains a high value when it is incremented in rapid succession and a small value when incremented either slowly or a small number of times in rapid succession.) Note that destination TARP terminal **110** may generate decoy packets equal in number and size to those TARP packets received to make it appear it is merely routing packets and is therefore not the destination terminal.

Referring to FIG. **5**, the following particular steps may be employed in the above-described method for routing TARP packets.

S0. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.

S2. The TARP packet may be probed in some way to authenticate the packet before attempting to decrypt it using the link key. That is, the router may determine that the packet is an authentic TARP packet by performing a selected operation on some data included with the clear IP header attached to the encrypted TARP packet contained in the payload. This makes it possible to avoid performing decryption on packets that are not authentic TARP packets.

S3. The TARP packet is decrypted to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.

S4. If the packet is a decoy packet, the perishable decoy counter is incremented.

S5. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the router may choose to throw it away. If the received packet is a decoy packet and it is determined that it should be thrown away (S6), control returns to step S0.

S7. The TTL parameter of the TARP header is decremented and it is determined if the TTL parameter is greater than zero.

S8. If the TTL parameter is greater than zero, a TARP address is randomly chosen from a list of TARP addresses maintained by the router and the link key and IP address corresponding to that TARP address memorized for use in creating a new IP packet containing the TARP packet.

S9. If the TTL parameter is zero or less, the link key and IP address corresponding to the TARP address of the destination are memorized for use in creating the new IP packet containing the TARP packet.

S10. The TARP packet is encrypted using the memorized link key.

S11. An IP header is added to the packet that contains the stored IP address, the encrypted TARP packet wrapped with an IP header, and the completed packet transmitted to the next hop or destination.

Referring to FIG. 6, the following particular steps may be employed in the above-described method for generating TARP packets.

S20. A background loop operation applies an algorithm that determines the generation of decoy IP packets. The loop is interrupted when a data stream containing IP packets is received for transmission.

S21. The received IP packets are grouped into a set consisting of messages with a constant IP destination address. The set is further broken down to coincide with a maximum size of an interleave window The set is encrypted, and interleaved into a set of payloads destined to become TARP packets.

S22. The TARP address corresponding to the IP address is determined from a lookup table and stored to generate the TARP header. An initial TTL count is generated and stored in the header. The TTL count may be random with minimum and maximum values or it may be fixed or determined by some other parameter.

S23. The window sequence numbers and interleave sequence numbers are recorded in the TARP headers of each packet.

S24. One TARP router address is randomly chosen for each TARP packet and the IP address corresponding to it stored for use in the clear IP header. The link key corresponding to this router is identified and used to encrypt TARP packets containing interleaved and encrypted data and TARP headers.

S25. A clear IP header with the first hop router's real IP address is generated and added to each of the encrypted TARP packets and the resulting packets.

Referring to FIG. 7, the following particular steps may be employed in the above-described method for receiving TARP packets.

S40. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.

S42. The TARP packet may be probed to authenticate the packet before attempting to decrypt it using the link key.

S43. The TARP packet is decrypted with the appropriate link key to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.

S44. If the packet is a decoy packet, the perishable decoy counter is incremented.

S45. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the receiver may choose to throw it away.

S46. The TARP packets are cached until all packets forming an interleave window are received.

S47. Once all packets of an interleave window are received, the packets are deinterleaved.

S48. The packets block of combined packets defining the interleave window is then decrypted using the session key.

S49. The decrypted block is then divided using the window sequence data and the IPT headers are converted into normal IPc headers. The window sequence numbers are integrated in the IPC headers.

S50. The packets are then handed up to the IP layer processes.

1. Scalability Enhancements

The IP agility feature described above relies on the ability to transmit IP address changes to all TARP routers. The embodiments including this feature will be referred to as "boutique" embodiments due to potential limitations in scaling these features up for a large network, such as the Internet. (The "boutique" embodiments would, however, be robust for use in smaller networks, such as small virtual private networks, for example). One problem with the boutique embodiments is that if IP address changes are to occur frequently, the message traffic required to update all routers sufficiently quickly creates a serious burden on the Internet when the TARP router and/or client population gets large. The bandwidth burden added to the networks, for example in ICMP packets, that would be used to update all the TARP routers could overwhelm the Internet for a large scale implementation that approached the scale of the Internet. In other words, the boutique system's scalability is limited.

A system can be constructed which trades some of the features of the above embodiments to provide the benefits of IP agility without the additional messaging burden. This is accomplished by IP address-hopping according to shared algorithms that govern IP addresses used between links participating in communications sessions between nodes such as TARP nodes. (Note that the IP hopping technique is also applicable to the boutique embodiment.) The IP agility feature discussed with respect to the boutique system can be modified so that it becomes decentralized under this scalable regime and governed by the above-described shared algorithm. Other features of the boutique system may be combined with this new type of IP-agility.

The new embodiment has the advantage of providing IP agility governed by a local algorithm and set of IP addresses exchanged by each communicating pair of nodes. This local governance is session-independent in that it may govern communications between a pair of nodes, irrespective of the session or end points being transferred between the directly communicating pair of nodes.

In the scalable embodiments, blocks of IP addresses are allocated to each node in the network. (This scalability will increase in the future, when Internet Protocol addresses are increased to 128-bit fields, vastly increasing the number of distinctly addressable nodes). Each node can thus use any of the IP addresses assigned to that node to communicate with other nodes in the network. Indeed, each pair of communicating nodes can use a plurality of source IP addresses and destination IP addresses for communicating with each other.

Each communicating pair of nodes in a chain participating in any session stores two blocks of IP addresses, called netblocks, and an algorithm and randomization seed for selecting, from each netblock, the next pair of source/destination IP addresses that will be used to transmit the next message. In other words, the algorithm governs the sequential selection of IP-address pairs, one sender and one receiver IP address, from each netblock. The combination of algorithm, seed, and netblock (IP address block) will be called a "hopblock." A router issues separate transmit and receive hopblocks to its clients. The send address and the receive address of the IP header of each outgoing packet sent by the client are filled with the send and receive IP addresses generated by the algorithm. The algorithm is "clocked" (indexed) by a counter so that each time a pair is used, the algorithm turns out a new transmit pair for the next packet to be sent.

The router's receive hopblock is identical to the client's transmit hopblock. The router uses the receive hopblock to predict what the send and receive IP address pair for the next expected packet from that client will be. Since packets can be received out of order, it is not possible for the router to predict with certainty what IP address pair will be on the next sequential packet. To account for this problem, the router generates a range of predictions encompassing the number of possible transmitted packet send/receive addresses, of which the next packet received could leap ahead. Thus, if there is a vanishingly small probability that a given packet will arrive at the router ahead of 5 packets transmitted by the client before the given packet, then the router can generate a series of 6 send/receive IP address pairs (or "hop window") to compare with the next received packet. When a packet is received, it is marked in the hop window as such, so that a second packet with the same IP address pair will be discarded. If an out-of-sequence packet does not arrive within a predetermined timeout period, it can be requested for retransmission or simply discarded from the receive table, depending upon the protocol in use for that communications session, or possibly by convention.

When the router receives the client's packet, it compares the send and receive IP addresses of the packet with the next N predicted send and receive IP address pairs and rejects the packet if it is not a member of this set. Received packets that do not have the predicted source/destination IP addresses falling with the window are rejected, thus thwarting possible hackers. (With the number of possible combinations, even a fairly large window would be hard to fall into at random.) If it is a member of this set, the router accepts the packet and processes it further. This link-based IP-hopping strategy, referred to as "IHOP," is a network element that stands on its own and is not necessarily accompanied by elements of the boutique system described above. If the routing agility feature described in connection with the boutique embodiment is combined with this link-based IP-hopping strategy, the router's next step would be to decrypt the TARP header to determine the destination TARP router for the packet and determine what should be the next hop for the packet. The TARP router would then forward the packet to a random TARP router or the destination TARP router with which the source TARP router has a link-based IP hopping communication established.

FIG. **8** shows how a client computer **801** and a TARP router **811** can establish a secure session. When client **801** seeks to establish an IHOP session with TARP router **811**, the client **801** sends "secure synchronization" request ("SSYN") packet **821** to the TARP router **811**. This SYN packet **821** contains the client's **801** authentication token, and may be sent to the router **811** in an encrypted format. The source and destination IP numbers on the packet **821** are the client's **801** current fixed IP address, and a "known" fixed IP address for the router **811**. (For security purposes, it may be desirable to reject any packets from outside of the local network that are destined for the router's known fixed IP address.) Upon receipt and validation of the client's **801** SSYN packet **821**, the router **811** responds by sending an encrypted "secure synchronization acknowledgment" ("SSYN ACK") **822** to the client **801**. This SSYN ACK **822** will contain the transmit and receive hopblocks that the client **801** will use when communicating with the TARP router **811**. The client **801** will acknowledge the TARP router's **811** response packet **822** by generating an encrypted SSYN ACK ACK packet **823** which will be sent from the client's **801** fixed IP address and to the TARP router's **811** known fixed IP address. The client **801** will simultaneously generate a SSYN ACK ACK packet; this SSYN ACK packet, referred to as the Secure Session Initiation (SSI) packet **824**, will be sent with the first {sender, receiver} IP pair in the client's transmit table **921** (FIG. **9**), as specified in the transmit hopblock provided by the TARP router **811** in the SSYN ACK packet **822**. The TARP router **811** will respond to the SSI packet **824** with an SSI ACK packet **825**, which will be sent with the first {sender, receiver} IP pair in the TARP router's transmit table **923**. Once these packets have been successfully exchanged, the secure communications session is established, and all further secure communications between the client **801** and the TARP router **811** will be conducted via this secure session, as long as synchronization is maintained. If synchronization is lost, then the client **801** and TARP router **802** may re-establish the secure session by the procedure outlined in FIG. **8** and described above.

While the secure session is active, both the client **901** and TARP router **911** (FIG. **9**) will maintain their respective transmit tables **921**, **923** and receive tables **922**, **924**, as provided by the TARP router during session synchronization **822**. It is important that the sequence of IP pairs in the client's transmit table **921** be identical to those in the TARP router's receive table **924**; similarly, the sequence of IP pairs in the client's receive table **922** must be identical to those in the router's transmit table **923**. This is required for the session synchronization to be maintained. The client **901** need maintain only one transmit table **921** and one receive table **922** during the course of the secure session. Each sequential packet sent by the client **901** will employ the next {send, receive} IP address pair in the transmit table, regardless of TCP or UDP session. The TARP router **911** will expect each packet arriving from the client **901** to bear the next IP address pair shown in its receive table.

Since packets can arrive out of order, however, the router **911** can maintain a "look ahead" buffer in its receive table, and will mark previously-received IP pairs as invalid for future packets; any future packet containing an IP pair that is in the look-ahead buffer but is marked as previously received will be discarded. Communications from the TARP router **911** to the client **901** are maintained in an identical manner; in particular, the router **911** will select the next IP address pair from its transmit table **923** when constructing a packet to send to the client **901**, and the client **901** will maintain a look-ahead buffer of expected IP pairs on packets that it is receiving. Each TARP router will maintain separate pairs of transmit and receive tables for each client that is currently engaged in a secure session with or through that TARP router.

While clients receive their hopblocks from the first server linking them to the Internet, routers exchange hopblocks. When a router establishes a link-based IP-hopping communication regime with another router, each router of the pair exchanges its transmit hopblock. The transmit hopblock of each router becomes the receive hopblock of the other router. The communication between routers is governed as described by the example of a client sending a packet to the first router.

While the above strategy works fine in the IP milieu, many local networks that are connected to the Internet are Ethernet systems. In Ethernet, the IP addresses of the destination devices must be translated into hardware addresses, and vice versa, using known processes ("address resolution protocol," and "reverse address resolution protocol"). However, if the link-based IP-hopping strategy is employed, the correlation process would become explosive and burdensome. An alternative to the link-based IP hopping strategy may be employed within an Ethernet network. The solution is to provide that the node linking the Internet to the Ethernet (call it the border node) use the link-based IP-hopping communication regime to communicate with nodes outside the Ethernet LAN. Within the Ethernet LAN, each TARP node would have a single IP address which would be addressed in the conventional way. Instead of comparing the {sender, receiver} IP address pairs to authenticate a packet, the intra-LAN TARP node would use one of the IP header extension fields to do so. Thus, the border node uses an algorithm shared by the intra-LAN TARP node to generate a symbol that is stored in the free field in the IP header, and the intra-LAN TARP node generates a range of symbols based on its prediction of the next expected packet to be received from that particular source IP address. The packet is rejected if it does not fall into the set of predicted symbols (for example, numerical values) or is accepted if it does. Communications from the intra-LAN TARP node to the border node are accomplished in the same manner, though the algorithm will necessarily be different for security reasons. Thus, each of the communicating nodes will generate transmit and receive tables in a similar manner to that of FIG. **9**; the intra-LAN TARP nodes transmit table will be identical to the border node's receive table, and the intra-LAN TARP node's receive table will be identical to the border node's transmit table.

The algorithm used for IP address-hopping can be any desired algorithm. For example, the algorithm can be a given pseudo-random number generator that generates numbers of the range covering the allowed IP addresses with a given seed. Alternatively, the session participants can assume a certain type of algorithm and specify simply a parameter for applying the algorithm. For example the assumed algorithm could be a particular pseudo-random number generator and the session participants could simply exchange seed values.

Note that there is no permanent physical distinction between the originating and destination terminal nodes. Either device at either end point can initiate a synchronization of the pair. Note also that the authentication/synchronization-request (and acknowledgment) and hopblock-exchange may all be served by a single message so that separate message exchanges may not be required.

As another extension to the stated architecture, multiple physical paths can be used by a client, in order to provide link redundancy and further thwart attempts at denial of service

and traffic monitoring. As shown in FIG. **10**, for example, client **1001** can establish three simultaneous sessions with each of three TARP routers provided by different ISPs **1011**, **1012**, **1013**. As an example, the client **1001** can use three different telephone lines **1021**, **1022**, **1023** to connect to the ISPs, or two telephone lines and a cable modem, etc. In this scheme, transmitted packets will be sent in a random fashion among the different physical paths. This architecture provides a high degree of communications redundancy, with improved immunity from denial-of-service attacks and traffic monitoring.

2. Further Extensions

The following describes various extensions to the techniques, systems, and methods described above. As described above, the security of communications occurring between computers in a computer network (such as the Internet, an Ethernet, or others) can be enhanced by using seemingly random source and destination Internet Protocol (IP) addresses for data packets transmitted over the network. This feature prevents eavesdroppers from determining which computers in the network are communicating with each other while permitting the two communicating computers to easily recognize whether a given received data packet is legitimate or not. In one embodiment of the above-described systems, an IP header extension field is used to authenticate incoming packets on an Ethernet.

Various extensions to the previously described techniques described herein include: (1) use of hopped hardware or "MAC" addresses in broadcast type network; (2) a self-synchronization technique that permits a computer to automatically regain synchronization with a sender; (3) synchronization algorithms that allow transmitting and receiving computers to quickly re-establish synchronization in the event of lost packets or other events; and (4) a fast-packet rejection mechanism for rejecting invalid packets. Any or all of these extensions can be combined with the features described above in any of various ways.

A. Hardware Address Hopping

Internet protocol-based communications techniques on a LAN—or across any dedicated physical medium—typically embed the IP packets within lower-level packets, often referred to as "frames." As shown in FIG. **11**, for example, a first Ethernet frame **1150** comprises a frame header **1101** and two embedded IP packets IP1 and IP2, while a second Ethernet frame **1160** comprises a different frame header **1104** and a single IP packet IP3. Each frame header generally includes a source hardware address **1101A** and a destination hardware address **10B**; other well-known fields in frame headers are omitted from FIG. **11** for clarity. Two hardware nodes communicating over a physical communication channel insert appropriate source and destination hardware addresses to indicate which nodes on the channel or network should receive the frame.

It may be possible for a nefarious listener to acquire information about the contents of a frame and/or its communicants by examining frames on a local network rather than (or in addition to) the IP packets themselves. This is especially true in broadcast media, such as Ethernet, where it is necessary to insert into the frame header the hardware address of the machine that generated the frame and the hardware address of the machine to which frame is being sent. All nodes on the network can potentially "see" all packets transmitted across the network. This can be a problem for secure communications, especially in cases where the communicants do not want for any third party to be able to identify who is engaging in the information exchange. One way to address this problem is to push the address-hopping scheme down to the hardware layer. In accordance with various embodiments of the invention, hardware addresses are "hopped" in a manner similar to that used to change IP addresses, such that a listener cannot determine which hardware node generated a particular message nor which node is the intended recipient.

FIG. 12A shows a system in which Media Access Control ("MAC") hardware addresses are "hopped" in order to increase security over a network such as an Ethernet. While the description refers to the exemplary case of an Ethernet environment, the inventive principles are equally applicable to other types of communications media. In the Ethernet case, the MAC address of the sender and receiver are inserted into the Ethernet frame and can be observed by anyone on the LAN who is within the broadcast range for that frame. For secure communications, it becomes desirable to generate frames with MAC addresses that are not attributable to any specific sender or receiver.

As shown in FIG. 12A, two computer nodes **1201** and **1202** communicate over a communication channel such as an Ethernet. Each node executes one or more application programs **1203** and **1218** that communicate by transmitting packets through communication software **1204** and **1217**, respectively. Examples of application programs include video conferencing, e-mail, word processing programs, telephony, and the like. Communication software **1204** and **1217** can comprise, for example, an OSI layered architecture or "stack" that standardizes various services provided at different levels of functionality.

The lowest levels of communication software **1204** and **1217** communicate with hardware components **1206** and **1214** respectively, each of which can include one or more registers **1207** and **1215** that allow the hardware to be reconfigured or controlled in accordance with various communication protocols. The hardware components (an Ethernet network interface card, for example) communicate with each other over the communication medium. Each hardware component is typically pre-assigned a fixed hardware address or MAC number that identifies the hardware component to other nodes on the network. One or more interface drivers control the operation of each card and can, for example, be configured to accept or reject packets from certain hardware addresses. As will be described in more detail below, various embodiments of the inventive principles provide for "hopping" different addresses using one or more algorithms and one or more moving windows that track a range of valid addresses to validate received packets. Packets transmitted according to one or more of the inventive principles will be generally referred to as "secure" packets or "secure communications" to differentiate them from ordinary data packets that are transmitted in the clear using ordinary, machine-correlated addresses.

One straightforward method of generating non-attributable MAC addresses is an extension of the IP hopping scheme. In this scenario, two machines on the same LAN that desire to communicate in a secure fashion exchange random-number generators and seeds, and create sequences of quasi-random MAC addresses for synchronized hopping. The implementation and synchronization issues are then similar to that of IP hopping.

This approach, however, runs the risk of using MAC addresses that are currently active on the LAN—which, in turn, could interrupt communications for those machines. Since an Ethernet MAC address is at present 48 bits in length, the chance of randomly misusing an active MAC address is actually quite small. However, if that figure is multiplied by a large number of nodes (as would be found on an extensive LAN), by a large number of frames (as might be the case with packet voice or streaming video), and by a large number of concurrent Virtual Private Networks (VPNs), then the chance that a non-secure machine's MAC address could be used in an address-hopped frame can become non-trivial. In short, any scheme that runs even a small risk of interrupting communications for other machines on the LAN is bound to receive resistance from prospective system administrators. Nevertheless, it is technically feasible, and can be implemented without risk on a LAN on which there is a small number of machines, or if all of the machines on the LAN are engaging in MAC-hopped communications.

Synchronized MAC address hopping may incur some overhead in the course of session establishment, especially if there are multiple sessions or multiple nodes involved in the communications. A simpler method of randomizing MAC addresses is to allow each node to receive and process every incident frame on the network. Typically, each network interface driver will check the destination MAC address in the header of every incident frame to see if it matches that machine's MAC address; if there is no match, then the frame is discarded. In one embodiment, however, these checks can be disabled, and every incident packet is passed to the TARP stack for processing. This will be referred to as "promiscuous" mode, since every incident frame is processed. Promiscuous mode allows the sender to use completely random, unsynchronized MAC addresses, since the destination machine is guaranteed to process the frame. The decision as to whether the packet was truly intended for that machine is handled by the TARP stack, which checks the source and destination IP addresses for a match in its IP synchronization tables. If no match is found, the packet is discarded; if there is a match, the packet is unwrapped, the inner header is evaluated, and if the inner header indicates that the packet is destined for that machine then the packet is forwarded to the IP stack—otherwise it is discarded.

One disadvantage of purely-random MAC address hopping is its impact on processing overhead; that is, since every incident frame must be processed, the machine's CPU is engaged considerably more often than if the network interface driver is discriminating and rejecting packets unilaterally. A compromise approach is to select either a single fixed MAC address or a small number of MAC addresses (e.g., one for each virtual private network on an Ethernet) to use for MAC-hopped communications, regardless of the actual recipient for which the message is intended. In this mode, the network interface driver can check each incident frame against one (or a few) pre-established MAC addresses, thereby freeing the CPU from the task of physical-layer packet discrimination. This scheme does not betray any useful information to an interloper on the LAN; in particular, every secure packet can already be identified by a unique packet type in the outer header. However, since all machines engaged in secure communications would either be using the same MAC address, or be selecting from a small pool of predetermined MAC addresses, the association between a specific machine and a specific MAC address is effectively broken.

In this scheme, the CPU will be engaged more often than it would be in non-secure communications (or in synchronized MAC address hopping), since the network interface driver cannot always unilaterally discriminate between secure packets that are destined for that machine, and secure packets from other VPNs. However, the non-secure traffic is easily eliminated at the network interface, thereby reducing the amount of processing required of the CPU. There are boundary conditions where these statements would not hold, of course—e.g., if all of the traffic on the LAN is secure traffic, then the CPU would be engaged to the same degree as it is in the purely-random address hopping case; alternatively, if each VPN on the LAN uses a different MAC address, then the network interface can perfectly discriminate secure frames destined for the local machine from those constituting other VPNs. These are engineering tradeoffs that might be best handled by providing administrative options for the users when installing the software and/or establishing VPNs.

Even in this scenario, however, there still remains a slight risk of selecting MAC addresses that are being used by one or more nodes on the LAN. One solution to this problem is to formally assign one address or a range of addresses for use in MAC-hopped communications. This is typically done via an assigned numbers registration authority; e.g., in the case of Ethernet, MAC address ranges are assigned to vendors by the Institute of Electrical and Electronics Engineers (IEEE). A formally-assigned range of addresses would ensure that secure frames do not conflict with any properly-configured and properly-functioning machines on the LAN.

Reference will now be made to FIGS. 12A and 12B in order to describe the many combinations and features that follow the inventive principles. As explained above, two computer nodes **1201** and **1202** are assumed to be communicating over a network or communication medium such as an Ethernet. A communication protocol in each node (**1204** and **1217**, respectively) contains a modified element **1205** and **1216** that performs certain functions that deviate from the standard communication protocols. In particular, computer node **1201** implements a first "hop" algorithm **1208X** that selects seemingly random source and destination IP addresses (and, in one embodiment, seemingly random IP header discriminator fields) in order to transmit each packet to the other computer node. For example, node **1201** maintains a transmit table **1208** containing triplets of source (S), destination (D), and discriminator fields (DS) that are inserted into outgoing IP packet headers. The table is generated through the use of an appropriate algorithm (e.g., a random number generator that is seeded with an appropriate seed) that is known to the recipient node **1202**. As each new IP packet is formed, the next sequential entry out of the sender's transmit table **1208** is used to populate the IP source, IP destination, and IP header extension field (e.g., discriminator field). It will be appreciated that the transmit table need not be created in advance but could instead be created on-the-fly by executing the algorithm when each packet is formed.

At the receiving node **1202**, the same IP hop algorithm **1222X** is maintained and used to generate a receive table **1222** that lists valid triplets of source IP address, destination IP address, and discriminator field. This is shown by virtue of the first five entries of transmit table **1208** matching the second five entries of receive table **1222**. (The tables may be slightly offset at any particular time due to lost packets, misordered packets, or transmission delays). Additionally, node **1202** maintains a receive window W3 that represents a list of valid IP source, IP destination, and discriminator fields that will be accepted when received as part of an incoming IP packet. As packets are received, window W3 slides down the list of valid entries, such that the possible valid entries change over time. Two packets that arrive out of order but are nevertheless matched to entries within window W3 will be accepted; those falling outside of window W3 will be rejected as invalid. The length of window W3 can be adjusted as necessary to reflect network delays or other factors.

Node **1202** maintains a similar transmit table **1221** for creating IP packets and frames destined for node **1201** using a potentially different hopping algorithm **1221X**, and node **1201** maintains a matching receive table **1209** using the same algorithm **1209X**. As node **1202** transmits packets to node **1201** using seemingly random IP source, IP destination, and/or discriminator fields, node **1201** matches the incoming packet values to those falling within window **W1** maintained in its receive table. In effect, transmit table **1208** of node **1201** is synchronized (i.e., entries are selected in the same order) to receive table **1222** of receiving node **1202**. Similarly, transmit table **1221** of node **1202** is synchronized to receive table **1209** of node **1201**. It will be appreciated that although a common algorithm is shown for the source, destination and discriminator fields in FIG. **12A** (using, e.g., a different seed for each of the three fields), an entirely different algorithm could in fact be used to establish values for each of these fields. It will also be appreciated that one or two of the fields can be "hopped" rather than all three as illustrated.

In accordance with another aspect of the invention, hardware or "MAC" addresses are hopped instead of or in addition to IP addresses and/or the discriminator field in order to improve security in a local area or broadcast-type network. To that end, node **1201** further maintains a transmit table **1210** using a transmit algorithm **1210X** to generate source and destination hardware addresses that are inserted into frame headers (e.g., fields **1101A** and **1101B** in FIG. **11**) that are synchronized to a corresponding receive table **1224** at node **1202**. Similarly, node **1202** maintains a different transmit table **1223** containing source and destination hardware addresses that is synchronized with a corresponding receive table **1211** at node **1201**. In this manner, outgoing hardware frames appear to be originating from and going to completely random nodes on the network, even though each recipient can determine whether a given packet is intended for it or not. It will be appreciated that the hardware hopping feature can be implemented at a different level in the communications protocol than the IP hopping feature (e.g., in a card driver or in a hardware card itself to improve performance).

FIG. **12B** shows three different embodiments or modes that can be employed using the aforementioned principles. In a first mode referred to as "promiscuous" mode, a common hardware address (e.g., a fixed address for source and another for destination) or else a completely random hardware address is used by all nodes on the network, such that a particular packet cannot be attributed to any one node. Each node must initially accept all packets containing the common (or random) hardware address and inspect the IP addresses or discriminator field to determine whether the packet is intended for that node. In this regard, either the IP addresses or the discriminator field or both can be varied in accordance with an algorithm as described above. As explained previously, this may increase each node's overhead-since additional processing is involved to determine whether a given packet has valid source and destination hardware addresses.

In a second mode referred to as "promiscuous per VPN" mode, a small set of fixed hardware addresses are used, with a fixed source/destination hardware address used for all nodes communicating over a virtual private network. For example, if there are six nodes on an Ethernet, and the network is to be split up into two private virtual networks such that nodes on one VPN can communicate with only the other two nodes on its own VPN, then two sets of hardware addresses could be used: one set for the first VPN and a second set for the second VPN. This would reduce the amount of overhead involved in checking for valid frames since only packets arriving from the designated VPN would need to be checked. IP addresses and one or more discriminator fields could still be hopped as before for secure communication within the VPN. Of course, this solution compromises the anonymity of the VPNs (i.e., an outsider can easily tell what traffic belongs in which VPN, though he cannot correlate it to a specific machine/person). It also requires the use of a discriminator field to mitigate the vulnerability to certain types of DoS attacks. (For example, without the discriminator field, an attacker on the LAN could stream frames containing the MAC addresses being used by the VPN; rejecting those frames could lead to excessive processing overhead. The discriminator field would provide a low-overhead means of rejecting the false packets.)

In a third mode referred to as "hardware hopping" mode, hardware addresses are varied as illustrated in FIG. 12A, such that hardware source and destination addresses are changed constantly in order to provide non-attributable addressing. Variations on these embodiments are of course possible, and the invention is not intended to be limited in any respect by these illustrative examples.

B. Extending the Address Space

Address hopping provides security and privacy. However, the level of protection is limited by the number of addresses in the blocks being hopped. A hopblock denotes a field or fields modulated on a packet-wise basis for the purpose of providing a VPN. For instance, if two nodes communicate with IP address hopping using hopblocks of 4 addresses (2 bits) each, there would be 16 possible address-pair combinations. A window of size 16 would result in most address pairs being accepted as valid most of the time. This limitation can be overcome by using a discriminator field in addition to or instead of the hopped address fields. The discriminator field would be hopped in exactly the same fashion as the address fields and it would be used to determine whether a packet should be processed by a receiver.

Suppose that two clients, each using four-bit hopblocks, would like the same level of protection afforded to clients communicating via IP hopping between two A blocks (24 address bits eligible for hopping). A discriminator field of 20 bits, used in conjunction with the 4 address bits eligible for hopping in the IP address field, provides this level of protection. A 24-bit discriminator field would provide a similar level of protection if the address fields were not hopped or ignored. Using a discriminator field offers the following advantages: (1) an arbitrarily high level of protection can be provided, and (2) address hopping is unnecessary to provide protection. This may be important in environments where address hopping would cause routing problems.

C. Synchronization Techniques

It is generally assumed that once a sending node and receiving node have exchanged algorithms and seeds (or similar information sufficient to generate quasi-random source and destination tables), subsequent communication between the two nodes will proceed smoothly. Realistically, however, two nodes may lose synchronization due to network delays or outages, or other problems. Consequently, it is desirable to provide means for re-establishing synchronization between nodes in a network that have lost synchronization.

One possible technique is to require that each node provide an acknowledgment upon successful receipt of each packet and, if no acknowledgment is received within a certain period of time, to re-send the unacknowledged packet. This approach, however, drives up overhead costs and may be

prohibitive in high-throughput environments such as streaming video or audio, for example.

A different approach is to employ an automatic synchronizing technique that will be referred to herein as "self-synchronization." In this approach, synchronization information is embedded into each packet, thereby enabling the receiver to re-synchronize itself upon receipt of a single packet if it determines that is has lost synchronization with the sender. (If communications are already in progress, and the receiver determines that it is still in sync with the sender, then there is no need to re-synchronize.) A receiver could detect that it was out of synchronization by, for example, employing a "deadman" timer that expires after a certain period of time, wherein the timer is reset with each valid packet. A time stamp could be hashed into the public sync field (see below) to preclude packet-retry attacks.

In one embodiment, a "sync field" is added to the header of each packet sent out by the sender. This sync field could appear in the clear or as part of an encrypted portion of the packet. Assuming that a sender and receiver have selected a random-number generator (RNG) and seed value, this combination of RNG and seed can be used to generate a random-number sequence (RNS). The RNS is then used to generate a sequence of source/destination IP pairs (and, if desired, discriminator fields and hardware source and destination addresses), as described above. It is not necessary, however, to generate the entire sequence (or the first N−1 values) in order to generate the Nth random number in the sequence; if the sequence index N is known, the random value corresponding to that index can be directly generated (see below). Different RNGs (and seeds) with different fundamental periods could be used to generate the source and destination IP sequences, but the basic concepts would still apply. For the sake of simplicity, the following discussion will assume that IP source and destination address pairs (only) are hopped using a single RNG sequencing mechanism.

In accordance with a "self-synchronization" feature, a sync field in each packet header provides an index (i.e., a sequence number) into the RNS that is being used to generate IP pairs. Plugging this index into the RNG that is being used to generate the RNS yields a specific random number value, which in turn yields a specific IP pair. That is, an IP pair can be generated directly from knowledge of the RNG, seed, and index number; it is not necessary, in this scheme, to generate the entire sequence of random numbers that precede the sequence value associated with the index number provided.

Since the communicants have presumably previously exchanged RNGs and seeds, the only new information that must be provided in order to generate an IP pair is the sequence number. If this number is provided by the sender in the packet header, then the receiver need only plug this number into the RNG in order to generate an IP pair—and thus verify that the IP pair appearing in the header of the packet is valid. In this scheme, if the sender and receiver lose synchronization, the receiver can immediately re-synchronize upon receipt of a single packet by simply comparing the IP pair in the packet header to the IP pair generated from the index number. Thus, synchronized communications can be resumed upon receipt of a single packet, making this scheme ideal for multicast communications. Taken to the extreme, it could obviate the need for synchronization tables entirely; that is, the sender and receiver could simply rely on the index number in the sync field to validate the IP pair on each packet, and thereby eliminate the tables entirely.

The aforementioned scheme may have some inherent security issues associated with it namely, the placement of the sync field. If the field is placed in the outer header, then an interloper could observe the values of the field and their relationship to the IP stream. This could potentially compromise the algorithm that is being used to generate the IP-address sequence, which would compromise the security of the communications. If, however, the value is placed in the inner header, then the sender must decrypt the inner header before it can extract the sync value and validate the IP pair; this opens up the receiver to certain types of denial-of-service (DoS) attacks, such as packet replay. That is, if the receiver must decrypt a packet before it can validate the IP pair, then it could potentially be forced to expend a significant amount of processing on decryption if an attacker simply retransmits previously valid packets. Other attack methodologies are possible in this scenario.

A possible compromise between algorithm security and processing speed is to split up the sync value between an inner (encrypted) and outer (unencrypted) header. That is, if the sync value is sufficiently long, it could potentially be split into a rapidly-changing part that can be viewed in the clear, and a fixed (or very slowly changing) part that must be protected. The part that can be viewed in the clear will be called the "public sync" portion and the part that must be protected will be called the "private sync" portion.

Both the public sync and private sync portions are needed to generate the complete sync value. The private portion, however, can be selected such that it is fixed or will change only occasionally. Thus, the private sync value can be stored by the recipient, thereby obviating the need to decrypt the header in order to retrieve it. If the sender and receiver have previously agreed upon the frequency with which the private part of the sync will change, then the receiver can selectively decrypt a single header in order to extract the new private sync if the communications gap that has led to lost synchronization has exceeded the lifetime of the previous private sync. This should not represent a burdensome amount of decryption, and thus should not open up the receiver to denial-of-service attack simply based on the need to occasionally decrypt a single header.

One implementation of this is to use a hashing function with a one-to-one mapping to generate the private and public sync portions from the sync value. This implementation is shown in FIG. **13**, where (for example) a first ISP **1302** is the sender and a second ISP **1303** is the receiver. (Other alternatives are possible from FIG. **13**.) A transmitted packet comprises a public or "outer" header **1305** that is not encrypted, and a private or "inner" header **1306** that is encrypted using for example a link key. Outer header **1305** includes a public sync portion while inner header **1306** contains the private sync portion. A receiving node decrypts the inner header using a decryption function **1307** in order to extract the private sync portion. This step is necessary only if the lifetime of the currently buffered private sync has expired. (If the currently-buffered private sync is still valid, then it is simply extracted from memory and "added" (which could be an inverse hash) to the public sync, as shown in step **1308**.) The public and decrypted private sync portions are combined in function **1308** in order to generate the combined sync **1309**. The combined sync (**1309**) is then fed into the RNG (**1310**) and compared to the IP address pair (**1311**) to validate or reject the packet.

An important consideration in this architecture is the concept of "future" and "past" where the public sync values are concerned. Though the sync values, themselves, should be random to prevent spoofing attacks, it may be important that the receiver be able to quickly identify a sync value that has already been sent—even if the packet containing that sync value was never actually received by the receiver. One solution is to hash a time stamp or sequence number into the public sync portion, which could be quickly extracted, checked, and discarded, thereby validating the public sync portion itself.

In one embodiment, packets can be checked by comparing the source/destination IP pair generated by the sync field with the pair appearing in the packet header. If (1) they match, (2) the time stamp is valid, and (3) the dead-man timer has expired, then re-synchronization occurs; otherwise, the packet is rejected. If enough processing power is available, the dead-man timer and synchronization tables can be avoided altogether, and the receiver would simply resynchronize (e.g., validate) on every packet.

The foregoing scheme may require large-integer (e.g., 160-bit) math, which may affect its implementation. Without such large-integer registers, processing throughput would be affected, thus potentially affecting security from a denial-of-service standpoint. Nevertheless, as large-integer math processing features become more prevalent, the costs of implementing such a feature will be reduced.

D. Other Synchronization Schemes

As explained above, if W or more consecutive packets are lost between a transmitter and receiver in a VPN (where W is the window size), the receiver's window will not have been updated and the transmitter will be transmitting packets not in the receiver's window. The sender and receiver will not recover synchronization until perhaps the random pairs in the window are repeated by chance. Therefore, there is a need to keep a transmitter and receiver in synchronization whenever possible and to re-establish synchronization whenever it is lost.

A "checkpoint" scheme can be used to regain synchronization between a sender and a receiver that have fallen out of synchronization. In this scheme, a checkpoint message comprising a random IP address pair is used for communicating synchronization information. In one embodiment, two messages are used to communicate synchronization information between a sender and a recipient:

1. SYNC_REQ is a message used by the sender to indicate that it wants to synchronize; and
2. SYNC_ACK is a message used by the receiver to inform the transmitter that it has been synchronized.

According to one variation of this approach, both the transmitter and receiver maintain three checkpoints (see FIG. **14**):

1. In the transmitter, ckpt_o ("checkpoint old") is the IP pair that was used to re-send the last SYNC_REQ packet to the receiver. In the receiver, ckpt_o ("checkpoint old") is the IP pair that receives repeated SYNC_REQ packets from the transmitter.
2. In the transmitter, ckpt_n ("checkpoint new") is the IP pair that will be used to send the next SYNC_REQ packet to the receiver. In the receiver, ckpt_n ("checkpoint new") is the IP pair that receives a new SYNC_REQ packet from the transmitter and which causes the receiver's window to be re-aligned, ckpt_o set to ckpt_n, a new ckpt_n to be generated and a new ckpt_r to be generated.
3. In the transmitter, ckpt_r is the IP pair that will be used to send the next SYNC_ACK packet to the receiver. In the receiver, ckpt_r is the IP pair that receives a new SYNC_ACK packet from the transmitter and which

causes a new ckpt_n to be generated. Since SYNC_ACK is transmitted from the receiver ISP to the sender ISP, the transmitter ckpt_r refers to the ckpt_r of the receiver and the receiver ckpt_r refers to the ckpt_r of the transmitter (see FIG. 14).

When a transmitter initiates synchronization, the IP pair it will use to transmit the next data packet is set to a predetermined value and when a receiver first receives a SYNC_REQ, the receiver window is updated to be centered on the transmitter's next IP pair. This is the primary mechanism for checkpoint synchronization.

Synchronization can be initiated by a packet counter (e.g., after every N packets transmitted, initiate a synchronization) or by a timer (every S seconds, initiate a synchronization) or a combination of both. See FIG. 15. From the transmitter's perspective, this technique operates as follows: (1) Each transmitter periodically transmits a "sync request" message to the receiver to make sure that it is in sync. (2) If the receiver is still in sync, it sends back a "sync ack" message. (If this works, no further action is necessary). (3) If no "sync ack" has been received within a period of time, the transmitter retransmits the sync request again. If the transmitter reaches the next checkpoint without receiving a "sync ack" response, then synchronization is broken, and the transmitter should stop transmitting. The transmitter will continue to send sync_reqs until it receives a sync_ack, at which point transmission is reestablished.

From the receiver's perspective, the scheme operates as follows: (1) when it receives a "sync request" request from the transmitter, it advances its window to the next checkpoint position (even skipping pairs if necessary), and sends a "sync ack" message to the transmitter. If sync was never lost, then the "jump ahead" really just advances to the next available pair of addresses in the table (i.e., normal advancement).

If an interloper intercepts the "sync request" messages and tries to interfere with communication by sending new ones, it will be ignored if the synchronization has been established or it will actually help to re-establish synchronization.

A window is realigned whenever a re-synchronization occurs. This realignment entails updating the receiver's window to straddle the address pairs used by the packet transmitted immediately after the transmission of the SYNC_REQ packet. Normally, the transmitter and receiver are in synchronization with one another. However, when network events occur, the receiver's window may have to be advanced by many steps during resynchronization. In this case, it is desirable to move the window ahead without having to step through the intervening random numbers sequentially. (This feature is also desirable for the auto-sync approach discussed above).

E. Random Number Generator with a Jump-Ahead capability

An attractive method for generating randomly hopped addresses is to use identical random number generators in the transmitter and receiver and advance them as packets are transmitted and received. There are many random number generation algorithms that could be used. Each one has strengths and weaknesses for address hopping applications.

Linear congruential random number generators (LCRs) are fast, simple and well characterized random number generators that can be made to jump ahead n steps efficiently. An LCR generates random numbers $X_1, X_2, X_3 \ldots X_k$ starting with seed X_0 using a recurrence

$$X_i=(a\,X_{i-1}+b) \bmod c, \qquad (1)$$

where a, b and c define a particular LCR. Another expression for X_i,

$$X_i=((a^i(X_0+b)-b)/(a-1)) \bmod c \qquad (2)$$

enables the jump-ahead capability. The factor a^i can grow very large even for modest i if left unfettered. Therefore some special properties of the modulo operation can be used to control the size and processing time required to compute (2). (2) can be rewritten as:

$$X_i=(a^i(X_0(a-1)+b)-b)/(a-1) \bmod c. \qquad (3)$$

It can be shown that:

$$(a_i(X_0(a-1)+b)-b)/(a-1) \bmod c=((a^i \bmod ((a-1)c)(X_0(a-1)+b)-b)/(a-1)) \bmod c \qquad (4).$$

$(X_0(a-1)+b)$ can be stored as $(X_0(a-1)+b) \bmod c$, b as b mod c and compute $a^i \bmod((a-1)c)$ (this requires O(log(i)) steps).

A practical implementation of this algorithm would jump a fixed distance, n, between synchronizations; this is tantamount to synchronizing every n packets. The window would commence n IP pairs from the start of the previous window. Using X_j^w, the random number at the j^{th} checkpoint, as X_0 and n as i, a node can store $a^n \bmod((a-1)c)$ once per LCR and set

$$X_{j+1}^w=X_{n(j+1)}=((a^n \bmod ((a-1)c)(X_j^w(a-1)+b)-b)/(a-1)) \bmod c, \qquad (5)$$

to generate the random number for the $j+1^{th}$ synchronization. Using this construction, a node could jump ahead an arbitrary (but fixed) distance between synchronizations in a constant amount of time (independent of n).

Pseudo-random number generators, in general, and LCRs, in particular, will eventually repeat their cycles. This repetition may present vulnerability in the IP hopping scheme. An adversary would simply have to wait for a repeat to predict future sequences. One way of coping with this vulnerability is to create a random number generator with a known long cycle. A random sequence can be replaced by a new random number generator before it repeats. LCRs can be constructed with known long cycles. This is not currently true of many random number generators.

Random number generators can be cryptographically insecure. An adversary can derive the RNG parameters by examining the output or part of the output. This is true of LCGs. This vulnerability can be mitigated by incorporating an encryptor, designed to scramble the output as part of the random number generator. The random number generator prevents an adversary from mounting an attack—e.g., a known plaintext attack—against the encryptor.

F. Random Number Generator Example

Consider a RNG where a=31, b=4 and c=15. For this case equation (1) becomes:

$$X_i=(31X_{i-1}+4) \bmod 15. \qquad (6)$$

If one sets $X_0=1$, equation (6) will produce the sequence 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 0, 4, 8, 12. This sequence will repeat indefinitely. For a jump ahead of 3 numbers in this sequence $a^n=31^3=29791$, $c*(a-1)=15*30=450$ and $a^n \bmod ((a-1)c)=31^3 \bmod(15*30)=29791 \bmod(450)=91$. Equation (5) becomes:

$$((91\ (X_i30+4)-4)/30) \bmod 15 \qquad (7).$$

Table 1 shows the jump ahead calculations from (7). The calculations start at 5 and jump ahead 3.

TABLE 1

I	X_i	$(X_i30 + 4)$	$91\ (X_i30 + 4) - 4$	$((91\ (X_i30 + 4) - 4)/30$	X_{i+3}
1	5	154	14010	467	2
4	2	64	5820	194	14
7	14	424	38580	1286	11
10	11	334	30390	1013	8
13	8	244	22200	740	5

G. Fast Packet Filter

Address hopping VPNs must rapidly determine whether a packet has a valid header and thus requires further processing, or has an invalid header (a hostile packet) and should be immediately rejected. Such rapid determinations will be referred to as "fast packet filtering." This capability protects the VPN from attacks by an adversary who streams hostile packets at the receiver at a high rate of speed in the hope of saturating the receiver's processor (a so-called "denial of service" attack). Fast packet filtering is an important feature for implementing VPNs on shared media such as Ethernet.

Assuming that all participants in a VPN share an unassigned "A" block of addresses, one possibility is to use an experimental "A" block that will never be assigned to any machine that is not address hopping on the shared medium. "A" blocks have a 24 bits of address that can be hopped as opposed to the 8 bits in "C" blocks. In this case a hopblock will be the "A" block. The use of the experimental "A" block is a likely option on an Ethernet because:

1. The addresses have no validity outside of the Ethernet and will not be routed out to a valid outside destination by a gateway.
2. There are 2^{24} (~16 million) addresses that can be hopped within each "A" block. This yields >280 trillion possible address pairs making it very unlikely that an adversary would guess a valid address. It also provides acceptably low probability of collision between separate VPNs (all VPNs on a shared medium independently generate random address pairs from the same "A" block).
3. The packets will not be received by someone on the Ethernet who is not on a VPN (unless the machine is in promiscuous mode) minimizing impact on non-VPN computers.

The Ethernet example will be used to describe one implementation of fast packet filtering. The ideal algorithm would quickly examine a packet header, determine whether the packet is hostile, and reject any hostile packets or determine which active IP pair the packet header matches. The problem is a classical associative memory problem. A variety of techniques have been developed to solve this problem (hashing, B-trees etc). Each of these approaches has its strengths and weaknesses. For instance, hash tables can be made to operate quite fast in a statistical sense, but can occasionally degenerate into a much slower algorithm. This slowness can persist for a period of time. Since there is a need to discard hostile packets quickly at all times, hashing would be unacceptable.

H. Presence Vector Algorithm

A presence vector is a bit vector of length 2^n that can be indexed by n-bit numbers (each ranging from 0 to 2^{n-1}). One can indicate the presence of k n-bit numbers (not necessarily unique), by setting the bits in the presence vector indexed by each number to 1. Otherwise, the bits in the presence vector are 0. An n-bit number, x, is one of the k numbers if and only if the x^{th} bit of the presence vector is 1. A fast packet filter can be implemented by indexing the presence vector and looking for a 1, which will be referred to as the "test."

For example, suppose one wanted to represent the number 135 using a presence vector. The 135^{th} bit of the vector would be set. Consequently, one could very quickly determine whether an address of 135 was valid by checking only one bit: the 135^{th} bit. The presence vectors could be created in advance corresponding to the table entries for the IP addresses. In effect, the incoming addresses can be used as indices into a long vector, making comparisons very fast. As each RNG generates a new address, the presence vector is updated to reflect the information. As the window moves, the presence vector is updated to zero out addresses that are no longer valid.

There is a trade-off between efficiency of the test and the amount of memory required for storing the presence vector(s). For instance, if one were to use the 48 bits of hopping addresses as an index, the presence vector would have to be 35 terabytes. Clearly, this is too large for practical purposes. Instead, the 48 bits can be divided into several smaller fields. For instance, one could subdivide the 48 bits into four 12-bit fields (see FIG. **16**). This reduces the storage requirement to 2048 bytes at the expense of occasionally having to process a hostile packet. In effect, instead of one long presence vector, the decomposed address portions must match all four shorter presence vectors before further processing is allowed. (If the first part of the address portion doesn't match the first presence vector, there is no need to check the remaining three presence vectors).

A presence vector will have a 1 in the y^{th} bit if and only if one or more addresses with a corresponding field of y are active. An address is active only if each presence vector indexed by the appropriate sub-field of the address is 1.

Consider a window of 32 active addresses and 3 checkpoints. A hostile packet will be rejected by the indexing of one presence vector more than 99% of the time. A hostile packet will be rejected by the indexing of all 4 presence vectors more than 99.9999995% of the time. On average, hostile packets will be rejected in less than 1.02 presence vector index operations.

The small percentage of hostile packets that pass the fast packet filter will be rejected when matching pairs are not found in the active window or are active checkpoints. Hostile packets that serendipitously match a header will be rejected when the VPN software attempts to decrypt the header. However, these cases will be extremely rare. There are many other ways this method can be configured to arbitrate the space/speed tradeoffs.

I. Further Synchronization Enhancements

A slightly modified form of the synchronization techniques described above can be employed. The basic principles of the previously described checkpoint synchronization scheme remain unchanged. The actions resulting from the reception of the checkpoints are, however, slightly different. In this variation, the receiver will maintain between OoO ("Out of Order") and 2×WINDOW_SIZE+OoO active addresses (1≦OoO≦WINDOW_SIZE and WINDOW_SIZE≧2). OoO and WINDOW_SIZE are engineerable parameters, where OoO is the minimum number of addresses needed to accommodate lost packets due to events in the network or out of order arrivals and WINDOW_SIZE is the number of packets transmitted before a SYNC_REQ is issued. FIG. **17** depicts a storage array for a receiver's active addresses.

The receiver starts with the first 2×WINDOW_SIZE addresses loaded and active (ready to receive data). As packets are received, the corresponding entries are marked as "used" and are no longer eligible to receive packets. The transmitter maintains a packet counter, initially set to 0, containing the number of data packets transmitted since the last initial transmission of a SYNC_REQ for which SYNC_ACK has been received. When the transmitter packet counter equals WINDOW_SIZE, the transmitter generates a SYNC_REQ and does its initial transmission. When the receiver receives a SYNC_REQ corresponding to its current CKPT_N, it generates the next WINDOW_SIZE addresses and starts loading them in order starting at the first location after the last active address wrapping around to the beginning of the array after the end of the array has been reached. The receiver's array might look like FIG. **18** when a SYNC_REQ has been received. In this case a couple of packets have been either lost or will be received out of order when the SYNC_REQ is received.

FIG. **19** shows the receiver's array after the new addresses have been generated. If the transmitter does not receive a SYNC_ACK, it will re-issue the SYNC_REQ at regular intervals. When the transmitter receives a SYNC_ACK, the packet counter is decremented by WINDOW_SIZE. If the packet counter reaches 2×WINDOW_SIZE–OoO then the transmitter ceases sending data packets until the appropriate SYNC_ACK is finally received. The transmitter then resumes sending data packets. Future behavior is essentially a repetition of this initial cycle. The advantages of this approach are:

1. There is no need for an efficient jump ahead in the random number generator,
2. No packet is ever transmitted that does not have a corresponding entry in the receiver side
3. No timer based re-synchronization is necessary. This is a consequence of 2.
4. The receiver will always have the ability to accept data messages transmitted within OoO messages of the most recently transmitted message.

J. Distributed Transmission Path Variant

Another embodiment incorporating various inventive principles is shown in FIG. **20**. In this embodiment, a message transmission system includes a first computer **2001** in communication with a second computer **2002** through a network **2011** of intermediary computers. In one variant of this embodiment, the network includes two edge routers **2003** and **2004** each of which is linked to a plurality of Internet Service Providers (ISPs) **2005** through **2010**. Each ISP is coupled to a plurality of other ISPs in an arrangement as shown in FIG. **20**, which is a representative configuration only and is not intended to be limiting. Each connection between ISPs is labeled in FIG. **20** to indicate a specific physical transmission path (e.g., AD is a physical path that links ISP A (element **2005**) to ISP D (element **2008**)). Packets arriving at each edge router are selectively transmitted to one of the ISPs to which the router is attached on the basis of a randomly or quasi-randomly selected basis.

As shown in FIG. **21**, computer **2001** or edge router **2003** incorporates a plurality of link transmission tables **2100** that identify, for each potential transmission path through the network, valid sets of IP addresses that can be used to transmit the packet. For example, AD table **2101** contains a plurality of IP source/destination pairs that are randomly or quasi-randomly generated. When a packet is to be transmitted from first computer **2001** to second computer **2002**, one of the link tables is randomly (or quasi-randomly) selected, and the next valid source/destination address pair from that table is used to transmit the packet through the network. If path AD is randomly selected, for example, the next source/destination IP address pair (which is pre-determined to transmit between ISP A (element **2005**) and ISP B (element **2008**)) is used to transmit the packet. If one of the transmission paths becomes degraded or inoperative, that link table can be set to a "down" condition as shown in table **2105**, thus preventing addresses from being selected from that table. Other transmission paths would be unaffected by this broken link.

3. Continuation-In-Part Improvements

The following describes various improvements and features that can be applied to the embodiments described above. The improvements include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities. Each is discussed separately below.

A. Load Balancer

Various embodiments described above include a system in which a transmitting node and a receiving node are coupled through a plurality of transmission paths, and wherein successive packets are distributed quasi-randomly over the plurality of paths. See, for example, FIGS. **20** and **21** and accompanying description. The improvement extends this basic concept to encompass distributing packets across different paths in such a manner that the loads on the paths are generally balanced according to transmission link quality.

In one embodiment, a system includes a transmitting node and a receiving node that are linked via a plurality of transmission paths having potentially varying transmission quality. Successive packets are transmitted over the paths based on a weight value distribution function for each path. The rate that packets will be transmitted over a given path can be different for each path. The relative "health" of each transmission path is monitored in order to identify paths that have become degraded. In one embodiment, the health of each path is monitored in the transmitter by comparing the number of packets transmitted to the number of packet acknowledgements received. Each transmission path may comprise a physically separate path (e.g., via dial-up phone line, computer network, router, bridge, or the like), or may comprise logically separate paths contained within a broadband communication medium (e.g., separate channels in an FDM, TDM, CDMA, or other type of modulated or unmodulated transmission link).

When the transmission quality of a path falls below a predetermined threshold and there are other paths that can transmit packets, the transmitter changes the weight value used for that path, making it less likely that a given packet will be transmitted over that path. The weight will preferably be set no lower than a minimum value that keeps nominal traffic on the path. The weights of the other available paths are altered to compensate for the change in the affected path. When the quality of a path degrades to where the transmitter is turned off by the synchronization function (i.e., no packets are arriving at the destination), the weight is set to zero. If all transmitters are turned off, no packets are sent.

Conventional TCP/IP protocols include a "throttling" feature that reduces the transmission rate of packets when it is determined that delays or errors are occurring in transmission. In this respect, timers are sometimes used to determine whether packets have been received. These conventional techniques for limiting transmission of packets, however, do not involve multiple transmission paths between two nodes wherein transmission across a particular path relative to the others is changed based on link quality.

According to certain embodiments, in order to damp oscillations that might otherwise occur if weight distributions are changed drastically (e.g., according to a step function), a linear or an exponential decay formula can be applied to gradually decrease the weight value over time that a degrading path will be used. Similarly, if the health of a degraded path improves, the weight value for that path is gradually increased.

Transmission link health can be evaluated by comparing the number of packets that are acknowledged within the transmission window (see embodiments discussed above) to the number of packets transmitted within that window and by the state of the transmitter (i.e., on or off). In other words, rather than accumulating general transmission statistics over time for a path, one specific implementation uses the "windowing" concepts described above to evaluate transmission path health.

The same scheme can be used to shift virtual circuit paths from an "unhealthy" path to a "healthy" one, and to select a path for a new virtual circuit.

FIG. **22A** shows a flowchart for adjusting weight values associated with a plurality of transmission links. It is assumed that software executing in one or more computer nodes executes the steps shown in FIG. **22A**. It is also assumed that the software can be stored on a computer-readable medium such as a magnetic or optical disk for execution by a computer.

Beginning in step **2201**, the transmission quality of a given transmission path is measured. As described above, this measurement can be based on a comparison between the number of packets transmitted over a particular link to the number of packet acknowledgements received over the link (e.g., per unit time, or in absolute terms). Alternatively, the quality can be evaluated by comparing the number of packets that are acknowledged within the transmission window to the number of packets that were transmitted within that window. In yet another variation, the number of missed synchronization messages can be used to indicate link quality. Many other variations are of course possible.

In step **2202**, a check is made to determine whether more than one transmitter (e.g., transmission path) is turned on. If not, the process is terminated and resumes at step **2201**.

In step **2203**, the link quality is compared to a given threshold (e.g., 50%, or any arbitrary number). If the quality falls below the threshold, then in step **2207** a check is made to determine whether the weight is above a minimum level (e.g., 1%). If not, then in step **2209** the weight is set to the minimum level and processing resumes at step **2201**. If the weight is above the minimum level, then in step **2208** the weight is gradually decreased for the path, then in step **2206** the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are increased).

If in step **2203** the quality of the path was greater than or equal to the threshold, then in step **2204** a check is made to determine whether the weight is less than a steady-state value for that path. If so, then in step **2205** the weight is increased toward the steady-state value, and in step **2206** the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are decreased). If in step **2204** the weight is not less than the steady-state value, then processing resumes at step **2201** without adjusting the weights.

The weights can be adjusted incrementally according to various functions, preferably by changing the value gradually. In one embodiment, a linearly decreasing function is used to adjust the weights; according to another embodiment, an exponential decay function is used. Gradually changing the weights helps to damp oscillators that might otherwise occur if the probabilities were abruptly.

Although not explicitly shown in FIG. **22A** the process can be performed only periodically (e.g., according to a time schedule), or it can be continuously run, such as in a background mode of operation. In one embodiment, the combined weights of all potential paths should add up to unity (e.g., when the weighting for one path is decreased, the corresponding weights that the other paths will be selected will increase).

Adjustments to weight values for other paths can be prorated. For example, a decrease of 10% in weight value for one path could result in an evenly distributed increase in the weights for the remaining paths. Alternatively, weightings could be adjusted according to a weighted formula as desired (e.g., favoring healthy paths over less healthy paths). In yet another variation, the difference in weight value can be amortized over the remaining links in a manner that is proportional to their traffic weighting.

FIG. **22B** shows steps that can be executed to shut down transmission links where a transmitter turns off. In step **2210**, a transmitter shut-down event occurs. In step **2211**, a test is made to determine whether at least one transmitter is still turned on. If not, then in step **2215** all packets are dropped until a transmitter turns on. If in step **2211** at least one transmitter is turned on, then in step **2212** the weight for the path is set to zero, and the weights for the remaining paths are adjusted accordingly.

FIG. **23** shows a computer node **2301** employing various principles of the above-described embodiments. It is assumed that two computer nodes of the type shown in FIG. **23** communicate over a plurality of separate physical transmission paths. As shown in FIG. **23**, four transmission paths X1 through X4 are defined for communicating between the two nodes. Each node includes a packet transmitter **2302** that operates in accordance with a transmit table **2308** as described above. (The packet transmitter could also operate without using the IP-hopping features described above, but the following description assumes that some form of hopping is employed in conjunction with the path selection mechanism.). The computer node also includes a packet receiver **2303** that operates in accordance with a receive table **2309**, including a moving window W that moves as valid packets are received. Invalid packets having source and destination addresses that do not fall within window W are rejected.

As each packet is readied for transmission, source and destination IP addresses (or other discriminator values) are selected from transmit table **2308** according to any of the various algorithms described above, and packets containing these source/destination address pairs, which correspond to the node to which the four transmission paths are linked, are generated to a transmission path switch **2307**. Switch **2307**, which can comprise a software function, selects from one of the available transmission paths according to a weight distribution table **2306**. For example, if the weight for path X1 is 0.2, then every fifth packet will be transmitted on path X1. A similar regime holds true for the other paths as shown. Initially, each link's weight value can be set such that it is proportional to its bandwidth, which will be referred to as its "steady-state" value.

Packet receiver **2303** generates an output to a link quality measurement function **2304** that operates as described above to determine the quality of each transmission path. (The input to packet receiver **2303** for receiving incoming packets is omitted for clarity). Link quality measurement function **2304** compares the link quality to a threshold for each transmission link and, if necessary, generates an output to weight adjustment function **2305**. If a weight adjustment is required, then the weights in table **2306** are adjusted accordingly, preferably according to a gradual (e.g., linearly or exponentially declining) function. In one embodiment, the weight values for all available paths are initially set to the same value, and only when paths degrade in quality are the weights changed to reflect differences.

Link quality measurement function **2304** can be made to operate as part of a synchronizer function as described above. That is, if resynchronization occurs and the receiver detects that synchronization has been lost (e.g., resulting in the synchronization window W being advanced out of sequence), that fact can be used to drive link quality measurement function **2304**. According to one embodiment, load balancing is performed using information garnered during the normal synchronization, augmented slightly to communicate link health from the receiver to the transmitter. The receiver maintains a count, MESS_R(W), of the messages received in synchronization window W. When it receives a synchronization request (SYNC_REQ) corresponding to the end of window W, the receiver includes counter MESS_R in the resulting synchronization acknowledgement (SYNC_ACK) sent back to the transmitter. This allows the transmitter to compare messages sent to messages received in order to asses the health of the link.

If synchronization is completely lost, weight adjustment function **2305** decreases the weight value on the affected path to zero. When synchronization is regained, the weight value for the affected path is gradually increased to its original value. Alternatively, link quality can be measured by evaluating the length of time required for the receiver to acknowledge a synchronization request. In one embodiment, separate transmit and receive tables are used for each transmission path.

When the transmitter receives a SYNC_ACK, the MESS_R is compared with the number of messages transmitted in a window (MESS_T). When the transmitter receives a SYNC_ACK, the traffic probabilities will be examined and adjusted if necessary. MESS_R is compared with the number of messages transmitted in a window (MESS_T). There are two possibilities:

1. If MESS_R is less than a threshold value, THRESH, then the link will be deemed to be unhealthy. If the transmitter was turned off, the transmitter is turned on and the weight P for that link will be set to a minimum value MIN. This will keep a trickle of traffic on the link for monitoring purposes until it recovers. If the transmitter was turned on, the weight P for that link will be set to:

$$P'=\alpha\times MIN+(1-\alpha)\times P \quad (1)$$

Equation 1 will exponentially damp the traffic weight value to MIN during sustained periods of degraded service.

2. If MESS_R for a link is greater than or equal to THRESH, the link will be deemed healthy. If the weight P for that link is greater than or equal to the steady state value S for that link, then P is left unaltered. If the weight P for that link is less than THRESH then P will be set to:

$$P'=\beta\times S+(1-\beta)\times P \quad (2)$$

where β is a parameter such that 0<=β<=1 that determines the damping rate of P.

Equation 2 will increase the traffic weight to S during sustained periods of acceptable service in a damped exponential fashion.

A detailed example will now be provided with reference to FIG. **24**. As shown in FIG. **24**, a first computer **2401** communicates with a second computer **2402** through two routers **2403** and **2404**. Each router is coupled to the other router through three transmission links. As described above, these may be physically diverse links or logical links (including virtual private networks).

Suppose that a first link L1 can sustain a transmission bandwidth of 100 Mb/s and has a window size of 32; link L2 can sustain 75 Mb/s and has a window size of 24; and link L3 can sustain 25 Mb/s and has a window size of 8. The combined links can thus sustain 200 Mb/s. The steady state traffic weights are 0.5 for link L1; 0.375 for link L2, and 0.125 for link L3. MIN=1 Mb/s, THRESH=0.8 MESS_T for each link, α=0.75 and β=0.5. These traffic weights will remain stable until a link stops for synchronization or reports a number of packets received less than its THRESH. Consider the following sequence of events:

1. Link L1 receives a SYNC_ACK containing a MESS_R of 24, indicating that only 75% of the MESS_T (32) messages transmitted in the last window were successfully received. Link 1 would be below THRESH (0.8). Consequently, link L1's traffic weight value would be reduced to 0.12825, while link L2's traffic weight value would be increased to 0.65812 and link L3's traffic weight value would be increased to 0.217938.

2. Link L2 and L3 remained healthy and link L1 stopped to synchronize. Then link L1's traffic weight value would be set to 0, link L2's traffic weight value would be set to 0.75, and link L33's traffic weight value would be set to 0.25.

3. Link L1 finally received a SYNC_ACK containing a MESS_R of 0 indicating that none of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be below THRESH. Link L1's traffic weight value would be increased to 0.005, link L2's traffic weight value would be decreased to 0.74625, and link L3's traffic weight value would be decreased to 0.24875.

4. Link L1 received a SYNC_ACK containing a MESS_R of 32 indicating that 100% of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.2525, while link L2's traffic weight value would be decreased to 0.560625 and link L3's traffic weight value would be decreased to 0.186875.

5. Link L1 received a SYNC_ACK containing a MESS_R of 32 indicating that 100% of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.37625; link L2's traffic weight value would be decreased to 0.4678125, and link L3's traffic weight value would be decreased to 0.1559375.

6. Link L1 remains healthy and the traffic probabilities approach their steady state traffic probabilities.

B. Use of a DNS Proxy to Transparently Create Virtual Private Networks

A second improvement concerns the automatic creation of a virtual private network (VPN) in response to a domain-name server look-up function.

Conventional Domain Name Servers (DNSs) provide a look-up function that returns the IP address of a requested computer or host. For example, when a computer user types in the web name "Yahoo.com," the user's web browser transmits a request to a DNS, which converts the name into a four-part IP address that is returned to the user's browser and then used by the browser to contact the destination web site.

This conventional scheme is shown in FIG. **25**. A user's computer **2501** includes a client application **2504** (for example, a web browser) and an IP protocol stack **2505**. When the user enters the name of a destination host, a request DNS REQ is made (through IP protocol stack **2505**) to a DNS **2502** to look up the IP address associated with the name. The DNS returns the IP address DNS RESP to client application **2504**, which is then able to use the IP address to communicate with the host **2503** through separate transactions such as PAGE REQ and PAGE RESP.

In the conventional architecture shown in FIG. **25**, nefarious listeners on the Internet could intercept the DNS REQ and DNS RESP packets and thus learn what IP addresses the user was contacting. For example, if a user wanted to set up a secure communication path with a web site having the name "Target.com," when the user's browser contacted a DNS to find the IP address for that web site, the true IP address of that web site would be revealed over the Internet as part of the DNS inquiry. This would hamper anonymous communications on the Internet.

One conventional scheme that provides secure virtual private networks over the Internet provides the DNS server with the public keys of the machines that the DNS server has the addresses for. This allows hosts to retrieve automatically the public keys of a host that the host is to communicate with so that the host can set up a VPN without having the user enter the public key of the destination host. One implementation of this standard is presently being developed as part of the FreeS/WAN project(RFC 2535).

The conventional scheme suffers from certain drawbacks. For example, any user can perform a DNS request. Moreover, DNS requests resolve to the same value for all users.

According to certain aspects of the invention, a specialized DNS server traps DNS requests and, if the request is from a special type of user (e.g., one for which secure communication services are defined), the server does not return the true IP address of the target node, but instead automatically sets up a virtual private network between the target node and the user. The VPN is preferably implemented using the IP address "hopping" features of the basic invention described above, such that the true identity of the two nodes cannot be determined even if packets during the communication are intercepted. For DNS requests that are determined to not require secure services (e.g., an unregistered user), the DNS server transparently "passes through" the request to provide a normal look-up function and return the IP address of the target web server, provided that the requesting host has permissions to resolve unsecured sites. Different users who make an identical DNS request could be provided with different results.

FIG. **26** shows a system employing various principles summarized above. A user's computer **2601** includes a conventional client (e.g., a web browser) **2605** and an IP protocol stack **2606** that preferably operates in accordance with an IP hopping function **2607** as outlined above. A modified DNS server **2602** includes a conventional DNS server function **2609** and a DNS proxy **2610**. A gatekeeper server **2603** is interposed between the modified DNS server and a secure target site **2704**. An "unsecure" target site **2611** is also accessible via conventional IP protocols.

According to one embodiment, DNS proxy **2610** intercepts all DNS lookup functions from client **2605** and determines whether access to a secure site has been requested. If access to a secure site has been requested (as determined, for example, by a domain name extension, or by reference to an internal table of such sites), DNS proxy **2610** determines whether the user has sufficient security privileges to access the site. If so, DNS proxy **2610** transmits a message to gatekeeper **2603** requesting that a virtual private network be created between user computer **2601** and secure target site **2604**. In one embodiment, gatekeeper **2603** creates "hopblocks" to be used by computer **2601** and secure target site **2604** for secure communication. Then, gatekeeper **2603** communicates these to user computer **2601**. Thereafter, DNS proxy **2610** returns to user computer **2601** the resolved address passed to it by the gatekeeper (this address could be different from the actual target computer) **2604**, preferably using a secure administrative VPN. The address that is returned need not be the actual address of the destination computer.

Had the user requested lookup of a non-secure web site such as site **2611**, DNS proxy would merely pass through to conventional DNS server **2609** the look-up request, which would be handled in a conventional manner, returning the IP address of non-secure web site **2611**. If the user had requested lookup of a secure web site but lacked credentials to create such a connection, DNS proxy **2610** would return a "host unknown" error to the user. In this manner, different users requesting access to the same DNS name could be provided with different look-up results.

Gatekeeper **2603** can be implemented on a separate computer (as shown in FIG. **26**) or as a function within modified DNS server **2602**. In general, it is anticipated that gatekeeper **2703** facilitates the allocation and exchange of information needed to communicate securely, such as using "hopped" IP addresses. Secure hosts such as site **2604** are assumed to be equipped with a secure communication function such as an IP hopping function **2608**.

It will be appreciated that the functions of DNS proxy **2610** and DNS server **2609** can be combined into a single server for convenience. Moreover, although element **2602** is shown as combining the functions of two servers, the two servers can be made to operate independently.

FIG. **27** shows steps that can be executed by DNS proxy server **2610** to handle requests for DNS look-up for secure hosts. In step **2701**, a DNS look-up request is received for a target host. In step **2702**, a check is made to determine whether access to a secure host was requested. If not, then in step **2703** the DNS request is passed to conventional DNS server **2609**, which looks up the IP address of the target site and returns it to the user's application for further processing.

In step **2702**, if access to a secure host was requested, then in step **2704** a further check is made to determine whether the user is authorized to connect to the secure host. Such a check can be made with reference to an internally stored list of authorized IP addresses, or can be made by communicating with gatekeeper **2603** (e.g., over an "administrative" VPN that is secure). It will be appreciated that different levels of security can also be provided for different categories of hosts. For example, some sites may be designated as having a certain security level, and the security level of the user requesting access must match that security level. The user's security level can also be determined by transmitting a request message back to the user's computer requiring that it prove that it has sufficient privileges.

If the user is not authorized to access the secure site, then a "host unknown" message is returned (step **2705**). If the user has sufficient security privileges, then in step **2706** a secure VPN is established between the user's computer and the secure target site. As described above, this is preferably done by allocating a hopping regime that will be carried out between the user's computer and the secure target site, and is preferably performed transparently to the user (i.e., the user need not be involved in creating the secure link). As described in various embodiments of this application, any of various fields can be "hopped" (e.g., IP source/destination addresses; a field in the header; etc.) in order to communicate securely.

Some or all of the security functions can be embedded in gatekeeper **2603**, such that it handles all requests to connect to secure sites. In this embodiment, DNS proxy **2610** communicates with gatekeeper **2603** to determine (preferably over a secure administrative VPN) whether the user has access to a particular web site. Various scenarios for implementing these features are described by way of example below:

Scenario #1: Client has permission to access target computer, and gatekeeper has a rule to make a VPN for the client. In this scenario, the client's DNS request would be received by the DNS proxy server **2610**, which would forward the request to gatekeeper **2603**. The gatekeeper would establish a VPN between the client and the requested target. The gatekeeper would provide the address of the destination to the DNS proxy, which would then return the resolved name as a result. The resolved address can be transmitted back to the client in a secure administrative VPN.

Scenario #2: Client does not have permission to access target computer. In this scenario, the client's DNS request would be received by the DNS proxy server **2610**, which would forward the request to gatekeeper **2603**. The gatekeeper would reject the request, informing DNS proxy server **2610** that it was unable to find the target computer. The DNS proxy **2610** would then return a "host unknown" error message to the client.

Scenario #3: Client has permission to connect using a normal non-VPN link, and the gatekeeper does not have a rule to set up a VPN for the client to the target site. In this scenario, the client's DNS request is received by DNS proxy server **2610**, which would check its rules and determine that no VPN is needed. Gatekeeper **2603** would then inform the DNS proxy server to forward the request to conventional DNS server **2609**, which would resolve the request and return the result to the DNS proxy server and then back to the client.

Scenario #4: Client does not have permission to establish a normal/non-VPN link, and the gatekeeper does not have a rule to make a VPN for the client to the target site. In this scenario, the DNS proxy server would receive the client's DNS request and forward it to gatekeeper **2603**. Gatekeeper **2603** would determine that no special VPN was needed, but that the client is not authorized to communicate with non-VPN members. The gatekeeper would reject the request, causing DNS proxy server **2610** to return an error message to the client.

C. Large Link to Small Link Bandwidth Management

One feature of the basic architecture is the ability to prevent so-called "denial of service" attacks that can occur if a computer hacker floods a known Internet node with packets, thus preventing the node from communicating with other nodes.

Because IP addresses or other fields are "hopped" and packets arriving with invalid addresses are quickly discarded, Internet nodes are protected against flooding targeted at a single IP address.

In a system in which a computer is coupled through a link having a limited bandwidth (e.g., an edge router) to a node that can support a much higher-bandwidth link (e.g., an Internet Service Provider), a potential weakness could be exploited by a determined hacker. Referring to FIG. **28**, suppose that a first host computer **2801** is communicating with a second host computer **2804** using the IP address hopping principles described above. The first host computer is coupled through an edge router **2802** to an Internet Service Provider (ISP) **2803** through a low bandwidth link (LOW BW), and is in turn coupled to second host computer **2804** through parts of the Internet through a high bandwidth link (HIGH BW). In this architecture, the ISP is able to support a high bandwidth to the internet, but a much lower bandwidth to the edge router **2802**.

Suppose that a computer hacker is able to transmit a large quantity of dummy packets addressed to first host computer **2801** across high bandwidth link HIGH BW. Normally, host computer **2801** would be able to quickly reject the packets since they would not fall within the acceptance window permitted by the IP address hopping scheme. However, because the packets must travel across low bandwidth link LOW BW, the packets overwhelm the lower bandwidth link before they are received by host computer **2801**. Consequently, the link to host computer **2801** is effectively flooded before the packets can be discarded.

According to one inventive improvement, a "link guard" function **2805** is inserted into the high-bandwidth node (e.g., ISP **2803**) that quickly discards packets destined for a low-bandwidth target node if they are not valid packets. Each packet destined for a low-bandwidth node is cryptographically authenticated to determine whether it belongs to a VPN. If it is not a valid VPN packet, the packet is discarded at the high-bandwidth node. If the packet is authenticated as belonging to a VPN, the packet is passed with high preference. If the packet is a valid non-VPN packet, it is passed with a lower quality of service (e.g., lower priority).

In one embodiment, the ISP distinguishes between VPN and non-VPN packets using the protocol of the packet. In the case of IPSEC [rfc **2401**], the packets have IP protocols **420** and **421**. In the case of the TARP VPN, the packets will have an IP protocol that is not yet defined. The ISP's link guard, **2805**, maintains a table of valid VPNs which it uses to validate whether VPN packets are cryptographically valid. According to one embodiment, packets that do not fall within any hop windows used by nodes on the low-bandwidth link are rejected, or are sent with a lower quality of service. One approach for doing this is to provide a copy of the IP hopping tables used by the low-bandwidth nodes to the high-bandwidth node, such that both the high-bandwidth and low-bandwidth nodes track hopped packets (e.g., the high-bandwidth node moves its hopping window as valid packets are received). In such a scenario, the high-bandwidth node discards packets that do not fall within the hopping window before they are transmitted over the low-bandwidth link. Thus, for example, ISP **2903** maintains a copy **2910** of the receive table used by host computer **2901**. Incoming packets that do not fall within this receive table are discarded. According to a different embodiment, link guard **2805** validates each VPN packet using a keyed hashed message authentication code (HMAC) [rfc **2104**].

According to another embodiment, separate VPNs (using, for example, hopblocks) can be established for communicating between the low-bandwidth node and the high-bandwidth node (i.e., packets arriving at the high-bandwidth node are converted into different packets before being transmitted to the low-bandwidth node).

As shown in FIG. 29, for example, suppose that a first host computer 2900 is communicating with a second host computer 2902 over the Internet, and the path includes a high bandwidth link HIGH BW to an ISP 2901 and a low bandwidth link LOW BW through an edge router 2904. In accordance with the basic architecture described above, first host computer 2900 and second host computer 2902 would exchange hopblocks (or a hopblock algorithm) and would be able to create matching transmit and receive tables 2905, 2906, 2912 and 2913. Then in accordance with the basic architecture, the two computers would transmit packets having seemingly random IP source and destination addresses, and each would move a corresponding hopping window in its receive table as valid packets were received.

Suppose that a nefarious computer hacker 2903 was able to deduce that packets having a certain range of IP addresses (e.g., addresses 100 to 200 for the sake of simplicity) are being transmitted to ISP 2901, and that these packets are being forwarded over a low-bandwidth link. Hacker computer 2903 could thus "flood" packets having addresses falling into the range 100 to 200, expecting that they would be forwarded along low bandwidth link LOW BW, thus causing the low bandwidth link to become overwhelmed. The fast packet reject mechanism in first host computer 3000 would be of little use in rejecting these packets, since the low bandwidth link was effectively jammed before the packets could be rejected. In accordance with one aspect of the improvement, however, VPN link guard 2911 would prevent the attack from impacting the performance of VPN traffic because the packets would either be rejected as invalid VPN packets or given a lower quality of service than VPN traffic over the lower bandwidth link. A denial-of-service flood attack could, however, still disrupt non-VPN traffic.

According to one embodiment of the improvement, ISP 2901 maintains a separate VPN with first host computer 2900, and thus translates packets arriving at the ISP into packets having a different IP header before they are transmitted to host computer 2900. The cryptographic keys used to authenticate VPN packets at the link guard 2911 and the cryptographic keys used to encrypt and decrypt the VPN packets at host 2902 and host 2901 can be different, so that link guard 2911 does not have access to the private host data; it only has the capability to authenticate those packets.

According to yet a third embodiment, the low-bandwidth node can transmit a special message to the high-bandwidth node instructing it to shut down all transmissions on a particular IP address, such that only hopped packets will pass through to the low-bandwidth node. This embodiment would prevent a hacker from flooding packets using a single IP address. According to yet a fourth embodiment, the high-bandwidth node can be configured to discard packets transmitted to the low-bandwidth node if the transmission rate exceeds a certain predetermined threshold for any given IP address; this would allow hopped packets to go through. In this respect, link guard 2911 can be used to detect that the rate of packets on a given IP address are exceeding a threshold rate; further packets addressed to that same IP address would be dropped or transmitted at a lower priority (e.g., delayed).

D. Traffic Limiter

In a system in which multiple nodes are communicating using "hopping" technology, a treasonous insider could internally flood the system with packets. In order to prevent this possibility, one inventive improvement involves setting up "contracts" between nodes in the system, such that a receiver can impose a bandwidth limitation on each packet sender. One technique for doing this is to delay acceptance of a checkpoint synchronization request from a sender until a certain time period (e.g., one minute) has elapsed. Each receiver can effectively control the rate at which its hopping window moves by delaying "SYNC ACK" responses to "SYNC_REQ" messages.

A simple modification to the checkpoint synchronizer will serve to protect a receiver from accidental or deliberate overload from an internally treasonous client. This modification is based on the observation that a receiver will not update its tables until a SYNC_REQ is received on hopped address CKPT_N. It is a simple matter of deferring the generation of a new CKPT_N until an appropriate interval after previous checkpoints.

Suppose a receiver wished to restrict reception from a transmitter to 100 packets a second, and that checkpoint synchronization messages were triggered every 50 packets. A compliant transmitter would not issue new SYNC_REQ messages more often than every 0.5 seconds. The receiver could delay a non-compliant transmitter from synchronizing by delaying the issuance of CKPT_N for 0.5 second after the last SYNC_REQ was accepted.

In general, if M receivers need to restrict N transmitters issuing new SYNC_REQ messages after every W messages to sending R messages a second in aggregate, each receiver could defer issuing a new CKPT_N until M×N×W/R seconds have elapsed since the last SYNC_REQ has been received and accepted. If the transmitter exceeds this rate between a pair of checkpoints, it will issue the new checkpoint before the receiver is ready to receive it, and the SYNC_REQ will be discarded by the receiver. After this, the transmitter will reissue the SYNC_REQ every T1 seconds until it receives a SYNC_ACK. The receiver will eventually update CKPT_N and the SYNC_REQ will be acknowledged. If the transmission rate greatly exceeds the allowed rate, the transmitter will stop until it is compliant. If the transmitter exceeds the allowed rate by a little, it will eventually stop after several rounds of delayed synchronization until it is in compliance. Hacking the transmitter's code to not shut off only permits the transmitter to lose the acceptance window. In this case it can recover the window and proceed only after it is compliant again.

Two practical issues should be considered when implementing the above scheme:

1. The receiver rate should be slightly higher than the permitted rate in order to allow for statistical fluctuations in traffic arrival times and non-uniform load balancing.

2. Since a transmitter will rightfully continue to transmit for a period after a SYNC_REQ is transmitted, the algorithm above can artificially reduce the transmitter's bandwidth. If events prevent a compliant transmitter from synchronizing for a period (e.g. the network dropping a SYNC_REQ or a SYNC_ACK) a SYNC_REQ will be accepted later than expected. After this, the transmitter will transmit fewer than expected messages before encountering the next checkpoint. The new checkpoint will not have been activated and the transmitter will have to retransmit the SYNC_REQ. This will appear to the receiver as if the transmitter is not compliant. Therefore, the next checkpoint will be accepted late from the transmitter's perspective. This has the effect of reducing the transmitter's allowed packet rate until the transmitter transmits at a packet rate below the agreed upon rate for a period of time.

To guard against this, the receiver should keep track of the times that the last C SYNC_REQs were received and accepted and use the minimum of M×N×W/R seconds after the last SYNC_REQ has been received and accepted, 2×M×N×W/R seconds after next to the last SYNC_REQ has been received and accepted, C×M×N×W/R seconds after $(C-1)^{th}$ to the last SYNC_REQ has been received, as the time to activate CKPT_N. This prevents the receiver from inappropriately limiting the transmitter's packet rate if at least one out of the last C SYNC_REQs was processed on the first attempt.

FIG. **30** shows a system employing the above-described principles. In FIG. **30**, two computers **3000** and **3001** are assumed to be communicating over a network N in accordance with the "hopping" principles described above (e.g., hopped IP addresses, discriminator values, etc.). For the sake of simplicity, computer **3000** will be referred to as the receiving computer and computer **3001** will be referred to as the transmitting computer, although full duplex operation is of course contemplated. Moreover, although only a single transmitter is shown, multiple transmitters can transmit to receiver **3000**.

As described above, receiving computer **3000** maintains a receive table **3002** including a window W that defines valid IP address pairs that will be accepted when appearing in incoming data packets. Transmitting computer **3001** maintains a transmit table **3003** from which the next IP address pairs will be selected when transmitting a packet to receiving computer **3000**. (For the sake of illustration, window W is also illustrated with reference to transmit table **3003**). As transmitting computer moves through its table, it will eventually generate a SYNC_REQ message as illustrated in function **3010**. This is a request to receiver **3000** to synchronize the receive table **3002**, from which transmitter **3001** expects a response in the form of a CKPT_N (included as part of a SYNC_ACK message). If transmitting computer **3001** transmits more messages than its allotment, it will prematurely generate the SYNC_REQ message. (If it has been altered to remove the SYNC_REQ message generation altogether, it will fall out of synchronization since receiver **3000** will quickly reject packets that fall outside of window W, and the extra packets generated by transmitter **3001** will be discarded).

In accordance with the improvements described above, receiving computer **3000** performs certain steps when a SYNC_REQ message is received, as illustrated in FIG. **30**. In step **3004**, receiving computer **3000** receives the SYNC_REQ message. In step **3005**, a check is made to determine whether the request is a duplicate. If so, it is discarded in step **3006**. In step **3007**, a check is made to determine whether the SYNC_REQ received from transmitter **3001** was received at a rate that exceeds the allowable rate R (i.e., the period between the time of the last SYNC_REQ message). The value R can be a constant, or it can be made to fluctuate as desired. If the rate exceeds R, then in step **3008** the next activation of the next CKPT_N hopping table entry is delayed by W/R seconds after the last SYNC_REQ has been accepted.

Otherwise, if the rate has not been exceeded, then in step **3109** the next CKPT_N value is calculated and inserted into the receiver's hopping table prior to the next SYNC_REQ from thetransmitter **3101**. Transmitter **3101** then processes the SYNC_REQ in the normal manner.

E. Signaling Synchronizer

In a system in which a large number of users communicate with a central node using secure hopping technology, a large amount of memory must be set aside for hopping tables and their supporting data structures. For example, if one million subscribers to a web site occasionally communicate with the web site, the site must maintain one million hopping tables, thus using up valuable computer resources, even though only a small percentage of the users may actually be using the system at any one time. A desirable solution would be a system that permits a certain maximum number of simultaneous links to be maintained, but which would "recognize" millions of registered users at any one time. In other words, out of a population of a million registered users, a few thousand at a time could simultaneously communicate with a central server, without requiring that the server maintain one million hopping tables of appreciable size.

One solution is to partition the central node into two nodes: a signaling server that performs session initiation for user log-on and log-off (and requires only minimally sized tables), and a transport server that contains larger hopping tables for the users. The signaling server listens for the millions of known users and performs a fast-packet reject of other (bogus) packets. When a packet is received from a known user, the signaling server activates a virtual private link (VPL) between the user and the transport server, where hopping tables are allocated and maintained. When the user logs onto the signaling server, the user's computer is provided with hop tables for communicating with the transport server, thus activating the VPL. The VPLs can be torn down when they become inactive for a time period, or they can be torn down upon user log-out. Communication with the signaling server to allow user log-on and log-off can be accomplished using a specialized version of the checkpoint scheme described above.

FIG. **31** shows a system employing certain of the above-described principles. In FIG. **31**, a signaling server **3101** and a transport server **3102** communicate over a link. Signaling server **3101** contains a large number of small tables **3106** and **3107** that contain enough information to authenticate a communication request with one or more clients **3103** and **3104**. As described in more detail below, these small tables may advantageously be constructed as a special case of the synchronizing checkpoint tables described previously. Transport server **3102**, which is preferably a separate computer in communication with signaling server **3101**, contains a smaller number of larger hopping tables **3108**, **3109**, and **3110** that can be allocated to create a VPN with one of the client computers.

According to one embodiment, a client that has previously registered with the system (e.g., via a system administration function, a user registration procedure, or some other method) transmits a request for information from a computer (e.g., a web site). In one variation, the request is made using a "hopped" packet, such that signaling server **3101** will quickly reject invalid packets from unauthorized computers such as hacker computer **3105**. An "administrative" VPN can be established between all of the clients and the signaling server in order to ensure that a hacker cannot flood signaling server **3101** with bogus packets. Details of this scheme are provided below.

Signaling server **3101** receives the request **3111** and uses it to determine that client **3103** is a validly registered user. Next, signaling server **3101** issues a request to transport server **3102** to allocate a hopping table (or hopping algorithm or other regime) for the purpose of creating a VPN with client **3103**. The allocated hopping parameters are returned to signaling server **3101** (path **3113**), which then supplies the hopping parameters to client **3103** via path **3114**, preferably in encrypted form.

Thereafter, client **3103** communicates with transport server **3102** using the normal hopping techniques described above. It will be appreciated that although signaling server **3101** and transport server **3102** are illustrated as being two separate computers, they could of course be combined into a single computer and their functions performed on the single computer. Alternatively, it is possible to partition the functions shown in FIG. **31** differently from as shown without departing from the inventive principles.

One advantage of the above-described architecture is that signaling server **3101** need only maintain a small amount of information on a large number of potential users, yet it retains the capability of quickly rejecting packets from unauthorized users such as hacker computer **3105**. Larger data tables needed to perform the hopping and synchronization functions are instead maintained in a transport server **3102**, and a smaller number of these tables are needed since they are only allocated for "active" links. After a VPN has become inactive for a certain time period (e.g., one hour), the VPN can be automatically torn down by transport server **3102** or signaling server **3101**.

A more detailed description will now be provided regarding how a special case of the checkpoint synchronization feature can be used to implement the signaling scheme described above.

The signaling synchronizer may be required to support many (millions) of standing, low bandwidth connections. It therefore should minimize per-VPL memory usage while providing the security offered by hopping technology. In order to reduce memory usage in the signaling server, the data hopping tables can be completely eliminated and data can be carried as part of the SYNC_REQ message. The table used by the server side (receiver) and client side (transmitter) is shown schematically as element **3106** in FIG. **31**.

The meaning and behaviors of CKPT_N, CKPT_O and CKPT_R remain the same from the previous description, except that CKPT_N can receive a combined data and SYNC_REQ message or a SYNC_REQ message without the data.

The protocol is a straightforward extension of the earlier synchronizer. Assume that a client transmitter is on and the tables are synchronized. The initial tables can be generated "out of band." For example, a client can log into a web server to establish an account over the Internet. The client will receive keys etc encrypted over the Internet. Meanwhile, the server will set up the signaling VPN on the signaling server.

Assuming that a client application wishes to send a packet to the server on the client's standing signaling VPL:

1. The client sends the message marked as a data message on the inner header using the transmitter's CKPT_N address. It turns the transmitter off and starts a timer TI noting CKPT_O. Messages can be one of three types: DATA, SYNC_REQ and SYNC_ACK. In the normal algorithm, some potential problems can be prevented by identifying each message type as part of the encrypted inner header field. In this algorithm, it is important to distinguish a data packet and a SYNC_REQ in the signaling synchronizer since the data and the SYNC_REQ come in on the same address.

2. When the server receives a data message on its CKPT_N, it verifies the message and passes it up the stack. The message can be verified by checking message type and and other information (i.e., user credentials) contained in the inner header It replaces its CKPT_O with CKPT_N and generates the next CKPT_N. It updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.

3. When the client side receiver receives a SYNC_ACK on its CKPT_R with a payload matching its transmitter side CKPT_O and the transmitter is off, the transmitter is turned on and the receiver side CKPT_R is updated. If the SYNC_ACK's payload does not match the transmitter side CKPT_O or the transmitter is on, the SYNC_ACK is simply discarded.

4. TI expires: If the transmitter is off and the client's transmitter side CKPT_O matches the CKPT_O associated with the timer, it starts timer Ti noting CKPT_O again, and a SYNC_REQ is sent using the transmitter's CKPT_O address. Otherwise, no action is taken.

5. When the server receives a SYNC_REQ on its CKPT_N, it replaces its CKPT_O with CKPT_N and generates the next CKPT_N. It updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.

6. When the server receives a SYNC_REQ on its CKPT_O, it updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.

FIG. **32** shows message flows to highlight the protocol. Reading from top to bottom, the client sends data to the server using its transmitter side CKPT_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT_N into CKPT_O and updates CKPT_N. This message is successfully received and a passed up the stack. It also synchronizes the receiver i.e., the server loads CKPT_N into CKPT_O and generates a new CKPT_N, it generates a new CKPT_R in the server side transmitter and transmits a SYNC_ACK containing the server side receiver's CKPT_O the server. The SYNC_ACK is successfully received at the client. The client side receiver's CKPT_R is updated, the transmitter is turned on and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

Next, the client sends data to the server using its transmitter side CKPT_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT_N into CKPT_O and updates CKPT_N. This message is lost. The client side timer expires and as a result a SYNC_REQ is transmitted on the client side transmitter's CKPT_O (this will keep happening until the SYNC_ACK has been received at the client). The SYNC_REQ is successfully received at the server. It synchronizes the receiver i.e., the server loads CKPT_N into CKPT_O and generates a new CKPT_N, it generates an new CKPT_R in the server side transmitter and transmits a SYNC_ACK containing the server side receiver's CKPT_O the server. The SYNC_ACK is successfully received at the client. The client side receiver's CKPT_R is updated, the transmitter is turned off and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

There are numerous other scenarios that follow this flow. For example, the SYNC_ACK could be lost. The transmitter would continue to re-send the SYNC_REQ until the receiver synchronizes and responds.

The above-described procedures allow a client to be authenticated at signaling server **3201** while maintaining the ability of signaling server **3201** to quickly reject invalid packets, such as might be generated by hacker computer **3205**. In various embodiments, the signaling synchronizer is really a derivative of the synchronizer. It provides the same protection as the hopping protocol, and it does so for a large number of low bandwidth connections.

F. One-click Secure On-line Communications and Secure Domain Name Service

The present invention provides a technique for establishing a secure communication link between a first computer and a second computer over a computer network. Preferably, a user enables a secure communication link using a single click of a mouse, or a corresponding minimal input from another input device, such as a keystroke entered on a keyboard or a click entered through a trackball. Alternatively, the secure link is automatically established as a default setting at boot-up of the computer (i.e., no click). FIG. **33** shows a system block diagram **3300** of a computer network in which the one-click secure communication method of the present invention is suitable. In FIG. **33**, a computer terminal or client computer **3301**, such as a personal computer (PC), is connected to a computer network **3302**, such as the Internet, through an ISP **3303**. Alternatively, computer **3301** can be connected to computer network **3302** through an edge router. Computer **3301** includes an input device, such as a keyboard and/or mouse, and a display device, such as a monitor. Computer **3301** can communicate conventionally with another computer **3304** connected to computer network **3302** over a communication link **3305** using a browser **3306** that is installed and operates on computer **3301** in a well-known manner.

Computer **3304** can be, for example, a server computer that is used for conducting e-commerce. In the situation when computer network **3302** is the Internet, computer **3304** typically will have a standard top-level domain name such as .com, .net, .org, .edu, .mil or .gov.

FIG. **34** shows a flow diagram **3400** for installing and establishing a "one-click" secure communication link over a computer network according to the present invention. At step **3401**, computer **3301** is connected to server computer **3304** over a non-VPN communication link **3305**. Web browser **3306** displays a web page associated with server **3304** in a well-known manner. According to one variation of the invention, the display of computer **3301** contains a hyperlink, or an icon representing a hyperlink, for selecting a virtual private network (VPN) communication link ("go secure" hyperlink) through computer network **3302** between terminal **3301** and server **3304**. Preferably, the "go secure" hyperlink is displayed as part of the web page downloaded from server computer **3304**, thereby indicating that the entity providing server **3304** also provides VPN capability.

By displaying the "go secure" hyperlink, a user at computer **3301** is informed that the current communication link between computer **3301** and server computer **3304** is a non-secure, non-VPN communication link. At step **3402**, it is determined whether a user of computer **3301** has selected the "go secure" hyperlink. If not, processing resumes using a non-secure (conventional) communication method (not shown). If, at step **3402**, it is determined that the user has selected the "go secure" hyperlink, flow continues to step **3403** where an object associated with the hyperlink determines whether a VPN communication software module has already been installed on computer **3301**. Alternatively, a user can enter a command into computer **3301** to "go secure."

If, at step **3403**, the object determines that the software module has been installed, flow continues to step **3407**. If, at step **3403**, the object determines that the software module has not been installed, flow continues to step **3404** where a non-VPN communication link **3307** is launched between computer **3301** and a website **3308** over computer network **3302** in a well-known manner. Website **3308** is accessible by all computer terminals connected to computer network **3302** through a non-VPN communication link. Once connected to website **3308**, a software module for establishing a secure communication link over computer network **3302** can be downloaded and installed. Flow continues to step **3405** where, after computer **3301** connects to website **3308**, the software module for establishing a communication link is downloaded and installed in a well-known manner on computer terminal **3301** as software module **3309**. At step **3405**, a user can optionally select parameters for the software module, such as enabling a secure communication link mode of communication for all communication links over computer network **3302**. At step **3406**, the communication link between computer **3301** and website **3308** is then terminated in a well-known manner.

By clicking on the "go secure" hyperlink, a user at computer **3301** has enabled a secure communication mode of communication between computer **3301** and server computer **3304**. According to one variation of the invention, the user is not required to do anything more than merely click the "go secure" hyperlink. The user does not need to enter any user identification information, passwords or encryption keys for establishing a secure communication link. All procedures required for establishing a secure communication link between computer **3301** and server computer **3304** are performed transparently to a user at computer **3301**.

At step **3407**, a secure VPN communications mode of operation has been enabled and software module **3309** begins to establish a VPN communication link. In one embodiment, software module **3309** automatically replaces the top-level domain name for server **3304** within browser **3406** with a secure top-level domain name for server computer **3304**. For example, if the top-level domain name for server **3304** is .com, software module **3309** replaces the .com top-level domain name with a scom top-level domain name, where the "s" stands for secure. Alternatively, software module **3409** can replace the top-level domain name of server **3304** with any other non-standard top-level domain name.

Because the secure top-level domain name is a non-standard domain name, a query to a standard domain name service (DNS) will return a message indicating that the universal resource locator (URL) is unknown. According to the invention, software module **3309** contains the URL for querying a secure domain name service (SDNS) for obtaining the URL for a secure top-level domain name. In this regard, software module **3309** accesses a secure portal **3310** that interfaces a secure network **3311** to computer network **3302**. Secure network **3311** includes an internal router **3312**, a secure domain name service (SDNS) **3313**, a VPN gatekeeper **3314** and a secure proxy **3315**. The secure network can include other network services, such as e-mail **3316**, a plurality of chatrooms (of which only one chatroom **3317** is shown), and a standard domain name service (STD DNS) **3318**. Of course, secure network **3311** can include other resources and services that are not shown in FIG. **33**.

When software module **3309** replaces the standard top-level domain name for server **3304** with the secure top-level domain name, software module **3309** sends a query to SDNS **3313** at step **3408** through secure portal **3310** preferably using an administrative VPN communication link **3319**. In this configuration, secure portal **3310** can only be accessed using a VPN communication link. Preferably, such a VPN communication link can be based on a technique of inserting a source and destination IP address pair into each data packet that is selected according to a pseudo-random sequence; an IP address hopping regime that pseudorandomly changes IP addresses in packets transmitted between a client computer and a secure target computer; periodically changing at least one field in a series of data packets according to a known sequence; an Internet Protocol (IP) address in a header of each data packet that is compared to a table of valid IP addresses maintained in a table in the second computer; and/or a comparison of the IP address in the header of each data packet to a moving window of valid IP addresses, and rejecting data packets having IP addresses that do not fall within the moving window. Other types of VPNs can alternatively be used. Secure portal **3310** authenticates the query from software module **3309** based on the particular information hopping technique used for VPN communication link **3319**.

SDNS **3313** contains a cross-reference database of secure domain names and corresponding secure network addresses. That is, for each secure domain name, SDNS **3313** stores a computer network address corresponding to the secure domain name. An entity can register a secure domain name in SDNS **3313** so that a user who desires a secure communication link to the website of the entity can automatically obtain the secure computer network address for the secure website. Moreover, an entity can register several secure domain names, with each respective secure domain name representing a different priority level of access in a hierarchy of access levels to a secure website. For example, a securities trading website can provide users secure access so that a denial of service attack on the website will be ineffectual with respect to users subscribing to the secure website service. Different levels of subscription can be arranged based on, for example, an escalating fee, so that a user can select a desired level of guarantee for connecting to the secure securities trading website. When a user queries SDNS **3313** for the secure computer network address for the securities trading website, SDNS **3313** determines the particular secure computer network address based on the user's identity and the user's subscription level.

At step **3409**, SDNS **3313** accesses VPN gatekeeper **3314** for establishing a VPN communication link between software module **3309** and secure server **3320**. Server **3320** can only be accessed through a VPN communication link. VPN gatekeeper **3314** provisions computer **3301** and secure web server computer **3320**, or a secure edge router for server computer **3320**, thereby creating the VPN. Secure server computer **3320** can be a separate server computer from server computer **3304**, or can be the same server computer having both non-VPN and VPN communication link capability, such as shown by server computer **3322**. Returning to FIG. **34**, in step **3410**, SDNS **3313** returns a secure URL to software module **3309** for the .scom server address for a secure server **3320** corresponding to server **3304**.

Alternatively, SDNS **3313** can be accessed through secure portal **3310** "in the clear", that is, without using an administrative VPN communication link. In this situation, secure portal **3310** preferably authenticates the query using any well-known technique, such as a cryptographic technique, before allowing the query to proceed to SDNS **3319**. Because the initial communication link in this situation is not a VPN communication link, the reply to the query can be "in the clear." The querying computer can use the clear reply for establishing a VPN link to the desired domain name. Alternatively, the query to SDNS **3313** can be in the clear, and SDNS **3313** and gatekeeper **3314** can operate to establish a VPN communication link to the querying computer for sending the reply.

At step **3411**, software module **3309** accesses secure server **3320** through VPN communication link **3321** based on the VPN resources allocated by VPN gatekeeper **3314**. At step **3412**, web browser **3306** displays a secure icon indicating that the current communication link to server **3320** is a secure VPN communication link. Further communication between

computers **3301** and **3320** occurs via the VPN, e.g., using a "hopping" regime as discussed above. When VPN link **3321** is terminated at step **3413**, flow continues to step **3414** where software module **3309** automatically replaces the secure top-level domain name with the corresponding non-secure top-level domain name for server **3304**. Browser **3306** accesses a standard DNS **3325** for obtaining the non-secure URL for server **3304**. Browser **3306** then connects to server **3304** in a well-known manner. At step **3415**, browser **3306** displays the "go secure" hyperlink or icon for selecting a VPN communication link between terminal **3301** and server **3304**. By again displaying the "go secure" hyperlink, a user is informed that the current communication link is a non-secure, non-VPN communication link.

When software module **3309** is being installed or when the user is off-line, the user can optionally specify that all communication links established over computer network **3302** are secure communication links. Thus, anytime that a communication link is established, the link is a VPN link. Consequently, software module **3309** transparently accesses SDNS **3313** for obtaining the URL for a selected secure website. In other words, in one embodiment, the user need not "click" on the secure option each time secure communication is to be effected.

Additionally, a user at computer **3301** can optionally select a secure communication link through proxy computer **3315**. Accordingly, computer **3301** can establish a VPN communication link **3323** with secure server computer **3320** through proxy computer **3315**. Alternatively, computer **3301** can establish a non-VPN communication link **3324** to a non-secure website, such as non-secure server computer **3304**.

FIG. **35** shows a flow diagram **3500** for registering a secure domain name according to the present invention. At step **3501**, a requester accesses website **3308** and logs into a secure domain name registry service that is available through website **3308**. At step **3502**, the requestor completes an online registration form for registering a secure domain name having a top-level domain name, such as .com, .net, .org, .edu, .mil or .gov. Of course, other secure top-level domain names can also be used. Preferably, the requestor must have previously registered a non-secure domain name corresponding to the equivalent secure domain name that is being requested. For example, a requestor attempting to register secure domain name "website.scom" must have previously registered the corresponding non-secure domain name "website.com".

At step **3503**, the secure domain name registry service at website **3308** queries a non-secure domain name server database, such as standard DNS **3322**, using, for example, a whois query, for determining ownership information relating to the non-secure domain name corresponding to the requested secure domain name. At step **3504**, the secure domain name registry service at website **3308** receives a reply from standard DNS **3322** and at step **3505** determines whether there is conflicting ownership information for the corresponding non-secure domain name. If there is no conflicting ownership information, flow continues to step **3507**, otherwise flow continues to step **3506** where the requestor is informed of the conflicting ownership information. Flow returns to step **3502**.

When there is no conflicting ownership information at step **3505**, the secure domain name registry service (website **3308**) informs the requestor that there is no conflicting ownership information and prompts the requester to verify the information entered into the online form and select an approved form of payment. After confirmation of the entered information and appropriate payment information, flow continues to step **3508** where the newly registered secure domain name sent to SDNS **3313** over communication link **3326**.

If, at step **3505**, the requested secure domain name does not have a corresponding equivalent non-secure domain name, the present invention informs the requestor of the situation and prompts the requester for acquiring the corresponding equivalent non-secure domain name for an increased fee. By accepting the offer, the present invention automatically registers the corresponding equivalent non-secure domain name with standard DNS **3325** in a well-known manner. Flow then continues to step **3508**.

G. Tunneling Secure Address Hopping Protocol Through Existing Protocol Using Web Proxy

The present invention also provides a technique for implementing the field hopping schemes described above in an application program on the client side of a firewall between two computer networks, and in the network stack on the server side of the firewall. The present invention uses a new secure connectionless protocol that provides good denial of service rejection capabilities by layering the new protocol on top of an existing IP protocol, such as the ICMP, UDP or TCP protocols. Thus, this aspect of the present invention does not require changes in the Internet infrastructure.

According to the invention, communications are protected by a client-side proxy application program that accepts unencrypted, unprotected communication packets from a local browser application. The client-side proxy application program tunnels the unencrypted, unprotected communication packets through a new protocol, thereby protecting the communications from a denial of service at the server side. Of course, the unencrypted, unprotected communication packets can be encrypted prior to tunneling.

The client-side proxy application program is not an operating system extension and does not involve any modifications to the operating system network stack and drivers. Consequently, the client is easier to install, remove and support in comparison to a VPN. Moreover, the client-side proxy application can be allowed through a corporate firewall using a much smaller "hole" in the firewall and is less of a security risk in comparison to allowing a protocol layer VPN through a corporate firewall.

The server-side implementation of the present invention authenticates valid field-hopped packets as valid or invalid very early in the server packet processing, similar to a standard virtual private network, for greatly minimizing the impact of a denial of service attempt in comparison to normal TCP/IP and HTTP communications, thereby protecting the server from invalid communications.

FIG. **36** shows a system block diagram of a computer network **3600** in which a virtual private connection according to the present invention can be configured to more easily traverse a firewall between two computer networks. FIG. **37** shows a flow diagram **3700** for establishing a virtual private connection that is encapsulated using an existing network protocol.

In FIG. **36** a local area network (LAN) **3601** is connected to another computer network **3602**, such as the Internet, through a firewall arrangement **3603**. Firewall arrangement operates in a well-known manner to interface LAN **3601** to computer network **3602** and to protect LAN **3601** from attacks initiated outside of LAN **3601**.

A client computer **3604** is connected to LAN **3601** in a well-known manner. Client computer **3604** includes an operating system **3605** and a web browser **3606**. Operating system **3605** provides kernel mode functions for operating client computer **3604**. Browser **3606** is an application program for accessing computer network resources connected to LAN **3601** and computer network **3602** in a well-known manner. According to the present invention, a proxy application **3607** is also stored on client computer **3604** and operates at an application layer in conjunction with browser **3606**. Proxy application **3607** operates at the application layer within client computer **3604** and when enabled, modifies unprotected, unencrypted message packets generated by browser **3606** by inserting data into the message packets that are used for forming a virtual private connection between client computer **3604** and a server computer connected to LAN **3601** or computer network **3602**. According to the invention, a virtual private connection does not provide the same level of security to the client computer as a virtual private network. A virtual private connection can be conveniently authenticated so that, for example, a denial of service attack can be rapidly rejected, thereby providing different levels of service that can be subscribed to by a user.

Proxy application **3607** is conveniently installed and uninstalled by a user because proxy application **3607** operates at the application layer within client computer **3604**. On installation, proxy application **3607** preferably configures browser **3606** to use proxy application for all web communications. That is, the payload portion of all message packets is modified with the data for forming a virtual private connection between client computer **3604** and a server computer. Preferably, the data for forming the virtual private connection contains field-hopping data, such as described above in connection with VPNs. Also, the modified message packets preferably conform to the UDP protocol. Alternatively, the modified message packets can conform to the TCP/IP protocol or the ICMP protocol. Alternatively, proxy application **3606** can be selected and enabled through, for example, an option provided by browser **3606**. Additionally, proxy application **3607** can be enabled so that only the payload portion of specially designated message packets is modified with the data for forming a virtual private connection between client computer **3604** and a designated host computer. Specially designated message packets can be, for example, selected predetermined domain names.

Referring to FIG. **37**, at step **3701**, unprotected and unencrypted message packets are generated by browser **3606**. At step **3702**, proxy application **3607** modifies the payload portion of all message packets by tunneling the data for forming a virtual private connection between client computer **3604** and a destination server computer into the payload portion. At step, **3703**, the modified message packets are sent from client computer **3604** to, for example, website (server computer) **3608** over computer network **3602**.

Website **3608** includes a VPN guard portion **3609**, a server proxy portion **3610** and a web server portion **3611**. VPN guard portion **3609** is embedded within the kernel layer of the operating system of website **3608** so that large bandwidth attacks on website **3608** are rapidly rejected. When client computer **3604** initiates an authenticated connection to website **3608**, VPN guard portion **3609** is keyed with the hopping sequence contained in the message packets from client computer **3604**, thereby performing a strong authentication of the client packet streams entering website **3608** at step **3704**. VPN guard portion **3609** can be configured for providing different levels of authentication and, hence, quality of service, depending upon a subscribed level of service. That is, VPN guard portion **3609** can be configured to let all message packets through until a denial of service attack is detected, in which case VPN guard portion **3609** would allow only client packet streams conforming to a keyed hopping sequence, such as that of the present invention.

Server proxy portion **3610** also operates at the kernel layer within website **3608** and catches incoming message packets from client computer **3604** at the VPN level. At step **3705**, server proxy portion **3610** authenticates the message packets at the kernel level within host computer **3604** using the destination IP address, UDP ports and discriminator fields. The authenticated message packets are then forwarded to the authenticated message packets to web server portion **3611** as normal TCP web transactions.

At step **3705**, web server portion **3611** responds to message packets received from client computer **3604** in accordance with the particular nature of the message packets by generating reply message packets. For example, when a client computer requests a webpage, web server portion **3611** generates message packets corresponding to the requested webpage. At step **3706**, the reply message packets pass through server proxy portion **3610**, which inserts data into the payload portion of the message packets that are used for forming the virtual private connection between host computer **3608** and client computer **3604** over computer network **3602**. Preferably, the data for forming the virtual private connection is contains field-hopping data, such as described above in connection with VPNs. Server proxy portion **3610** operates at the kernel layer within host computer **3608** to insert the virtual private connection data into the payload portion of the reply message packets. Preferably, the modified message packets sent by host computer **3608** to client computer **3604** conform to the UDP protocol. Alternatively, the modified message packets can conform to the TCP/IP protocol or the ICMP protocol.

At step **3707**, the modified packets are sent from host computer **3608** over computer network **3602** and pass through firewall **3603**. Once through firewall **3603**, the modified packets are directed to client computer **3604** over LAN **3601** and are received at step **3708** by proxy application **3607** at the application layer within client computer **3604**. Proxy application **3607** operates to rapidly evaluate the modified message packets for determining whether the received packets should be accepted or dropped. If the virtual private connection data inserted into the received information packets conforms to expected virtual private connection data, then the received packets are accepted. Otherwise, the received packets are dropped.

While the present invention has been described in connection with the illustrated embodiments, it will be appreciated and understood that modifications may be made without departing from the true spirit and scope of the invention.

What is claimed is:

1. A system for providing a domain name service for establishing a secure communication link, the system comprising:

a domain name service system configured to be connected to a communication network, to store a plurality of domain names and corresponding network addresses, to receive a query for a network address, and to comprise an indication that the domain name service system supports establishing a secure communication link.

2. The system of claim **1**, wherein at least one of the plurality of domain names comprises a top-level domain name.

3. The system of claim **2**, wherein the top-level domain name is a non-standard top-level domain name.

4. The system of claim **3**, wherein the non-standard top-level domain name is one of .scom, .sorg, .snet, .sgov, .sedu, .smil and .sint.

5. The system of claim **2**, wherein the domain name service system is configured to authenticate the query using a cryptographic technique.

6. The system of claim **1**, wherein the communication network includes the Internet.

7. The system of claim **1**, wherein the domain name service system comprises an edge router.

8. The system of claim **1**, wherein the domain name service system is connectable to a virtual private network through the communication network.

9. The system of claim **8**, wherein the virtual private network is one of a plurality of secure communication links in a hierarchy of secure communication links.

10. The system of claim **8**, wherein the virtual private network is based on inserting into each data packet communicated over a secure communication link one or more data values that vary according to a pseudo-random sequence.

11. The system of claim **8**, wherein the virtual private network is based on a network address hopping regime that is used to pseudorandomly change network addresses in packets transmitted between a first device and a second device.

12. The system of claim **8**, wherein the virtual private network is based on comparing a value in each data packet transmitted between a first device and a second device to a moving window of valid values.

13. The system of claim **8**, wherein the virtual private network is based on a comparison of a discriminator field in a header of each data packet to a table of valid discriminator fields maintained for a first device.

14. The system of claim **1**, wherein the domain name service system is configured to respond to the query for the network address.

15. The system of claim **1**, wherein the domain name service system is configured to provide, in response to the query, the network address corresponding to a domain name from the plurality of domain names and the corresponding network addresses.

16. The system of claim **1**, wherein the domain name service system is configured to receive the query initiated from a first location, the query requesting the network address associated with a domain name, wherein the domain name service system is configured to provide the network address associated with a second location, and wherein the domain name service system is configured to support establishing a secure communication link between the first location and the second location.

17. The system of claim **1**, wherein the domain name service system is connected to a communication network, stores a plurality of domain names and corresponding network addresses, and comprises an indication that the domain name service system supports establishing a secure communication link.

18. The system of claim **1**, wherein at least one of the plurality of domain names is reserved for secure communication links.

19. The system of claim **1**, wherein the domain name service system comprises a server.

20. The system of claim **19**, wherein the domain name service system further comprises a domain name database, and wherein the domain name database stores the plurality of domain names and the corresponding network addresses.

21. The system of claim **1**, wherein the domain name service system comprises a server, wherein the server comprises a domain name database, and wherein the domain name database stores the plurality of domain names and the corresponding network addresses.

22. The system of claim **1**, wherein the domain name service system is configured to store the corresponding network addresses for use in establishing secure communication links.

23. The system of claim 1, wherein the domain name service system is configured to authenticate the query for the network address.

24. The system of claim 1, wherein at least one of the plurality of domain names comprises an indication that the domain name service system supports establishing a secure communication link.

25. The system of claim 1, wherein at least one of the plurality of domain names comprises a secure name.

26. The system of claim 1, wherein at least one of the plurality of domain names enables establishment of a secure communication link.

27. The system of claim 1, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location transparently to a user at the first location.

28. The system of claim 1, wherein the secure communication link uses encryption.

29. The system of claim 1, wherein the secure communication link is capable of supporting a plurality of services.

30. The system of claim 29, wherein the plurality of services comprises a plurality of communication protocols, a plurality of application programs, multiple sessions, or a combination thereof.

31. The system of claim 30, wherein the plurality of application programs comprises items selected from a group consisting of the following: video conferencing, e-mail, a word processing program, and telephony.

32. The system of claim 29, wherein the plurality of services comprises audio, video, or a combination thereof.

33. The system of claim 1, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location.

34. The system of claim 33, wherein the query is initiated from the first location, wherein the second location comprises a computer, and wherein the network address is an address associated with the computer.

35. The system of claim 1, wherein the domain name service system comprises a domain name database connected to a communication network and storing a plurality of domain names and corresponding network addresses for communication,

wherein the domain name database is configured so as to provide a network address corresponding to a domain name in response to a query in order to establish a secure communication link.

36. A machine-readable medium comprising instructions executable in a domain name service system, the instructions comprising code for:

connecting the domain name service system to a communication network;

storing a plurality of domain names and corresponding network addresses;

receiving a query for a network address; and

supporting an indication that the domain name service system supports establishing a secure communication link.

37. The machine-readable medium of claim 36, wherein the instructions comprise code for storing the plurality of domain names and corresponding network addresses including at least one top-level domain name.

38. The machine-readable medium of claim 36, wherein the instructions comprise code for responding to the query for the network address.

39. The machine-readable medium of claim 36, wherein the instructions comprise code for providing, in response to the query, the network address corresponding to a domain name from the plurality of domain names and the corresponding network addresses.

40. The machine-readable medium of claim 36, wherein the instructions comprise code for receiving the query for a network address associated with a domain name and initiated from a first location, and providing a network address associated with a second location, and establishing a secure communication link between the first location and the second location.

41. The machine-readable medium of claim 36, wherein the instructions comprise code for indicating that the domain name service system supports the establishment of a secure communication link.

42. The machine-readable medium of claim 36, wherein the instructions comprise code for reserving at least one of the plurality of domain names for secure communication links.

43. The machine-readable medium of claim 36, wherein the code resides on a server.

44. The machine-readable medium of claim 36, wherein the instructions comprise code for storing a plurality of domain names and corresponding network addresses so as to define a domain name database.

45. The machine-readable medium of claim 36, wherein the code resides on a server, and the instructions comprise code for creating a domain name database configured to store the plurality of domain names and the corresponding network addresses.

46. The machine-readable medium of claim 36, wherein the instructions comprise code for storing the corresponding network addresses for use in establishing secure communication links.

47. The machine-readable medium of claim 36, wherein the instructions comprise code for authenticating the query for the network address.

48. The machine-readable medium of claim 36, wherein at least one of the plurality of domain names includes an indication that the domain name service system supports the establishment of a secure communication link.

49. The machine-readable medium of claim 36, wherein at least one of the plurality of domain names includes a secure name.

50. The machine-readable medium of claim 36, wherein at least one of the plurality of domain names is configured so as to enable establishment of a secure communication link.

51. The machine-readable medium of claim 36, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location transparently to a user at the first location.

52. The machine-readable medium of claim 36, wherein the secure communication link uses encryption.

53. The machine-readable medium of claim 36, wherein the secure communication link is capable of supporting a plurality of services.

54. The machine-readable medium of claim 53, wherein the plurality of services comprises a plurality of communication protocols, a plurality of application programs, multiple sessions, or a combination thereof.

55. The machine-readable medium of claim 54, wherein the plurality of application programs comprises items selected from a group consisting of the following:

video conferencing, e-mail, a word processing program, and telephony.

56. The machine-readable medium of claim 53, wherein the plurality of services comprises audio, video, or a combination thereof.

57. The machine-readable medium of claim 36, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location.

58. The machine-readable medium of claim 57, wherein the instructions include code for receiving a query initiated from the first location, wherein the second location comprises a computer, and wherein the network address is an address associated with the computer.

59. The machine-readable medium of claim 36, wherein the domain name service system comprises a domain name database connected to a communication network and storing a plurality of domain names and corresponding network addresses for communication,

wherein the domain name database is configured so as to provide a network address corresponding to a domain name is response to the query in order to establish a secure communication link.

60. A method of providing a domain name service for establishing a secure communication link, the method comprising:

connecting a domain name service system to a communication network, the domain name service system comprising an indication that the domain name service system supports establishing a secure communication link;

storing a plurality of domain names and corresponding network addresses; and

receiving a query for a network address for communication.

* * * * *

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

March 30, 2011

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF:

U.S. PATENT: *7,490,151*
ISSUE DATE: *February 10, 2009*

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

T. Wallace
T. WALLACE
Certifying Officer

Plaintiffs' VirnetX Exhibit
VirnetX, Inc. v. Apple, Inc.
PX007
C.A. 6:10-cv-0417

US007490151B2

(12) **United States Patent**
Munger et al.

(10) **Patent No.: US 7,490,151 B2**
(45) **Date of Patent: Feb. 10, 2009**

(54) ESTABLISHMENT OF A SECURE COMMUNICATION LINK BASED ON A DOMAIN NAME SERVICE (DNS) REQUEST

(75) Inventors: **Edward Colby Munger**, Crownsville, MD (US); **Robert Dunham Short, III**, Leesburg, VA (US); **Victor Larson**, Fairfax, VA (US); **Michael Williamson**, South Riding, VA (US)

(73) Assignee: **Virnetx Inc.**, Scotts Valley Drive, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 818 days.

(21) Appl. No.: **10/259,494**

(22) Filed: **Sep. 30, 2002**

(65) **Prior Publication Data**

US 2003/0037142 A1 Feb. 20, 2003

Related U.S. Application Data

(60) Division of application No. 09/504,783, filed on Feb. 15, 2000, now Pat. No. 6,502,135, which is a continuation-in-part of application No. 09/429,643, filed on Oct. 29, 1999, now Pat. No. 7,010,604.

(60) Provisional application No. 60/137,704, filed on Jun. 7, 1999, provisional application No. 60/106,261, filed on Oct. 30, 1998.

(51) **Int. Cl.**
G06F 15/173 (2006.01)

(52) **U.S. Cl.** **709/225**; 709/229

(58) **Field of Classification Search** 709/217–225, 709/229; 713/201

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,933,846 A 6/1990 Humphrey et al.

(Continued)

FOREIGN PATENT DOCUMENTS

DE 199 24 575 12/1999

(Continued)

OTHER PUBLICATIONS

Search Report (dated Aug. 23, 2002), International Application No. PCT/US01/13260.

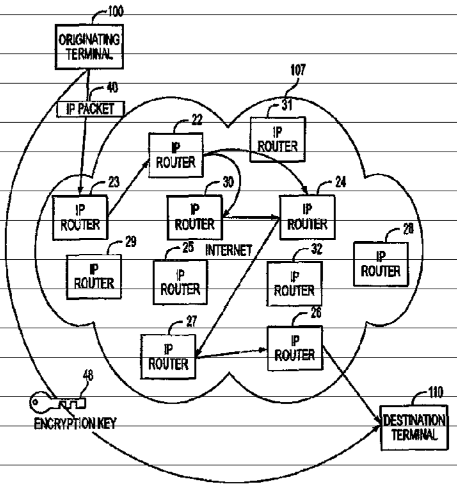
(Continued)

Primary Examiner—Krisna Lim
(74) *Attorney, Agent, or Firm*—McDermott Will & Emery

(57) **ABSTRACT**

A plurality of computer nodes communicate using seemingly random Internet Protocol source and destination addresses. Data packets matching criteria defined by a moving window of valid addresses are accepted for further processing, while those that do not meet the criteria are quickly rejected. Improvements to the basic design include (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities.

16 Claims, 35 Drawing Sheets



U.S. PATENT DOCUMENTS

4,988,990	A		1/1991	Warrior	
5,164,986	A	*	11/1992	Bright	380/273
5,276,735	A		1/1994	Boebert et al.	
5,311,593	A		5/1994	Carmi	
5,329,521	A		7/1994	Walsh et al.	
5,341,426	A		8/1994	Barney et al.	
5,367,643	A		11/1994	Chang et al.	
5,559,883	A		9/1996	Williams	
5,561,669	A		10/1996	Lenney et al.	
5,588,060	A		12/1996	Aziz	
5,625,626	A		4/1997	Umekita	
5,654,695	A		8/1997	Olnowich et al.	
5,682,480	A		10/1997	Nakagawa	
5,689,566	A		11/1997	Nguyen	
5,740,375	A		4/1998	Dunne et al.	
5,774,660	A		6/1998	Brendel et al.	
5,787,172	A		7/1998	Arnold	
5,790,548	A	*	8/1998	Sistanizadeh et al.	370/401
5,796,942	A		8/1998	Esbensen	
5,805,801	A		9/1998	Holloway et al.	
5,842,040	A		11/1998	Hughes et al.	
5,845,091	A		12/1998	Dunne et al.	
5,867,650	A		2/1999	Osterman	
5,870,610	A		2/1999	Beyda et al.	
5,878,231	A		3/1999	Baehr et al.	
5,892,903	A		4/1999	Klaus	
5,898,830	A	*	4/1999	Wesinger et al.	726/15
5,905,859	A		5/1999	Holloway et al.	
5,918,019	A		6/1999	Valencia	
5,996,016	A		11/1999	Thalheimer et al.	
6,006,259	A		12/1999	Adelman et al.	
6,006,272	A		12/1999	Aravamudan et al.	
6,016,318	A		1/2000	Tomoike	
6,016,512	A		1/2000	Huitema	
6,041,342	A		3/2000	Yamaguchi	
6,052,788	A		4/2000	Wesinger, Jr. et al.	
6,055,574	A		4/2000	Smorodinsky et al.	
6,061,736	A		5/2000	Rochberger et al.	
6,079,020	A	*	6/2000	Liu	713/201
6,092,200	A		7/2000	Muniyappa et al.	
6,101,182	A	*	8/2000	Sistanizadeh et al.	370/352
6,119,171	A		9/2000	Alkhatib	
6,119,234	A	*	9/2000	Aziz et al.	713/201
6,147,976	A		11/2000	Shand et al.	
6,157,957	A		12/2000	Berthaud	
6,158,011	A		12/2000	Chen et al.	
6,168,409	B1		1/2001	Fare	
6,175,867	B1		1/2001	Taghadoss	
6,178,409	B1		1/2001	Weber et al.	
6,178,505	B1		1/2001	Schneider et al.	
6,179,102	B1		1/2001	Weber et al.	
6,222,842	B1		4/2001	Sasyan et al.	
6,226,751	B1		5/2001	Arrow et al.	
6,233,618	B1		5/2001	Shannon	
6,243,360	B1		6/2001	Basilico	
6,243,749	B1		6/2001	Sitaraman et al.	
6,243,754	B1		6/2001	Guerin et al.	
6,256,671	B1	*	7/2001	Strentzsch et al.	709/227
6,263,445	B1		7/2001	Blumenau	
6,286,047	B1		9/2001	Ramanathan et al.	
6,301,223	B1		10/2001	Hrastar et al.	
6,308,274	B1		10/2001	Swift	
6,311,207	B1		10/2001	Mighdoll et al.	
6,324,161	B1		11/2001	Kirch	
6,330,562	B1		12/2001	Boden et al.	
6,332,158	B1	*	12/2001	Risley et al.	709/219
6,353,614	B1		3/2002	Borella et al.	
6,425,003	B1	*	7/2002	Herzog et al.	709/223
6,430,155	B1		8/2002	Davie et al.	
6,430,610	B1		8/2002	Carter	
6,487,598	B1		11/2002	Valencia	
6,502,135	B1	*	12/2002	Munger et al.	709/225
6,505,232	B1		1/2003	Mighdoll et al.	
6,510,154	B1		1/2003	Mayes et al.	
6,549,516	B1		4/2003	Albert et al.	
6,557,037	B1		4/2003	Provino	
6,571,296	B1		5/2003	Dillon	
6,571,338	B1		5/2003	Shaio et al.	
6,581,166	B1		6/2003	Hirst et al.	
6,606,708	B1	*	8/2003	Devine et al.	713/201
6,618,761	B2		9/2003	Munger et al.	
6,671,702	B2		12/2003	Kruglikov et al.	
6,687,551	B2		2/2004	Steindl	
6,714,970	B1		3/2004	Fiveash et al.	
6,717,949	B1		4/2004	Boden et al.	
6,751,738	B2	*	6/2004	Wesinger et al.	713/201
6,760,766	B1		7/2004	Sahlqvist	
6,826,616	B2		11/2004	Larson et al.	
6,839,759	B2		1/2005	Larson et al.	
7,010,604	B1		3/2006	Munger et al.	
7,133,930	B2		11/2006	Munger et al.	
7,188,180	B2		3/2007	Larson et al.	
7,197,563	B2		3/2007	Sheymov et al.	
2002/0004898	A1		1/2002	Droge	
2003/0196122	A1	*	10/2003	Wesinger et al.	713/201
2005/0055306	A1		3/2005	Miller et al.	
2006/0059337	A1	*	3/2006	Poyhonen et al.	713/165

FOREIGN PATENT DOCUMENTS

EP	0 814 589	12/1997
EP	0 814 589 A	12/1997
EP	0 838 930	4/1998
EP	0 838 930 A	4/1998
EP	836306 A1	4/1998
EP	0 858 189	8/1998
GB	2 317 792	4/1998
GB	2 317 792 A	4/1998
GB	2 334 181 A	8/1999
GB	2334181 A	8/1999
WO	9827783 A	6/1998
WO	WO 98/27783	6/1998
WO	WO 9827783 A	6/1998
WO	WO 98 55930	12/1998
WO	WO 98 59470	12/1998
WO	WO 99 38081	7/1999
WO	WO 99 48303	9/1999
WO	WO 00/17775	3/2000
WO	WO 00/70458	11/2000
WO	WO 01 50688	7/2001

OTHER PUBLICATIONS

Donald E. Eastlake, 3rd, "Domain Name System Security Extensions", Internet Draft, Apr. 1998, pp. 1-51.

D. B. Chapman et al., "Building Internet Firewalls", Nov. 1995, pp. 278-375.

P. Srisuresh et al., "DNA extensions to Network address Translators (DNS_ALG)", Internet Draft, Jul. 1998, pp. 1-27.

James E. Bellaire, "New Statement of Rules—Naming Internet Domains", Internet Newsgroup, Jul. 30, 1995, 1 page.

D. Clark, "US Calls for Private Domain-Name System", Computer Society, Aug. 1, 1998, pp. 22-25.

August Bequai, "Balancing Legal Concerns Over Crime and Security in Cyberspace", Computer & Security, vol. 17, No. 4, 1998, pp. 293-298.

Rich Winkel, "CAQ: Networking With Spooks: The NET & The Control Of Information", Internet Newsgroup, Jun. 21, 1997, 4 pages.

Search Report (dated Jun. 18, 2002), International Application No. PCT/US01/13260.

Search Report (dated Jun. 28, 2002), International Application No. PCT/US01/13261.

Donald E. Eastlake, "Domain Name System Security Extensions", DNS Security Working Group, Apr. 1998, 51 pages.

D. B. Chapman et al., "Building Internet Firewalls", Nov. 1995, pp. 278-297 and pp. 351-375.
P. Srisuresh et al., "DNS extensions to Network Address Translators", Jul. 1998, 27 pages.
Laurie Wells, "Security Icon", Oct. 19, 1998, 1 page.
W. Stallings, "Cryptography And Network Security", 2nd Edition, Chapter 13, IP Security, Jun. 8, 1998, pp. 399-440.
W. Stallings, "New Cryptography and Network Security Book", Jun. 8, 1998, 3 pages.
Search Report (dated Aug. 20, 2002), International Application No. PCT/US01/04340.
Shree Murthy et al., "Congestion-Oriented Shortest Multipath Routing", Proceedings of IEEE Infocom, 1996, pp. 1028-1036.
Jim Jones et al., "Distributed Denial of Service Attacks: Defenses", Global Integrity Corporation, 2000, pp. 1-14.
Fasbender, Kesdogan, and Kubitz: "Variable and Scalable Security: Protection of Location Information in Mobile IP", IEEE publication, 1996, pp. 963-967.
Laurie Wells (Lancasterbibelmail MSN COM); "Subject: Security Icon" Usenet Newsgroup, Oct. 19, 1998, XP002200606.
Davila J et al, "Implementation of Virtual Private Networks at the Transport Layer", Information Security, Second International Workshop, ISW '99. Proceedings (Lecture Springer-Verlag Berlin, Germany, [Online] 1999, pp. 85-102, XP002399276, ISBN 3-540-66695-B, retrieved from the Internet: URL: http://www.springerlink.com/content/4uac0tb0heccma89/fulltext.pdf> (Abstract).
Alan 0. Frier et al., "The SSL Protocol Version 3.0", Nov. 18, 1996, printed from http://www.netscape.com/eng/ssl13/ draft302.txt on Feb. 4, 2002, 56 pages.
Davila J et al, "Implementation of Virtual Private Networks at the Transport Layer", Information Security, Second International Workshop, ISW'99. Proceedings (Lecture Springer-Verlag Berlin, Germany, [Online] 1999, pp. 85-102, XP002399276, ISBN 3-540-66695-B, retrieved from the Internet: URL: http://www. springerlink.com/content/4uac0tb0hecoma89/fulltext.pdf>.
Dolev, Shlomi and Ostrovsky, Rafil, Efficient Anonymous Multicast and Reception (Extended Abstract), 16 pages.
F. Halsall, "Data Communications, Computer Networks and Open Systems", Chapter 4, Protocol Basics, 1996, pp. 198-203.
Glossary for the Linux FreeS/WAN project, printed from http://liberty.freeswan.org/freeswan_trees/freeswan-1.3/ doc/glossary.html on Feb. 21, 2002, 25 pages.
J. Gilmore, "Swan: Securing the Internet against Wiretapping", printed from http://liberty.freeswan.org/freeswan_trees/freeswan-1.3.doc/rationale.html on Feb. 21, 2002, 4 pages.
Linux FreeS/WAN Index File, printed from http://liberty.freewan.org/freeswan trees/freeswan-1.3/doc/ on Feb. 21, 2002, 3 pages.
Reiter, Michael K. and Rubin, Aviel D. (AT&T Labs—Research), Crowds: Anonymity for Web Transactions, pp. 1-23.
RFC 2401-Security Architecture for the Internet Protocol (RTP).
RFC 2543-SIP: Session Initiation Protocol (SIP or SIPS).
Rubin, Aviel D., Geer, Daniel, and Ranum, Marcus J. (Wiley Computer Publishing), "Web Security Sourcebook", pp. 82-94.
Search Report, IPER (dataed Nov. 13, 2002), International Application No. PCT/US01/04340.
Search Report, IPER (dated Feb. 6, 2002), International Application No. PCT/US01/13261.
Search Report, IPER (dated Jan. 14, 2003), International Application No. PCT/US01/13260.
Shankar, A.U. "A verified sliding window protocol with variable flow control". Proceedings of ACM SIGCOMM conference on Communications architectures & protocols. pp. 84-91, ACM Press, NY,NY 1986.

* cited by examiner

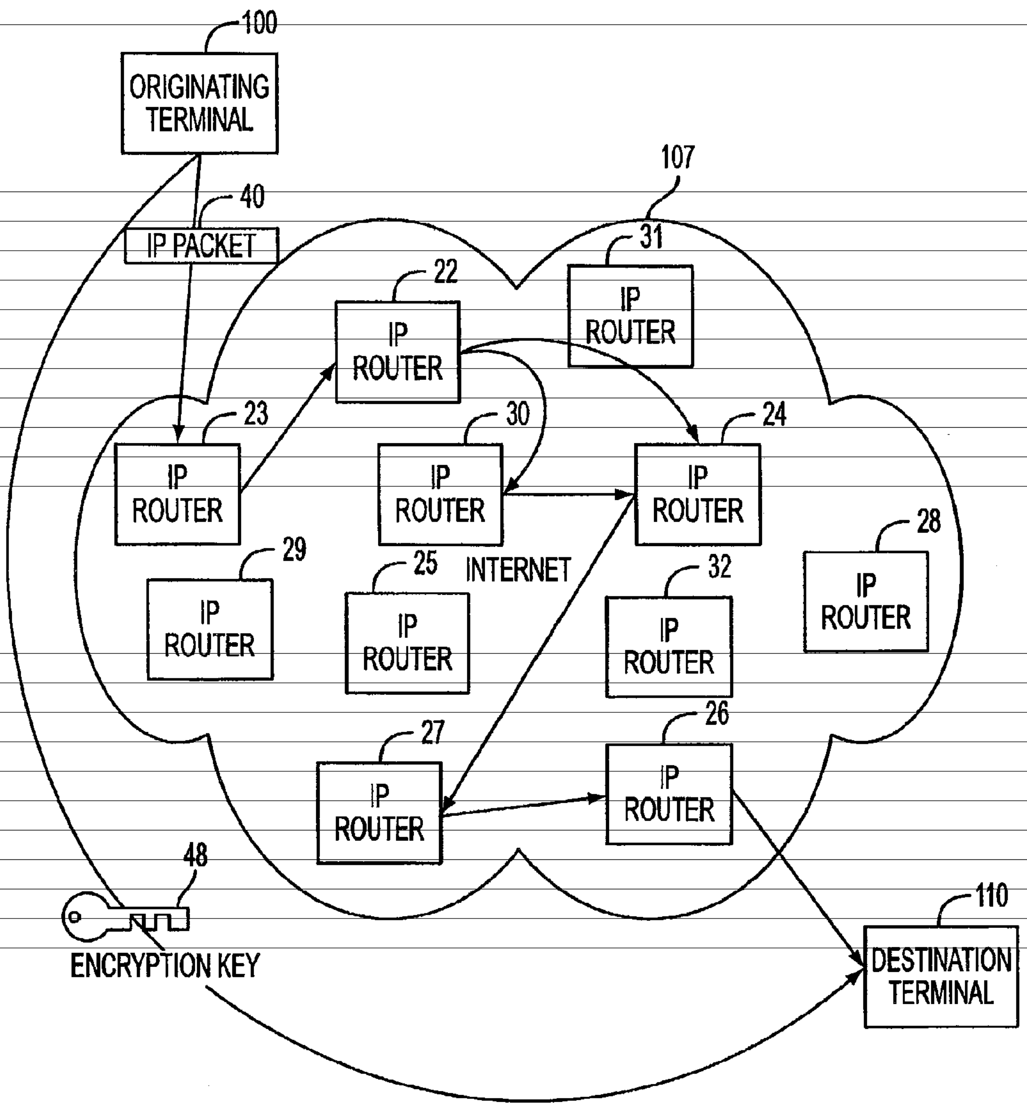

100
ORIGINATING TERMINAL
40
IP PACKET
107
31
IP ROUTER
22
IP ROUTER
23
IP ROUTER
30
IP ROUTER
24
IP ROUTER
29
IP ROUTER
25
IP ROUTER
INTERNET
32
IP ROUTER
28
IP ROUTER
27
IP ROUTER
26
IP ROUTER
48
ENCRYPTION KEY
110
DESTINATION TERMINAL

FIG. 1

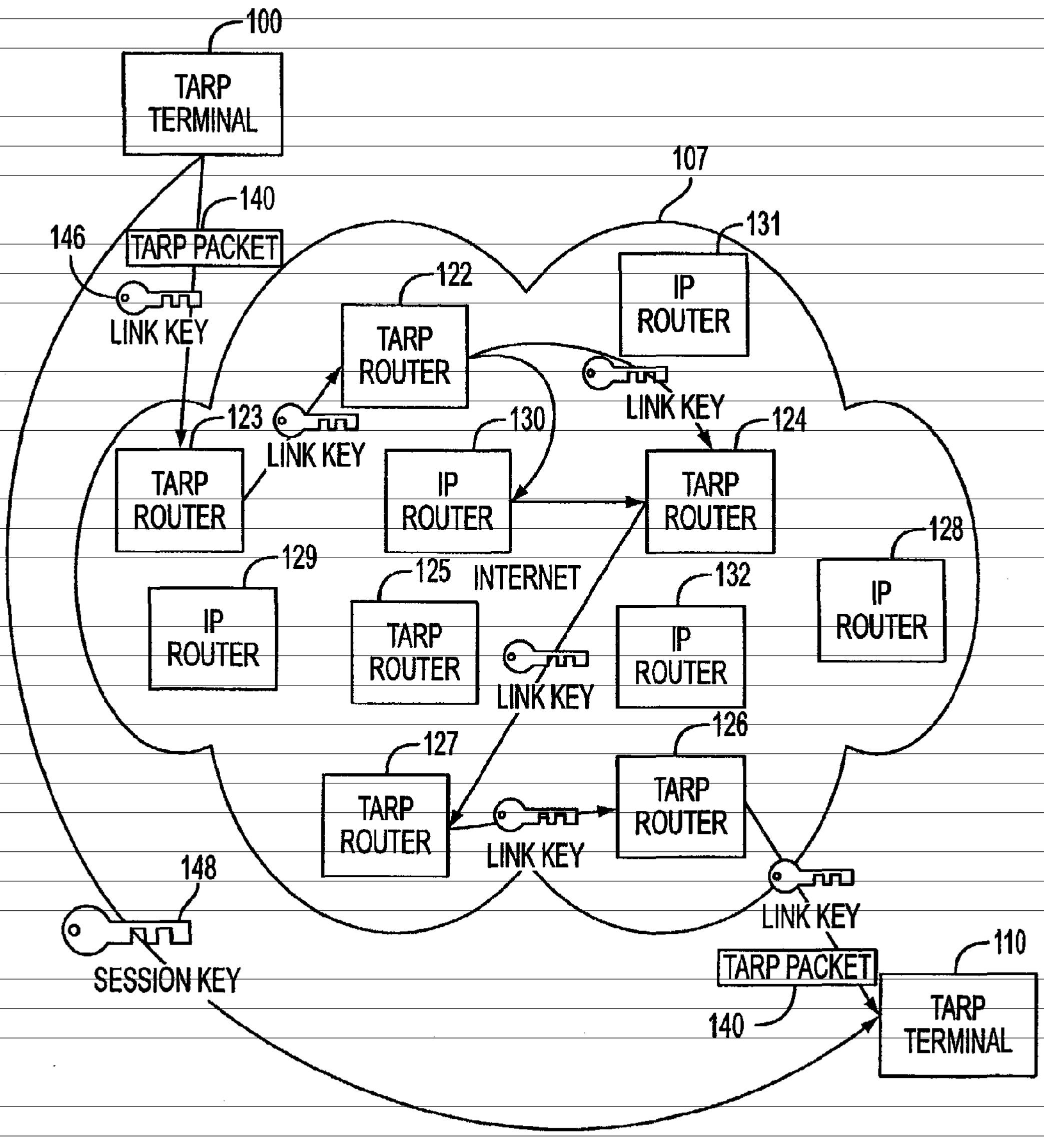

100
TARP TERMINAL
107
140
TARP PACKET
146
LINK KEY
131
IP ROUTER
122
TARP ROUTER
LINK KEY
123
LINK KEY
130
124
TARP ROUTER
IP ROUTER
TARP ROUTER
128
IP ROUTER
129
IP ROUTER
125
TARP ROUTER
INTERNET
132
IP ROUTER
LINK KEY
127
TARP ROUTER
LINK KEY
126
TARP ROUTER
LINK KEY
148
SESSION KEY
TARP PACKET
140
110
TARP TERMINAL

FIG. 2

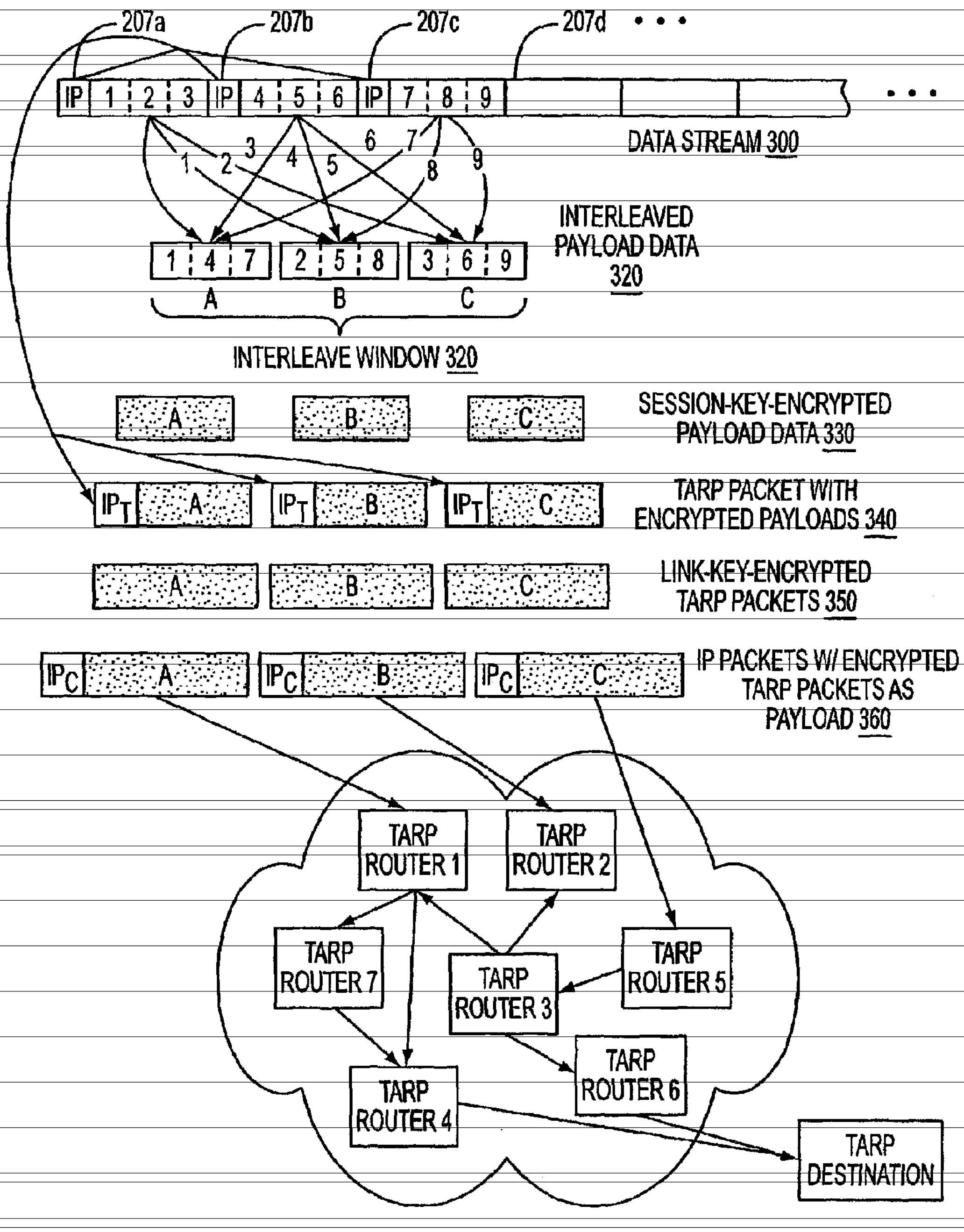

207a
207b
207c
207d
IP 1 2 3 IP 4 5 6 IP 7 8 9
DATA STREAM 300
1 2 3 4 5 6 7 8 9
1 4 7 2 5 8 3 6 9
A B C
INTERLEAVED PAYLOAD DATA 320
INTERLEAVE WINDOW 320
SESSION-KEY-ENCRYPTED PAYLOAD DATA 330
IPT A IPT B IPT C
TARP PACKET WITH ENCRYPTED PAYLOADS 340
LINK-KEY-ENCRYPTED TARP PACKETS 350
IPC A IPC B IPC C
IP PACKETS W/ ENCRYPTED TARP PACKETS AS PAYLOAD 360
TARP ROUTER 1
TARP ROUTER 2
TARP ROUTER 7
TARP ROUTER 3
TARP ROUTER 5
TARP ROUTER 6
TARP ROUTER 4
TARP DESTINATION

FIG. 3A

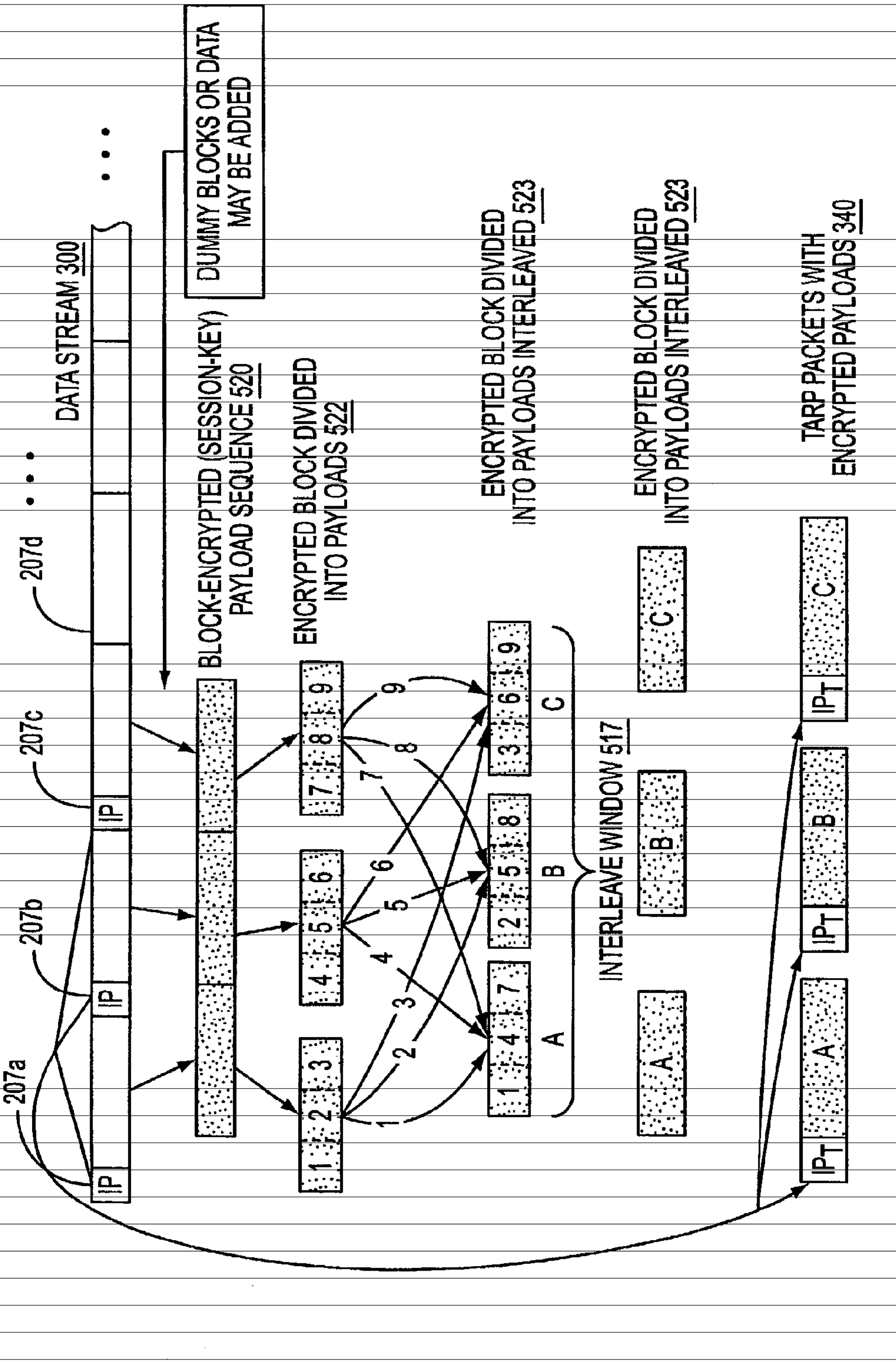

DATA STREAM 300
207a
207b
207c
207d
IP
IP
IP
DUMMY BLOCKS OR DATA MAY BE ADDED
BLOCK-ENCRYPTED (SESSION-KEY) PAYLOAD SEQUENCE 520
ENCRYPTED BLOCK DIVIDED INTO PAYLOADS 522
1 2 3
4 5 6
7 8 9
ENCRYPTED BLOCK DIVIDED INTO PAYLOADS INTERLEAVED 523
1 4 7
2 5 8
3 6 9
A
B
C
INTERLEAVE WINDOW 517
ENCRYPTED BLOCK DIVIDED INTO PAYLOADS INTERLEAVED 523
A
B
C
TARP PACKETS WITH ENCRYPTED PAYLOADS 340
IP_T
A
IP_T
B
IP_T
C

FIG. 3B

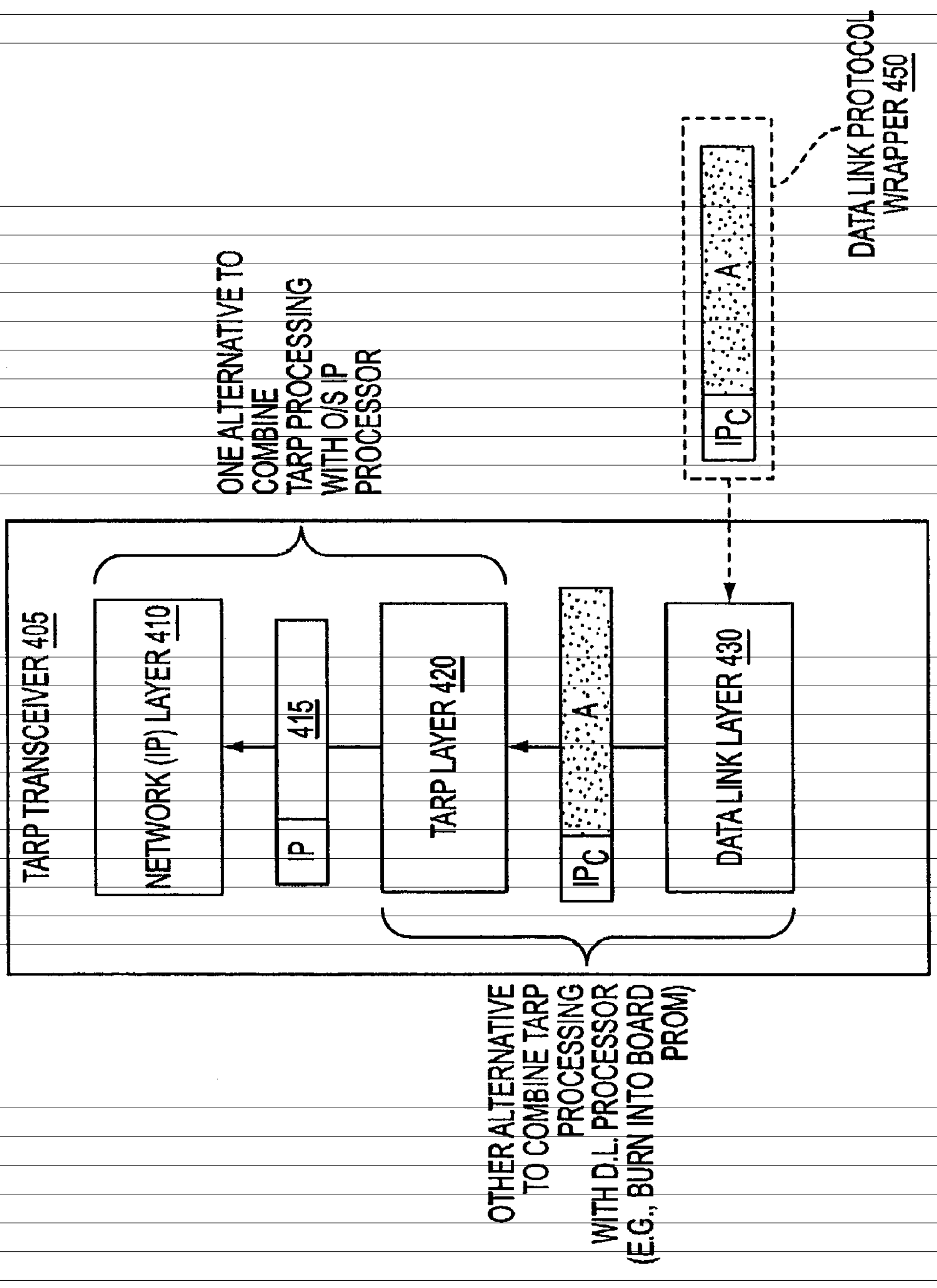

TARP TRANSCEIVER 405
NETWORK (IP) LAYER 410
IP
415
TARP LAYER 420
IPC
A
DATA LINK LAYER 430
ONE ALTERNATIVE TO COMBINE TARP PROCESSING WITH O/S IP PROCESSOR
OTHER ALTERNATIVE TO COMBINE TARP PROCESSING WITH D.L. PROCESSOR (E.G., BURN INTO BOARD PROM)
IPC
A
DATA LINK PROTOCOL WRAPPER 450

FIG. 4

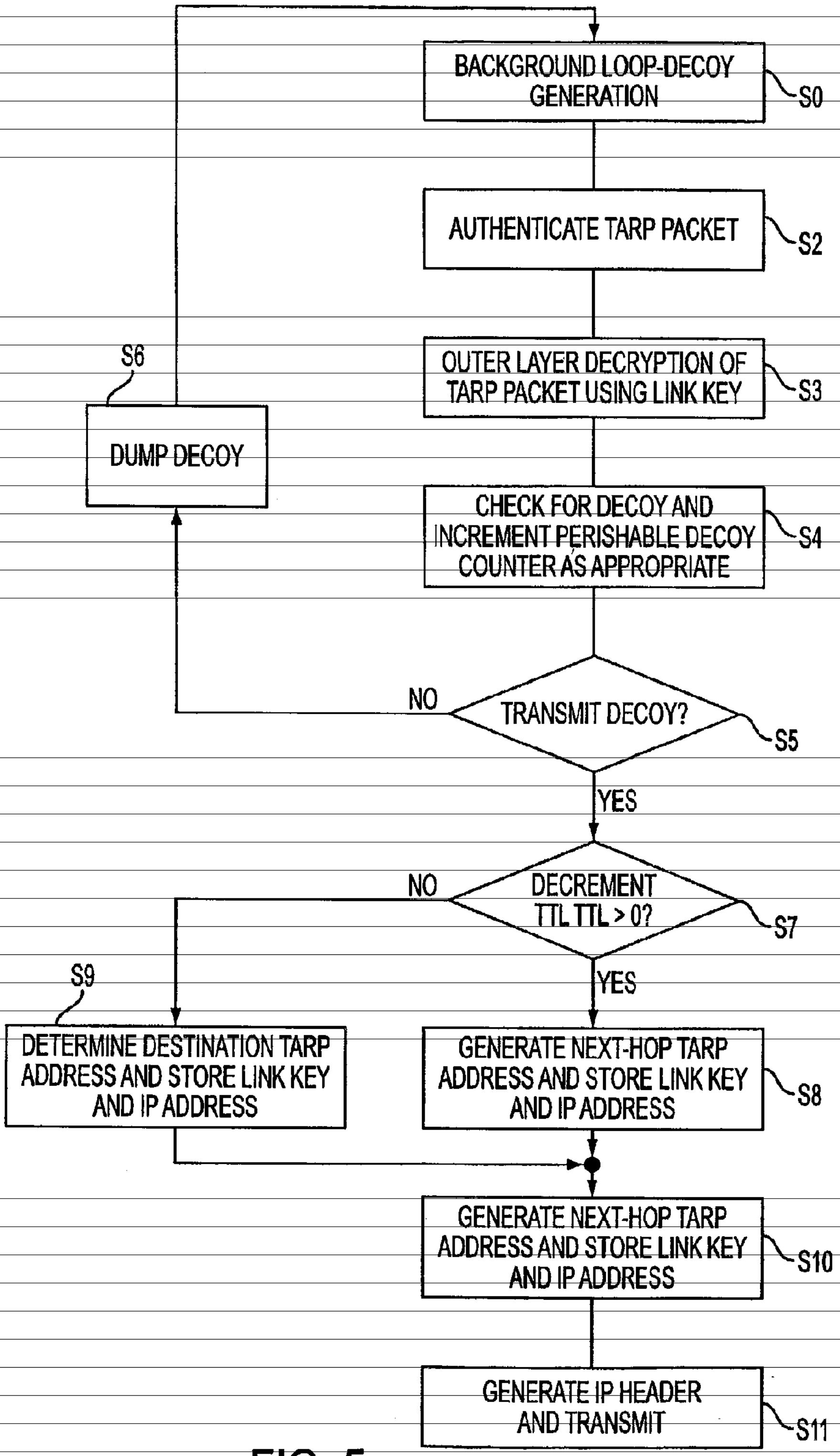

BACKGROUND LOOP-DECOY GENERATION
S0
AUTHENTICATE TARP PACKET
S2
OUTER LAYER DECRYPTION OF TARP PACKET USING LINK KEY
S3
CHECK FOR DECOY AND INCREMENT PERISHABLE DECOY COUNTER AS APPROPRIATE
S4
S6
DUMP DECOY
NO
TRANSMIT DECOY?
S5
YES
NO
DECREMENT TTL TTL > 0?
S7
YES
S9
DETERMINE DESTINATION TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
GENERATE NEXT-HOP TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
S8
GENERATE NEXT-HOP TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
S10
GENERATE IP HEADER AND TRANSMIT
S11

FIG. 5

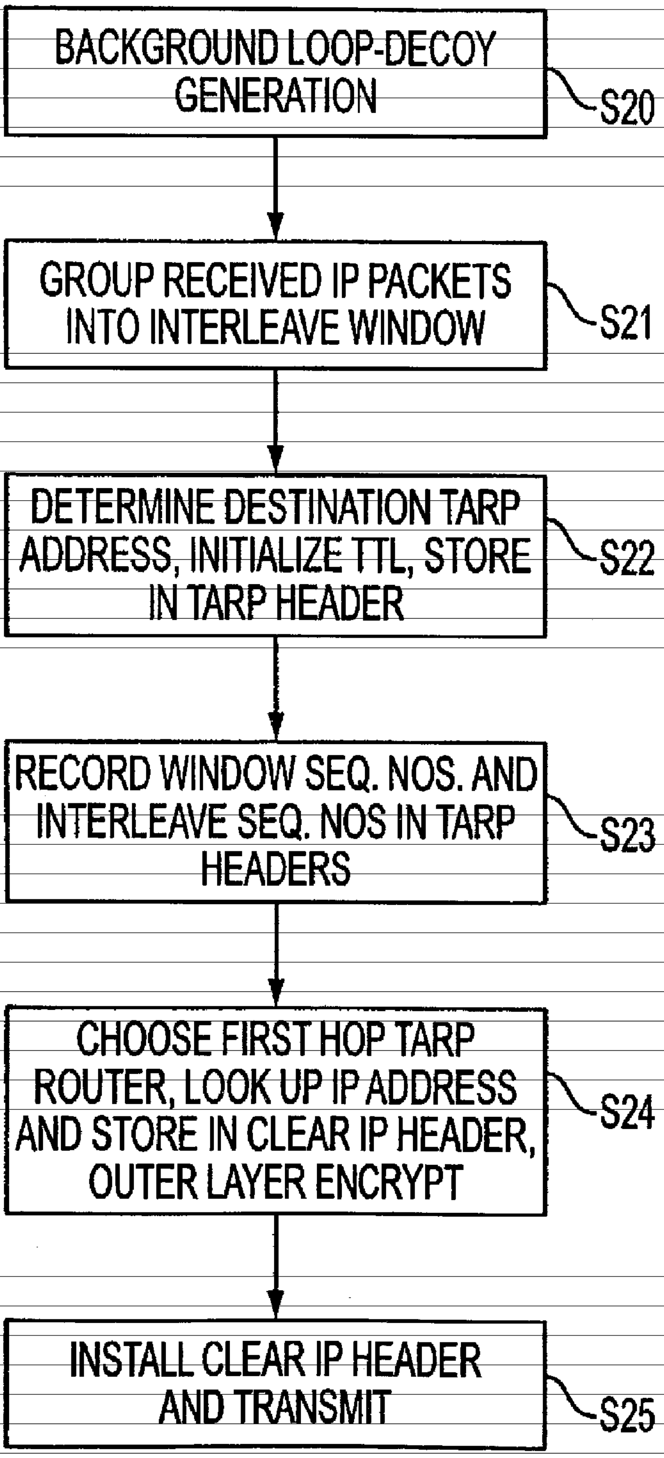

BACKGROUND LOOP-DECOY GENERATION
S20
GROUP RECEIVED IP PACKETS INTO INTERLEAVE WINDOW
S21
DETERMINE DESTINATION TARP ADDRESS, INITIALIZE TTL, STORE IN TARP HEADER
S22
RECORD WINDOW SEQ. NOS. AND INTERLEAVE SEQ. NOS IN TARP HEADERS
S23
CHOOSE FIRST HOP TARP ROUTER, LOOK UP IP ADDRESS AND STORE IN CLEAR IP HEADER, OUTER LAYER ENCRYPT
S24
INSTALL CLEAR IP HEADER AND TRANSMIT
S25

FIG. 6

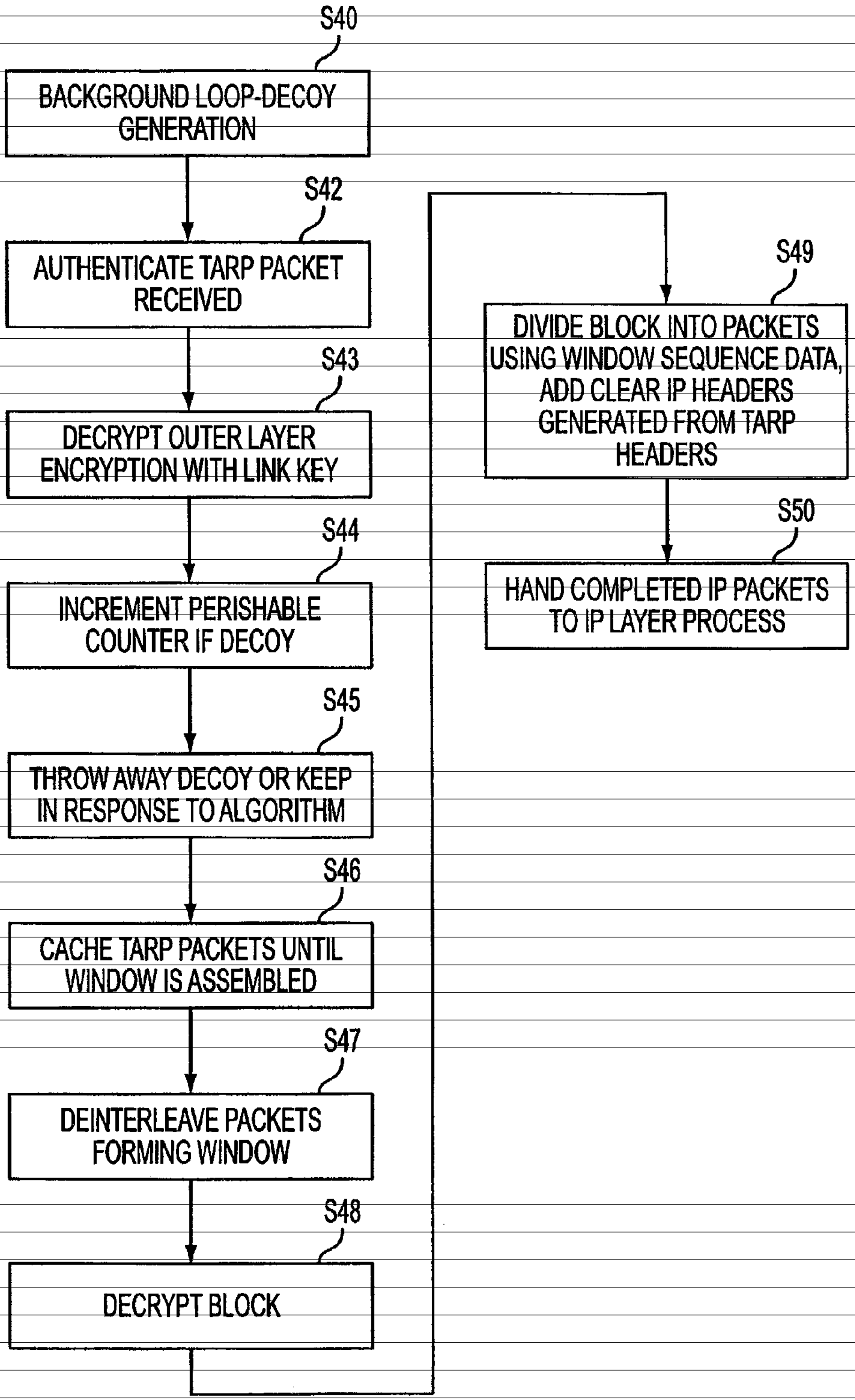

S40
BACKGROUND LOOP-DECOY GENERATION
S42
AUTHENTICATE TARP PACKET RECEIVED
S43
DECRYPT OUTER LAYER ENCRYPTION WITH LINK KEY
S44
INCREMENT PERISHABLE COUNTER IF DECOY
S45
THROW AWAY DECOY OR KEEP IN RESPONSE TO ALGORITHM
S46
CACHE TARP PACKETS UNTIL WINDOW IS ASSEMBLED
S47
DEINTERLEAVE PACKETS FORMING WINDOW
S48
DECRYPT BLOCK
S49
DIVIDE BLOCK INTO PACKETS USING WINDOW SEQUENCE DATA, ADD CLEAR IP HEADERS GENERATED FROM TARP HEADERS
S50
HAND COMPLETED IP PACKETS TO IP LAYER PROCESS

FIG. 7

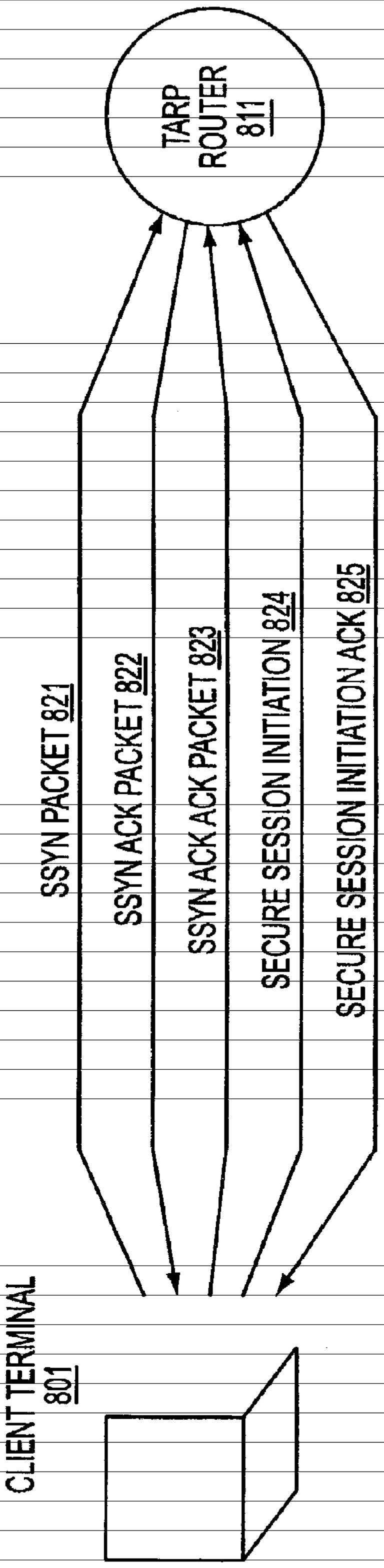

TARP ROUTER 811
SSYN PACKET 821
SSYN ACK PACKET 822
SSYN ACK ACK PACKET 823
SECURE SESSION INITIATION 824
SECURE SESSION INITIATION ACK 825
CLIENT TERMINAL 801

FIG. 8

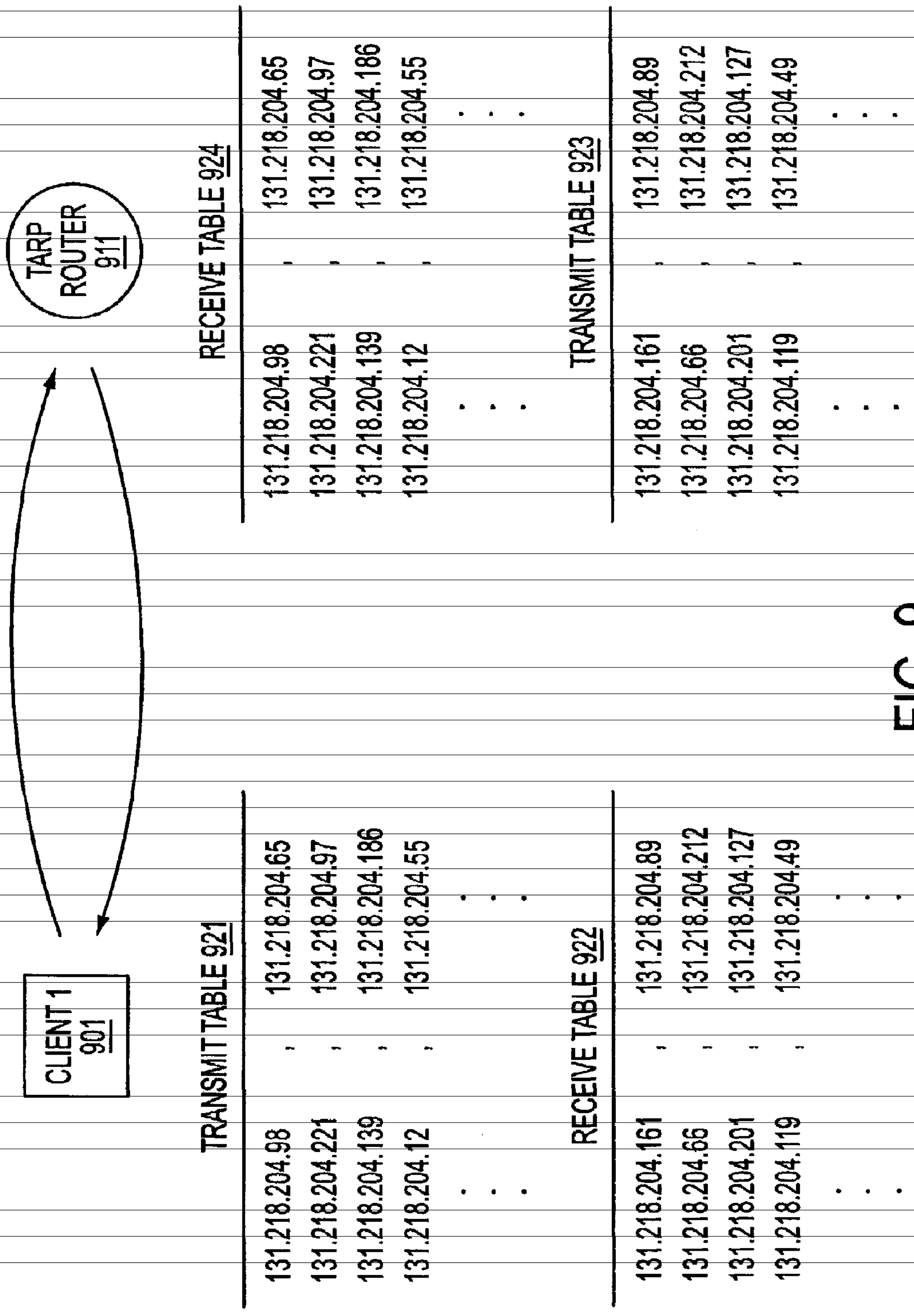

CLIENT 1
901
TARP ROUTER
911
TRANSMIT TABLE 921
131.218.204.98 , 131.218.204.65
131.218.204.221 , 131.218.204.97
131.218.204.139 , 131.218.204.186
131.218.204.12 , 131.218.204.55
. . . . . .
RECEIVE TABLE 922
131.218.204.161 , 131.218.204.89
131.218.204.66 , 131.218.204.212
131.218.204.201 , 131.218.204.127
131.218.204.119 , 131.218.204.49
. . . . . .
RECEIVE TABLE 924
131.218.204.98 , 131.218.204.65
131.218.204.221 , 131.218.204.97
131.218.204.139 , 131.218.204.186
131.218.204.12 , 131.218.204.55
. . . . . .
TRANSMIT TABLE 923
131.218.204.161 , 131.218.204.89
131.218.204.66 , 131.218.204.212
131.218.204.201 , 131.218.204.127
131.218.204.119 , 131.218.204.49
. . . . . .

FIG. 9

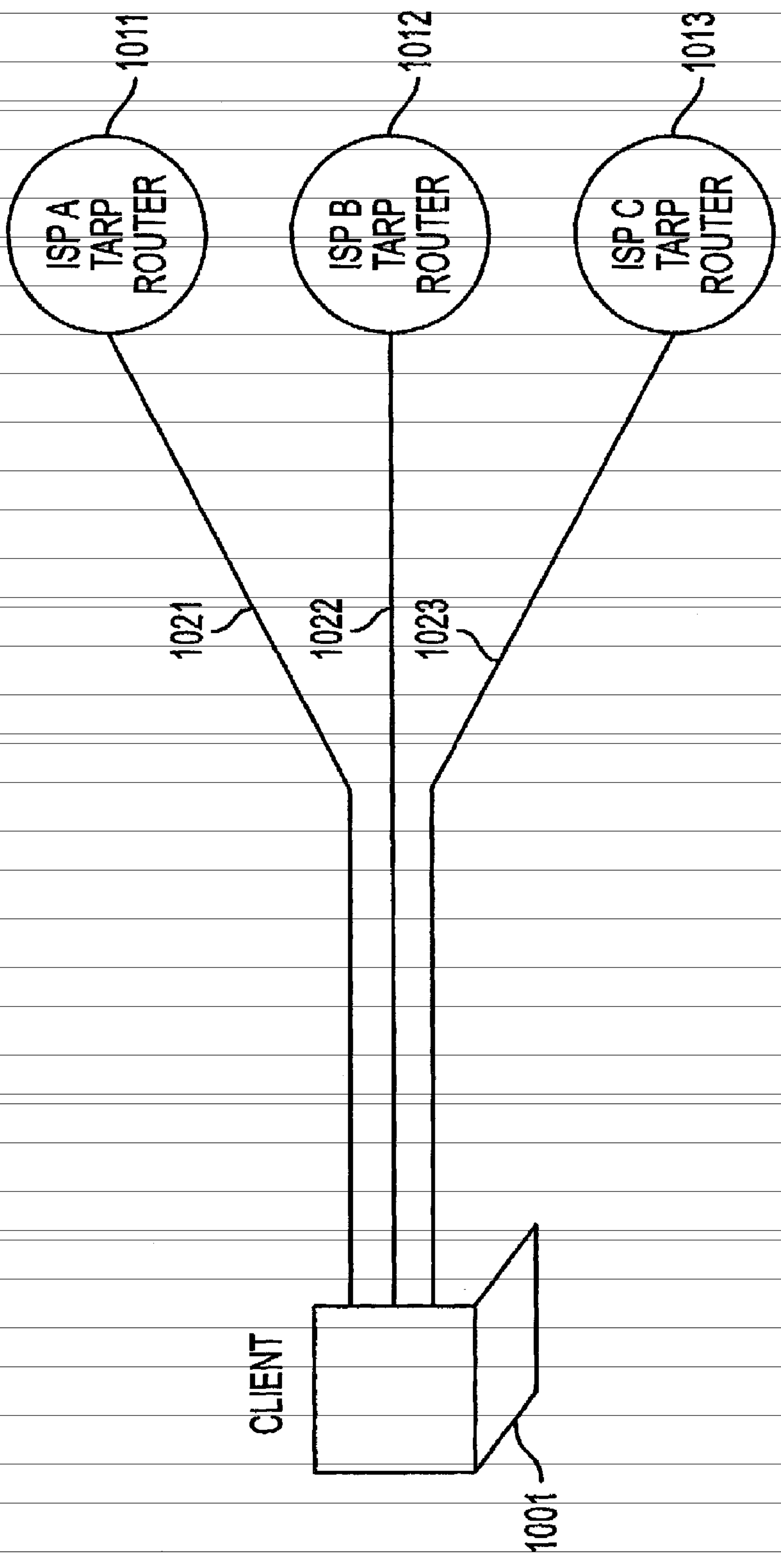

1011
1012
1013
ISP A TARP ROUTER
ISP B TARP ROUTER
ISP C TARP ROUTER
1021
1022
1023
CLIENT
1001

FIG. 10

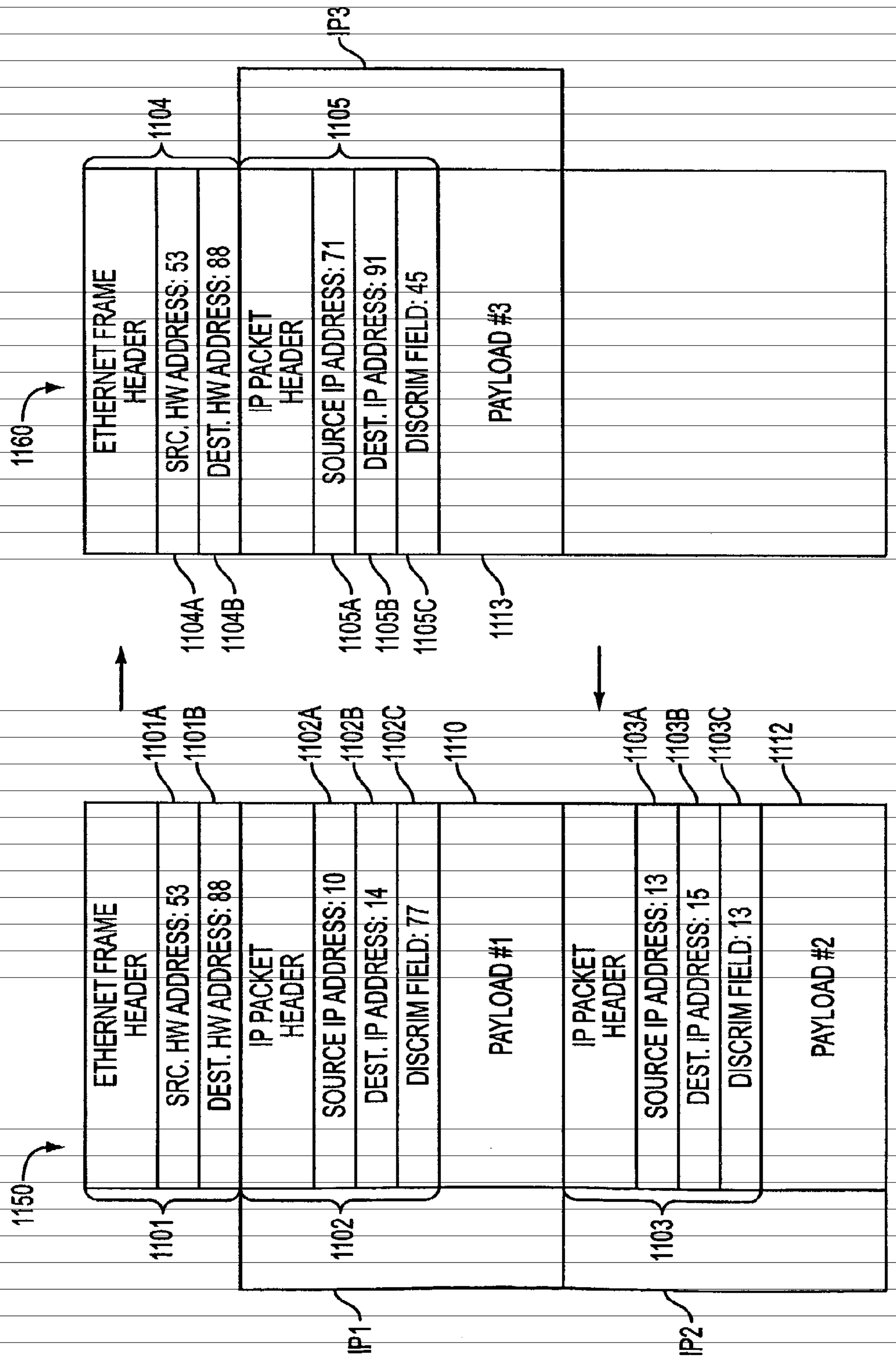

1150
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
IP PACKET HEADER
SOURCE IP ADDRESS: 10
DEST. IP ADDRESS: 14
DISCRIM FIELD: 77
PAYLOAD #1
IP PACKET HEADER
SOURCE IP ADDRESS: 13
DEST. IP ADDRESS: 15
DISCRIM FIELD: 13
PAYLOAD #2
1101
1101A
1101B
1102
1102A
1102B
1102C
1110
1103
1103A
1103B
1103C
1112
IP1
IP2
1160
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
IP PACKET HEADER
SOURCE IP ADDRESS: 71
DEST. IP ADDRESS: 91
DISCRIM FIELD: 45
PAYLOAD #3
1104
1104A
1104B
1105
1105A
1105B
1105C
1113
IP3

FIG. 11

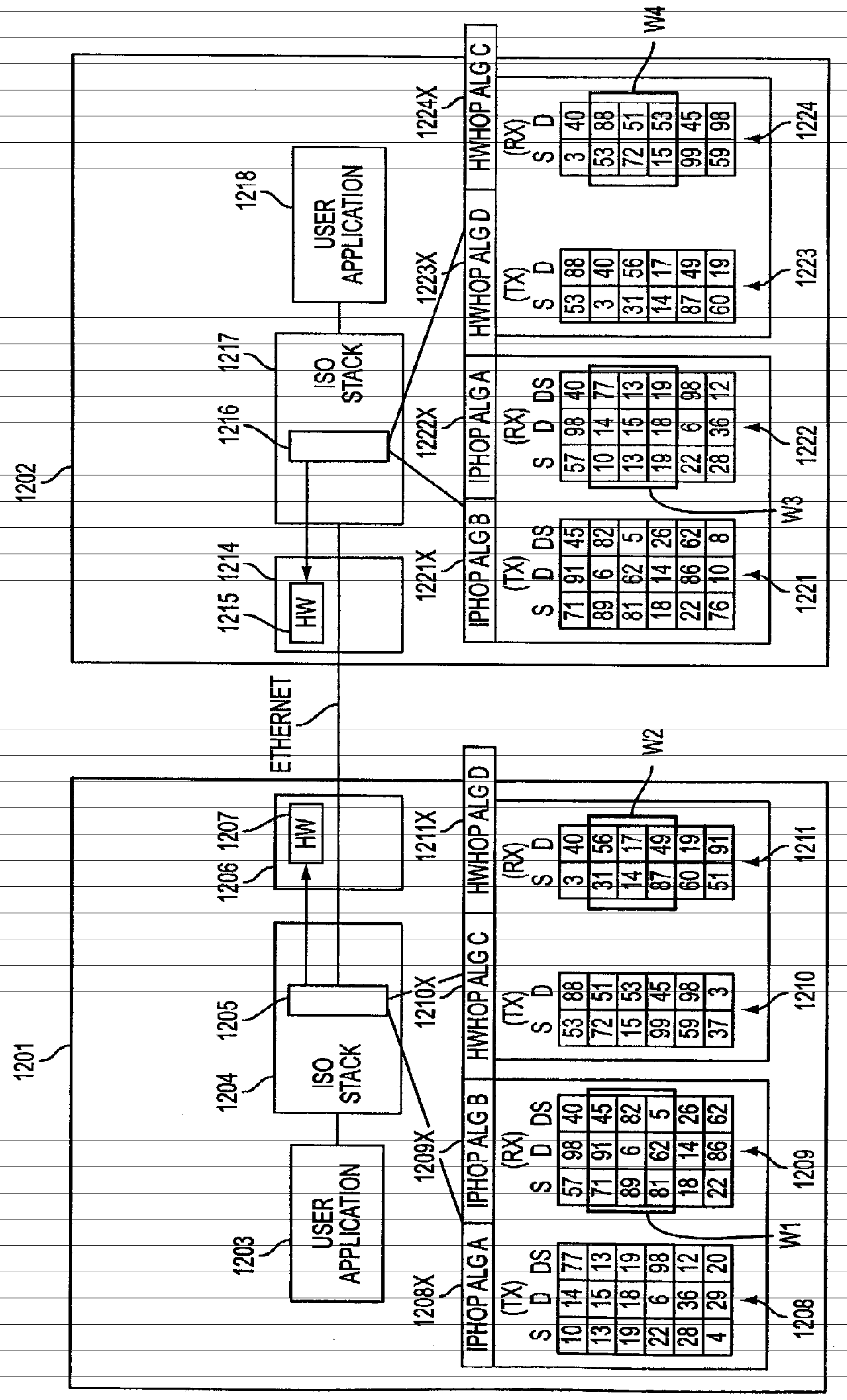

1201
1202
1203
USER APPLICATION
1204
ISO STACK
1205
1206
1207
HW
ETHERNET
1218
USER APPLICATION
1217
ISO STACK
1216
1214
1215
HW
IPHOP ALG A
IPHOP ALG B
HWHOP ALG C
HWHOP ALG D
1208X
1209X
1210X
1211X
(TX) S D DS
10 14 77
13 15 13
19 18 19
22 6 98
28 36 12
4 29 20
1208
(RX) S D DS
57 98 40
71 91 45
89 6 82
81 62 5
18 14 26
22 86 62
1209
W1
(TX) S D
53 88
72 51
15 53
99 45
59 98
37 3
1210
(RX) S D
3 40
31 56
14 17
87 49
60 19
51 91
1211
W2
IPHOP ALG B
IPHOP ALG A
HWHOP ALG D
HWHOP ALG C
1221X
1222X
1223X
1224X
(TX) S D DS
71 91 45
89 6 82
81 62 5
18 14 26
22 86 62
76 10 8
1221
(RX) S D DS
57 98 40
10 14 77
13 15 13
19 18 19
22 6 98
28 36 12
1222
W3
(TX) S D
53 88
3 40
31 56
14 17
87 49
60 19
1223
(RX) S D
3 40
53 88
72 51
15 53
99 45
59 98
1224
W4

FIG. 12A

MODE OR EMBODIMENT	HARDWARE ADDRESSES	IP ADDRESSES	DISCRIMINATOR FIELD VALUES
1. PROMISCUOUS	SAME FOR ALL NODES OR COMPLETELY RANDOM	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
2. PROMISCUOUS PER VPN	FIXED FOR EACH VPN	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
3. HARDWARE HOPPING	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC

FIG. 12B

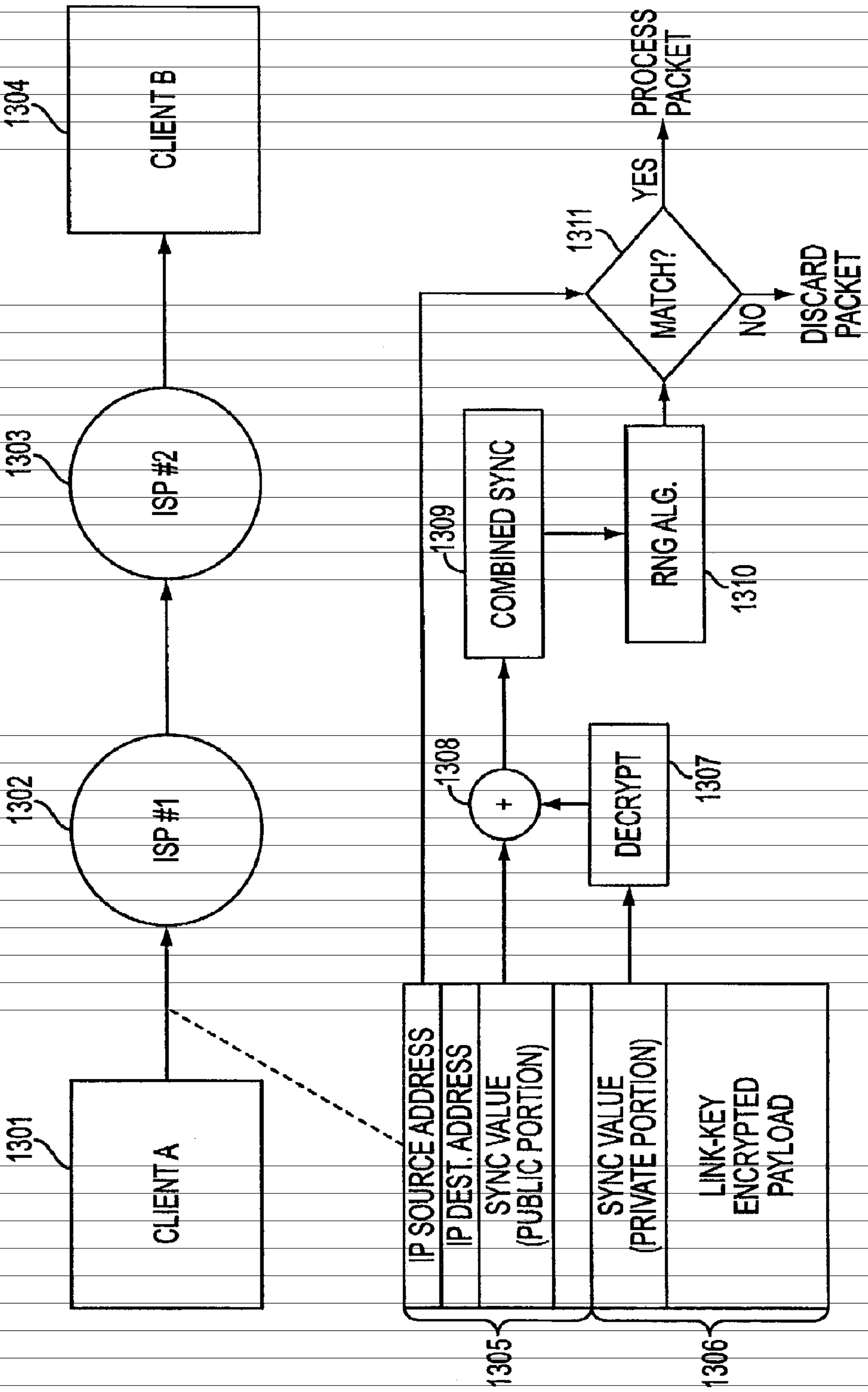
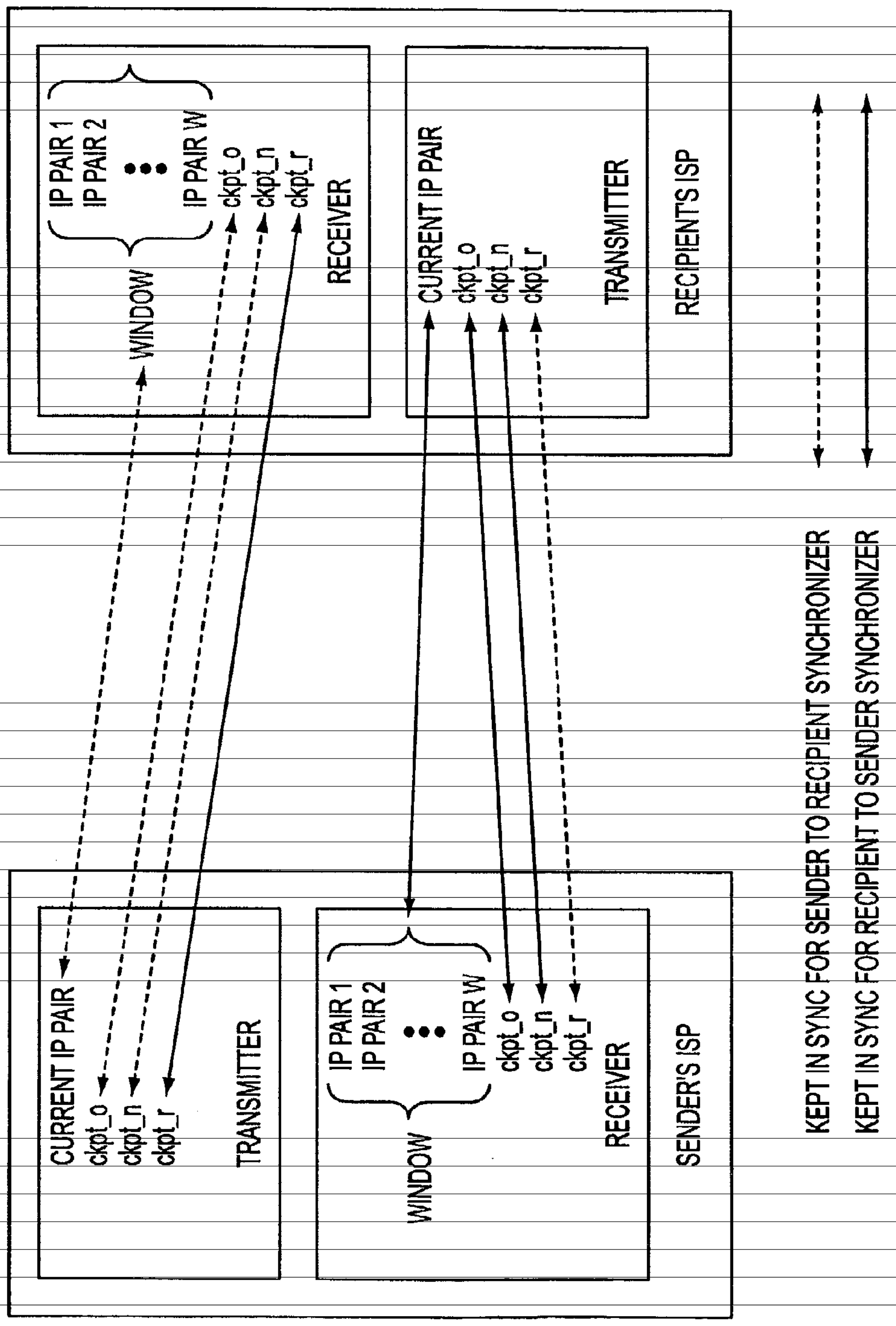

1301
CLIENT A
1302
ISP #1
1303
ISP #2
1304
CLIENT B
IP SOURCE ADDRESS
IP DEST. ADDRESS
SYNC VALUE (PUBLIC PORTION)
SYNC VALUE (PRIVATE PORTION)
LINK-KEY ENCRYPTED PAYLOAD
1305
1306
DECRYPT
1307
+
1308
COMBINED SYNC
1309
RNG ALG.
1310
MATCH?
1311
YES
PROCESS PACKET
NO
DISCARD PACKET

FIG. 13



RECIPIENT'S ISP
RECEIVER
WINDOW
IP PAIR 1
IP PAIR 2
IP PAIR W
ckpt_o
ckpt_n
ckpt_r
TRANSMITTER
CURRENT IP PAIR
ckpt_o
ckpt_n
ckpt_r
SENDER'S ISP
TRANSMITTER
CURRENT IP PAIR
ckpt_o
ckpt_n
ckpt_r
RECEIVER
WINDOW
IP PAIR 1
IP PAIR 2
IP PAIR W
ckpt_o
ckpt_n
ckpt_r
KEPT IN SYNC FOR SENDER TO RECIPIENT SYNCHRONIZER
KEPT IN SYNC FOR RECIPIENT TO SENDER SYNCHRONIZER

FIG. 14

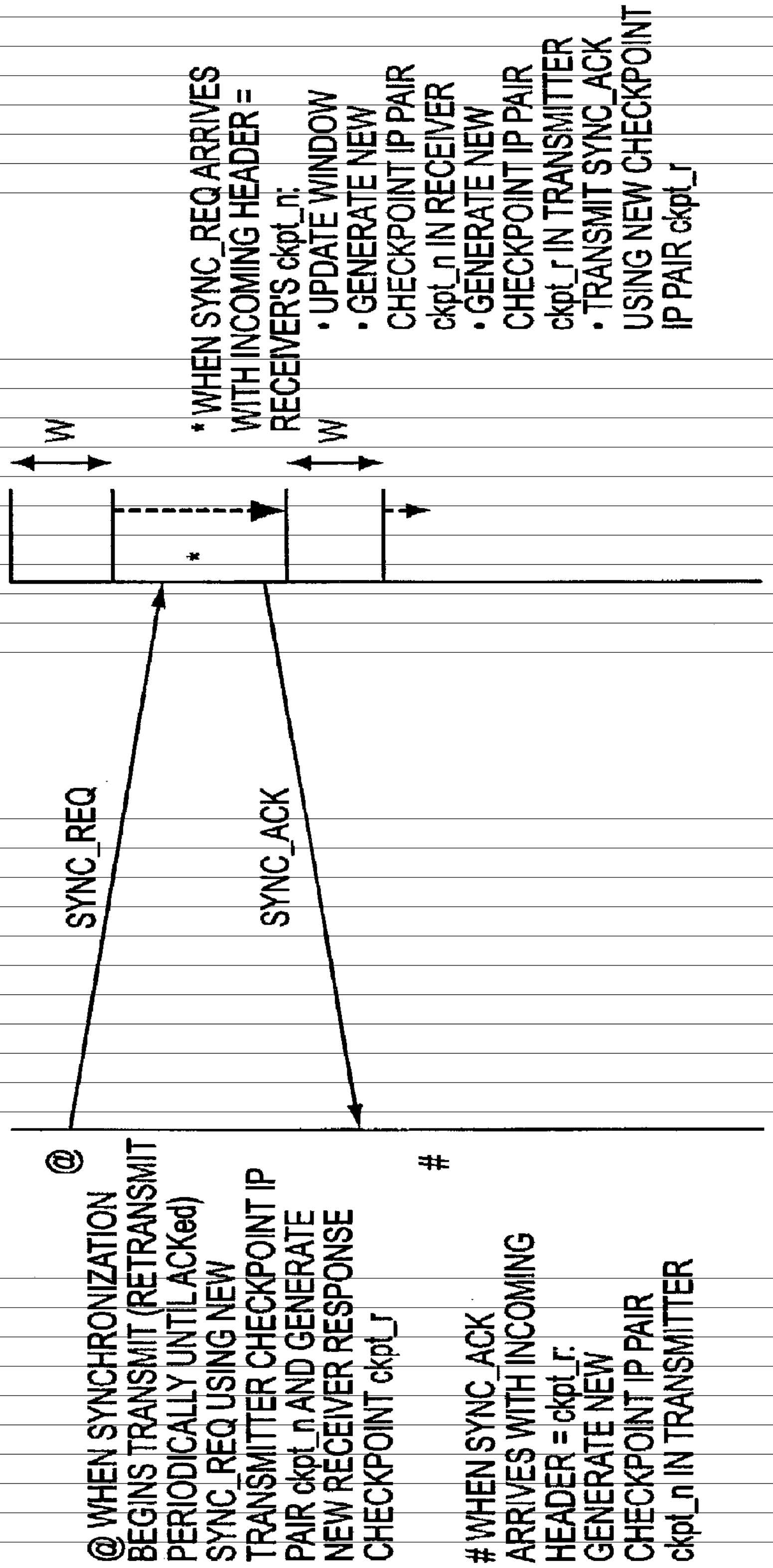

@ WHEN SYNCHRONIZATION BEGINS TRANSMIT (RETRANSMIT PERIODICALLY UNTIL ACKed) SYNC_REQ USING NEW TRANSMITTER CHECKPOINT IP PAIR ckpt_n AND GENERATE NEW RECEIVER RESPONSE CHECKPOINT ckpt_r
# WHEN SYNC_ACK ARRIVES WITH INCOMING HEADER = ckpt_r: GENERATE NEW CHECKPOINT IP PAIR ckpt_n IN TRANSMITTER
@
#
SYNC_REQ
SYNC_ACK
W
W
*
* WHEN SYNC_REQ ARRIVES WITH INCOMING HEADER = RECEIVER'S ckpt_n:
• UPDATE WINDOW
• GENERATE NEW CHECKPOINT IP PAIR ckpt_n IN RECEIVER
• GENERATE NEW CHECKPOINT IP PAIR ckpt_r IN TRANSMITTER
• TRANSMIT SYNC_ACK USING NEW CHECKPOINT IP PAIR ckpt_r

FIG. 15

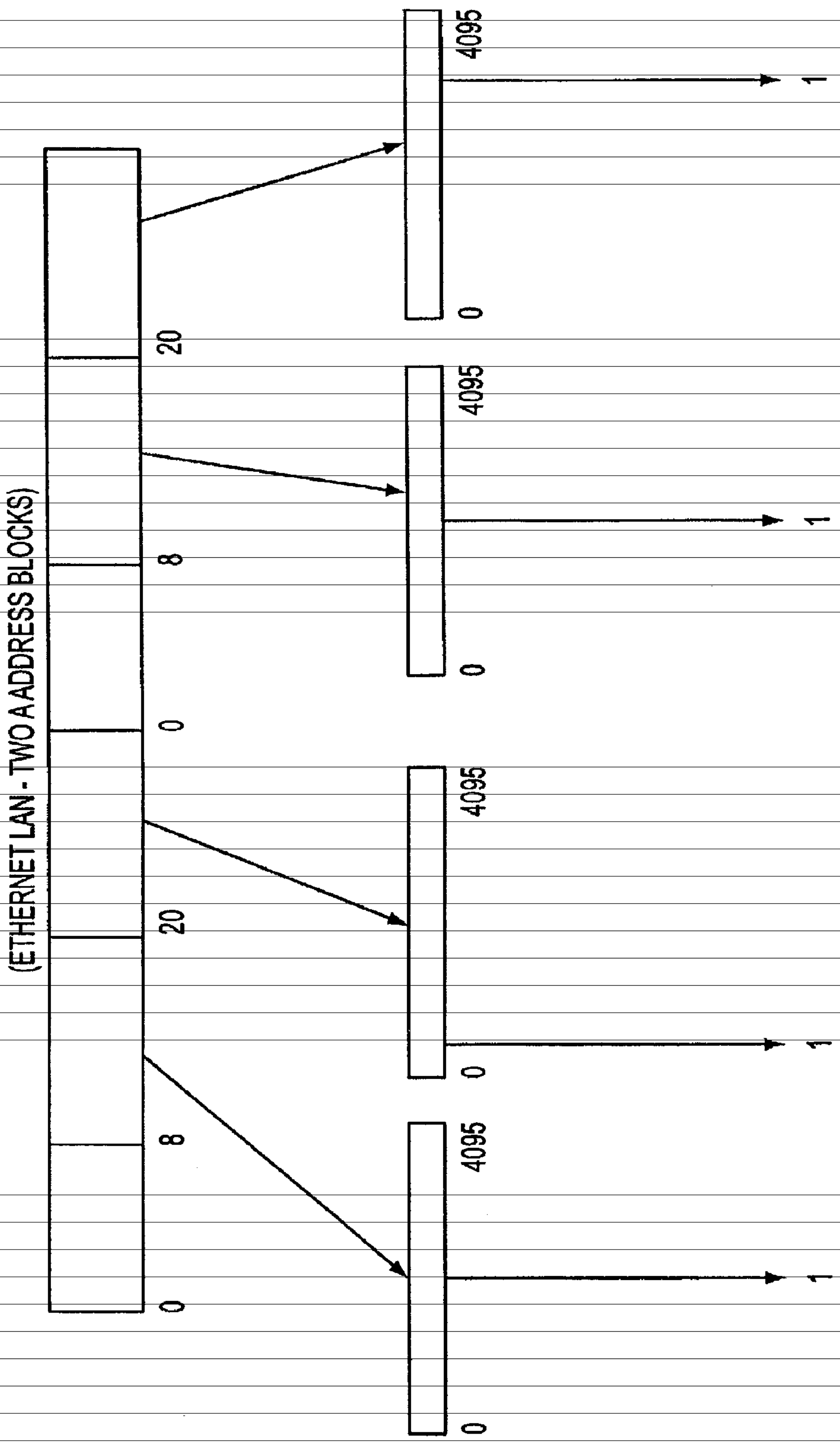

(ETHERNET LAN - TWO A ADDRESS BLOCKS)
0
8
20
0
8
20
0
4095
1
0
4095
1
0
4095
1
0
4095
1

FIG. 16

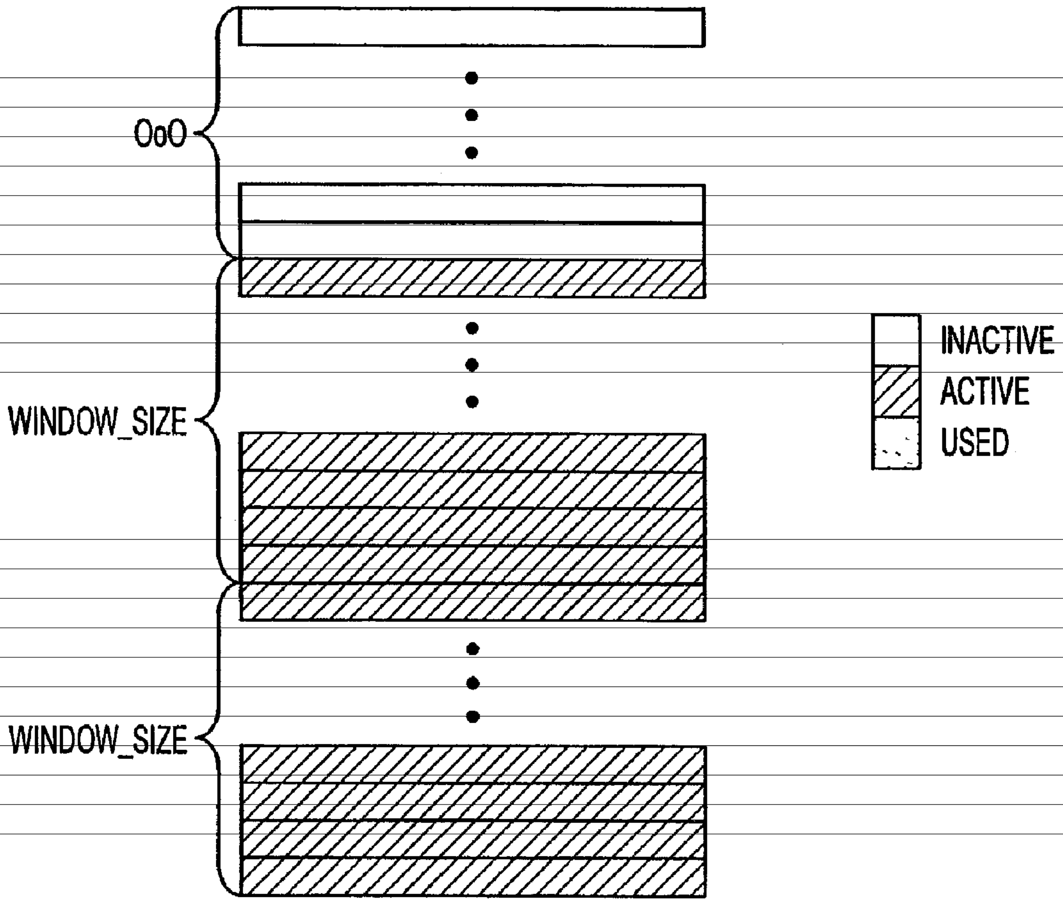


FIG. 17

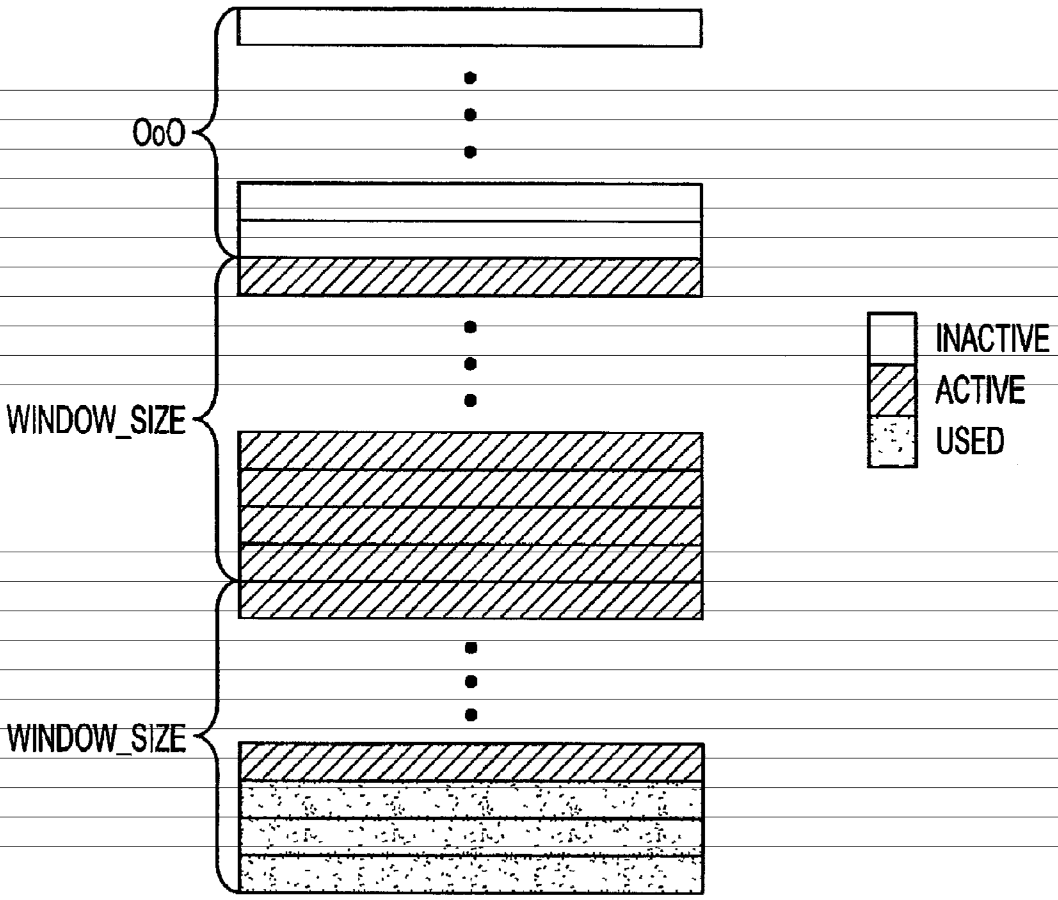


FIG. 18

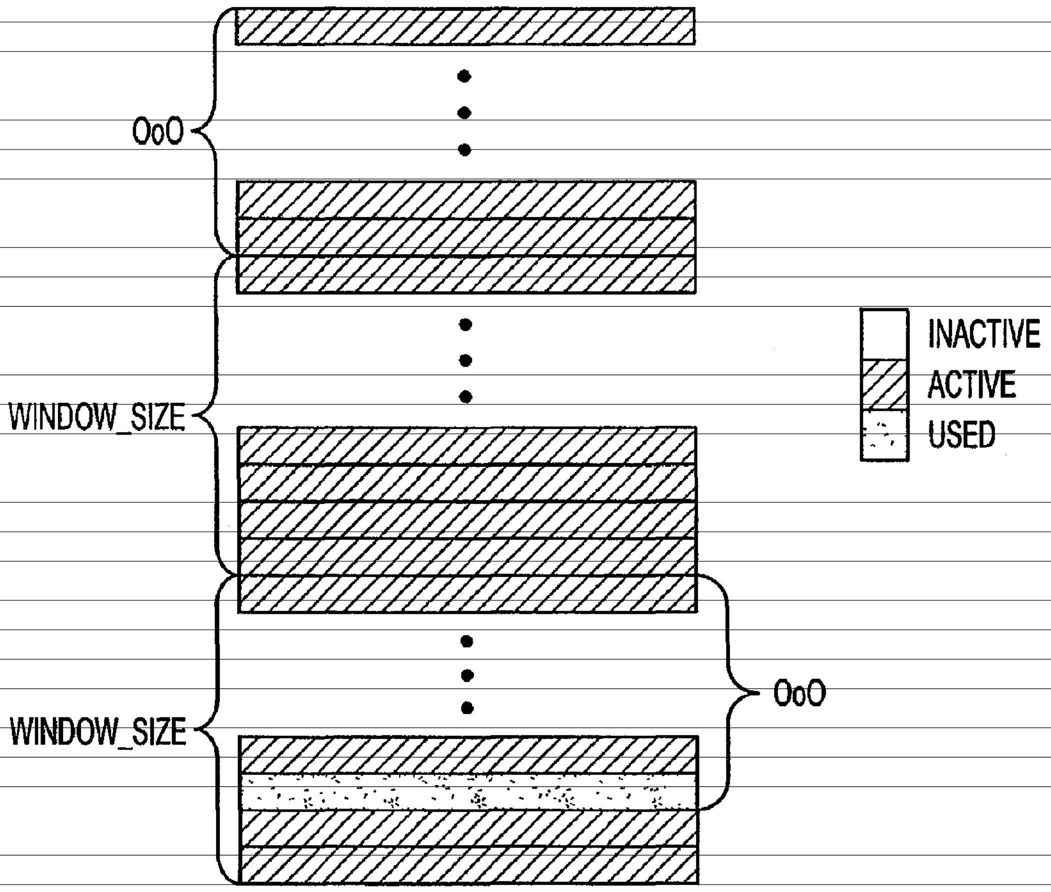


FIG. 19

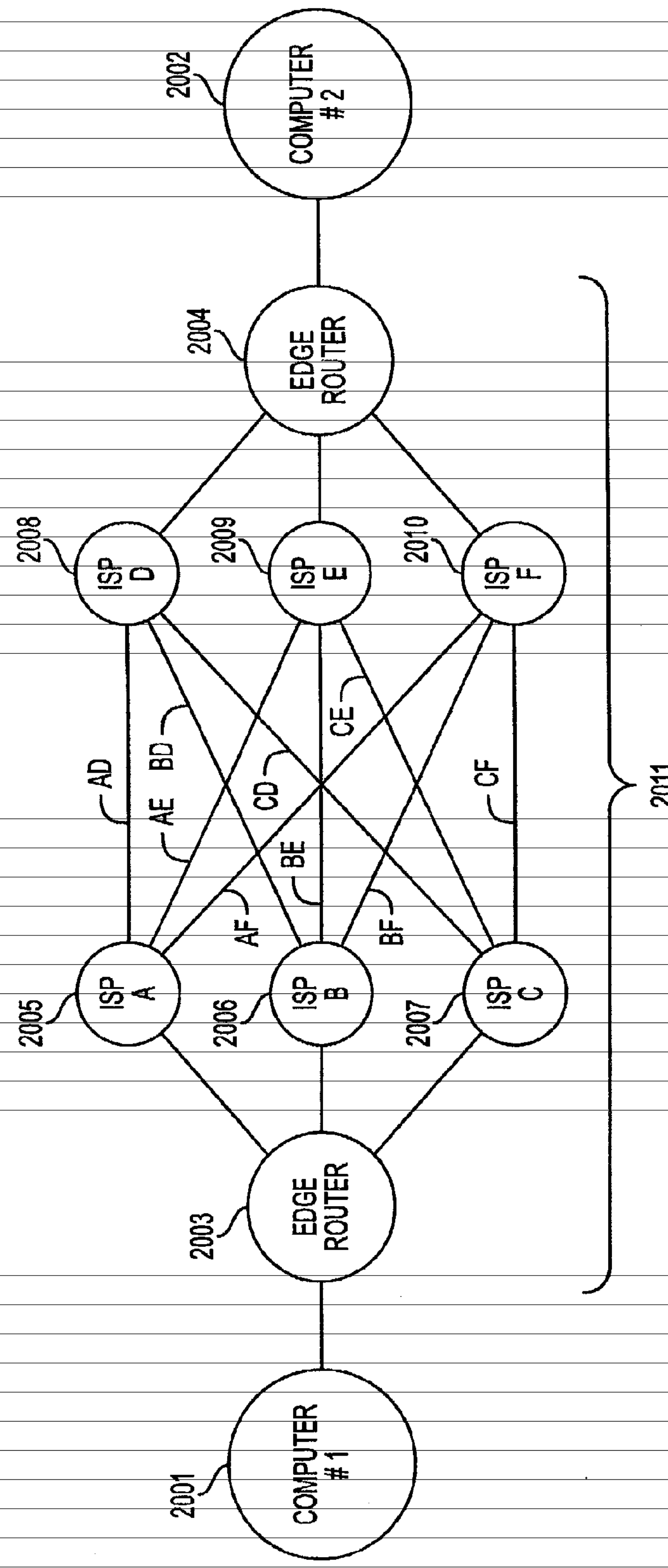
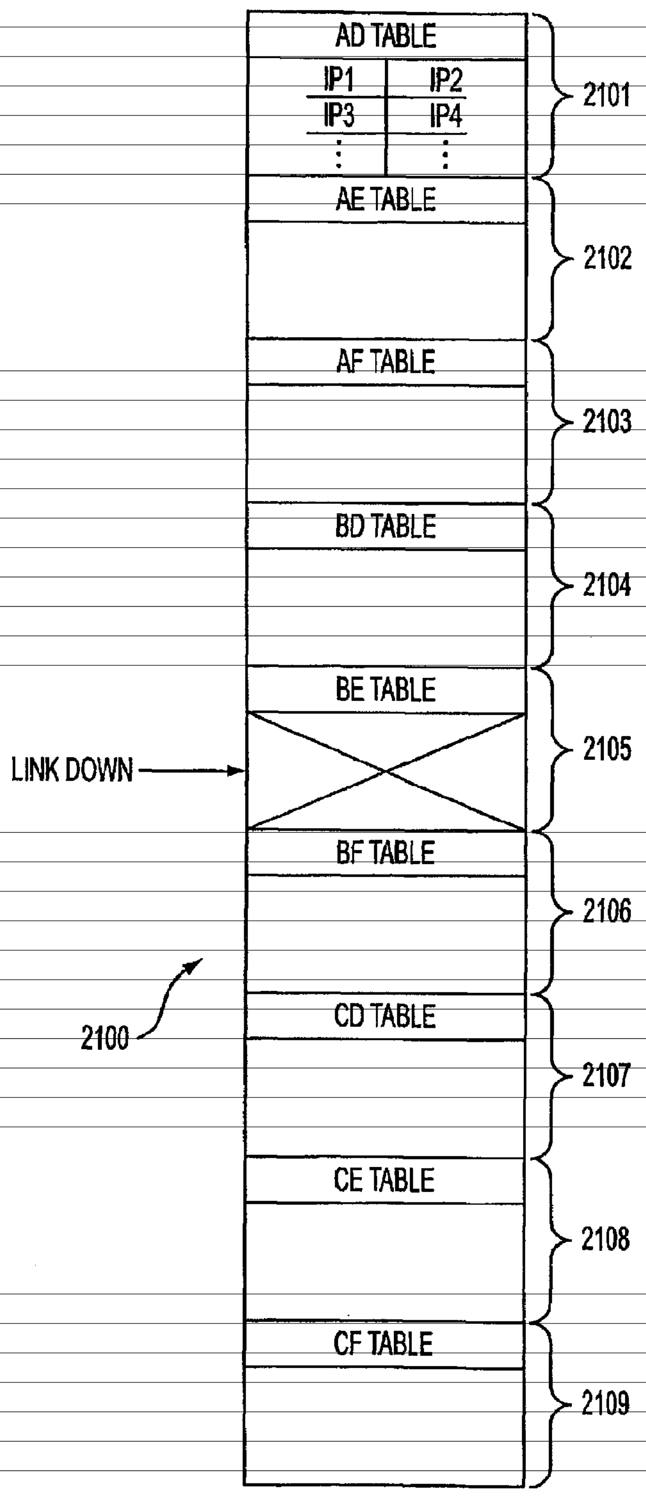

COMPUTER #1
2001
EDGE ROUTER
2003
ISP A
2005
ISP B
2006
ISP C
2007
AD
AE
AF
BD
BE
BF
CD
CE
CF
ISP D
2008
ISP E
2009
ISP F
2010
EDGE ROUTER
2004
COMPUTER #2
2002
2011

FIG. 20



AD TABLE
IP1
IP2
IP3
IP4
2101
AE TABLE
2102
AF TABLE
2103
BD TABLE
2104
BE TABLE
2105
LINK DOWN
BF TABLE
2106
2100
CD TABLE
2107
CE TABLE
2108
CF TABLE
2109

FIG. 21

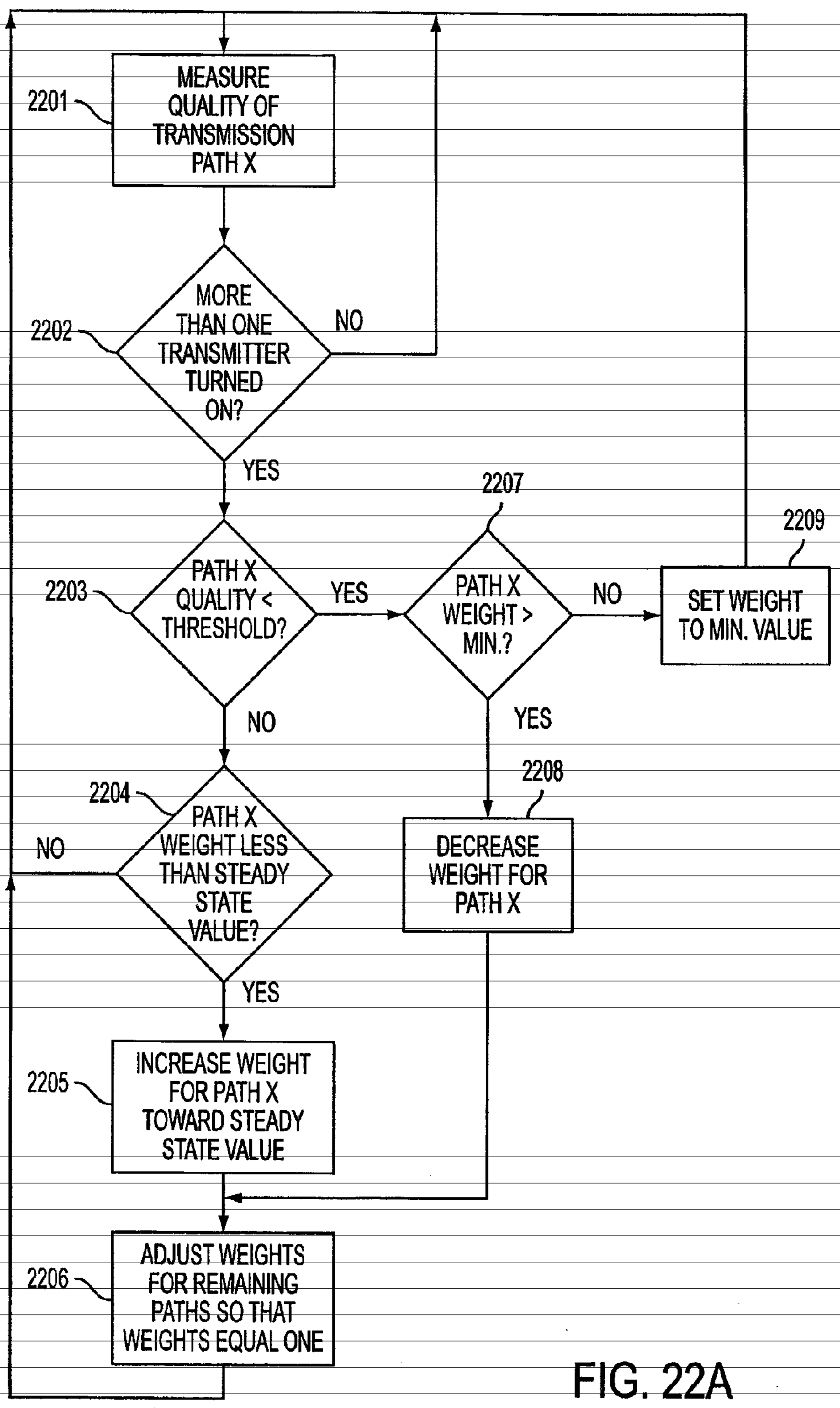

2201
MEASURE QUALITY OF TRANSMISSION PATH X
2202
MORE THAN ONE TRANSMITTER TURNED ON?
NO
YES
2203
PATH X QUALITY < THRESHOLD?
YES
NO
2207
PATH X WEIGHT > MIN.?
NO
YES
2209
SET WEIGHT TO MIN. VALUE
2204
PATH X WEIGHT LESS THAN STEADY STATE VALUE?
NO
YES
2208
DECREASE WEIGHT FOR PATH X
2205
INCREASE WEIGHT FOR PATH X TOWARD STEADY STATE VALUE
2206
ADJUST WEIGHTS FOR REMAINING PATHS SO THAT WEIGHTS EQUAL ONE

FIG. 22A

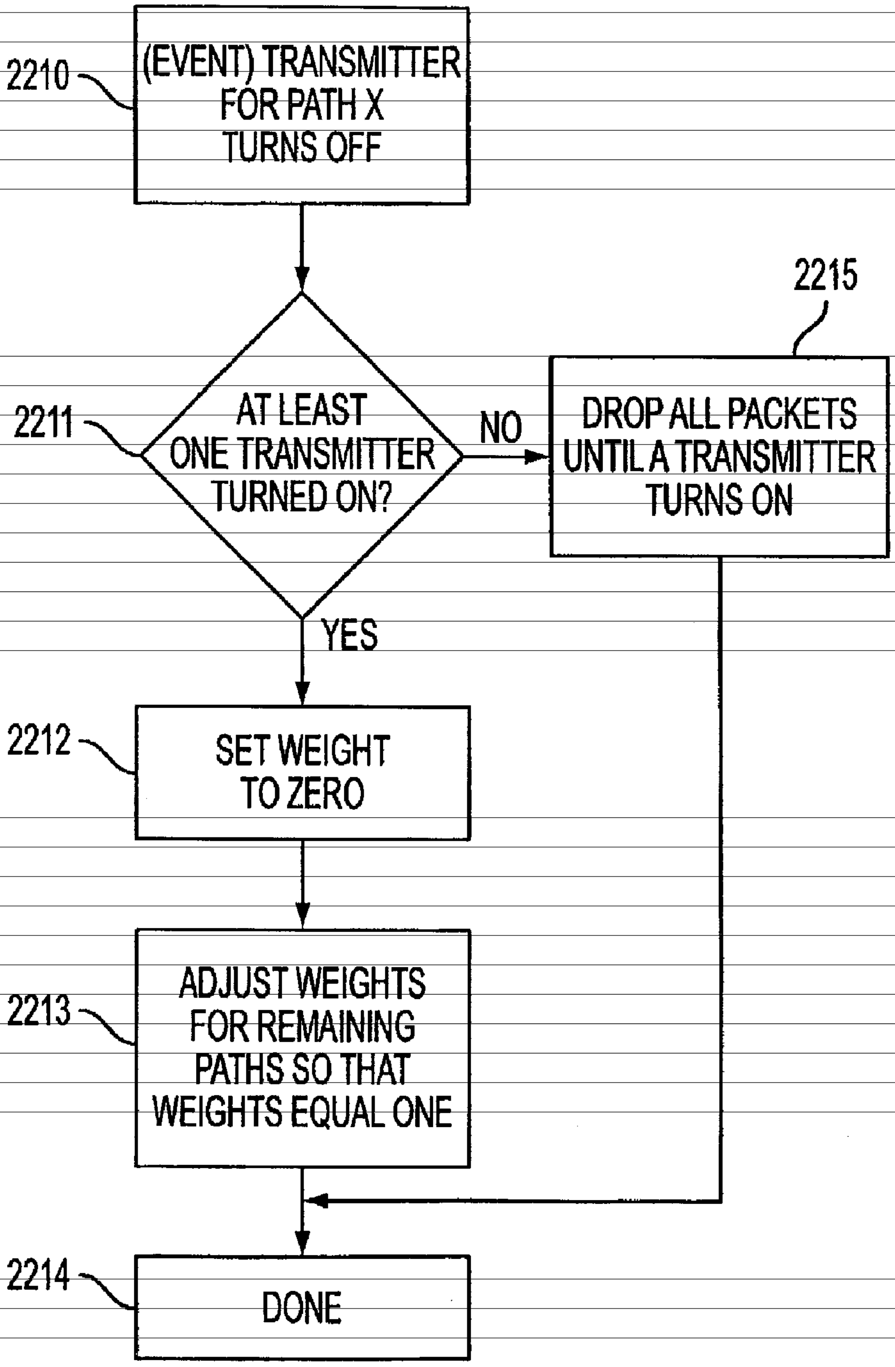

2210
(EVENT) TRANSMITTER FOR PATH X TURNS OFF
2211
AT LEAST ONE TRANSMITTER TURNED ON?
NO
YES
2215
DROP ALL PACKETS UNTIL A TRANSMITTER TURNS ON
2212
SET WEIGHT TO ZERO
2213
ADJUST WEIGHTS FOR REMAINING PATHS SO THAT WEIGHTS EQUAL ONE
2214
DONE

FIG. 22B

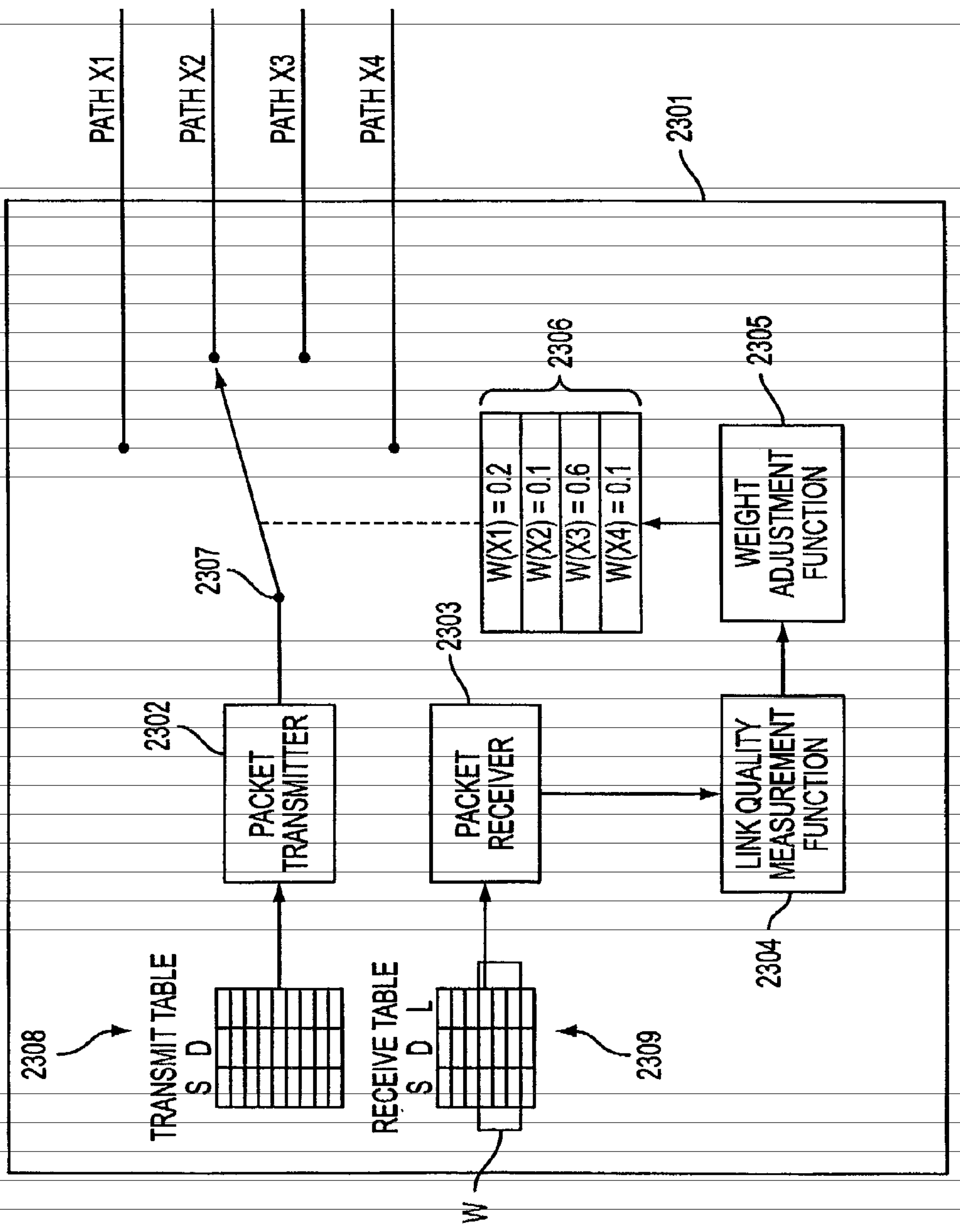

PATH X1
PATH X2
PATH X3
PATH X4
2301
2306
W(X1) = 0.2
W(X2) = 0.1
W(X3) = 0.6
W(X4) = 0.1
2305
WEIGHT ADJUSTMENT FUNCTION
2307
2302
PACKET TRANSMITTER
2303
PACKET RECEIVER
LINK QUALITY MEASUREMENT FUNCTION
2304
2308
TRANSMIT TABLE
S D
RECEIVE TABLE
S D L
2309
W

FIG. 23

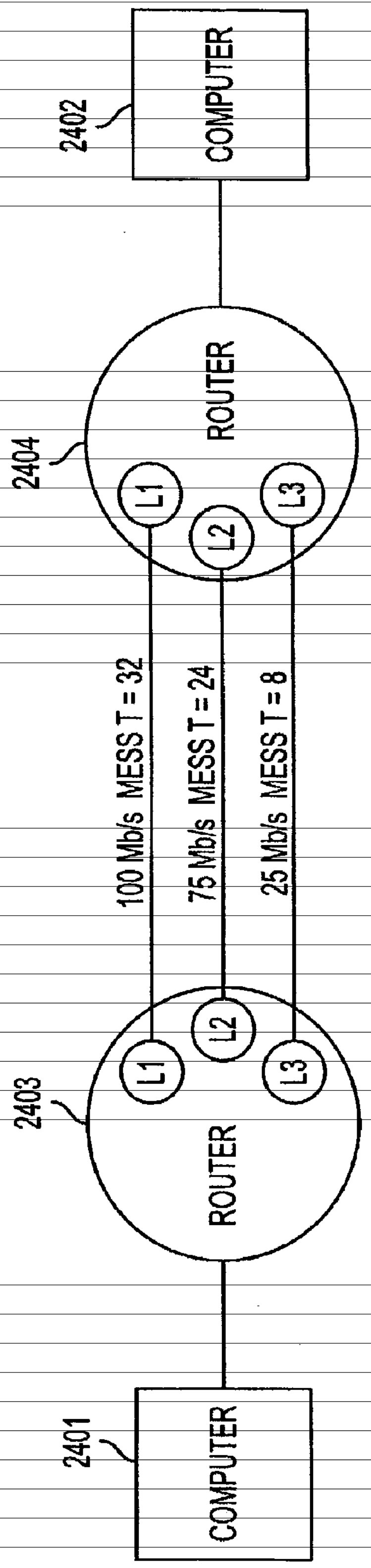

2402
COMPUTER
2404
ROUTER
L1
L2
L3
100 Mb/s MESS T = 32
75 Mb/s MESS T = 24
25 Mb/s MESS T = 8
2403
ROUTER
L1
L2
L3
2401
COMPUTER

FIG. 24

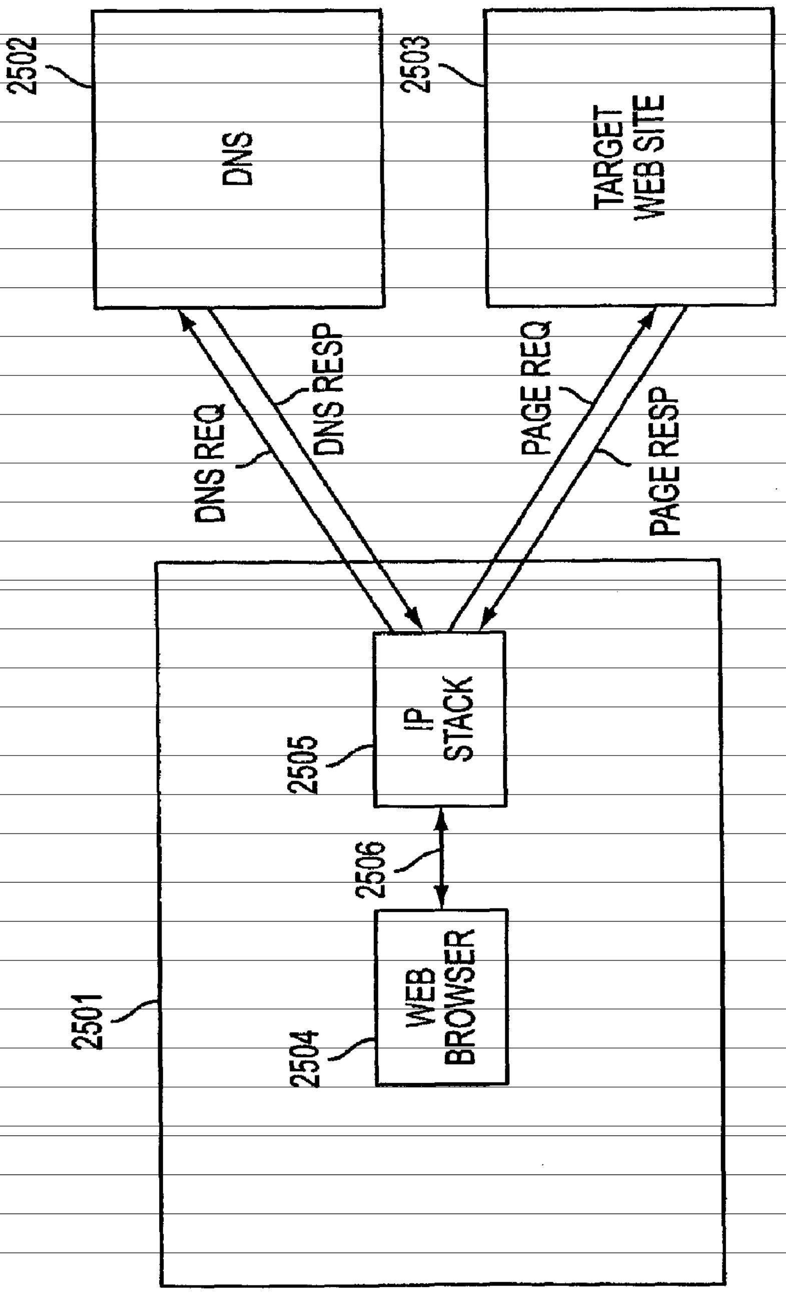


FIG. 25
(PRIOR ART)

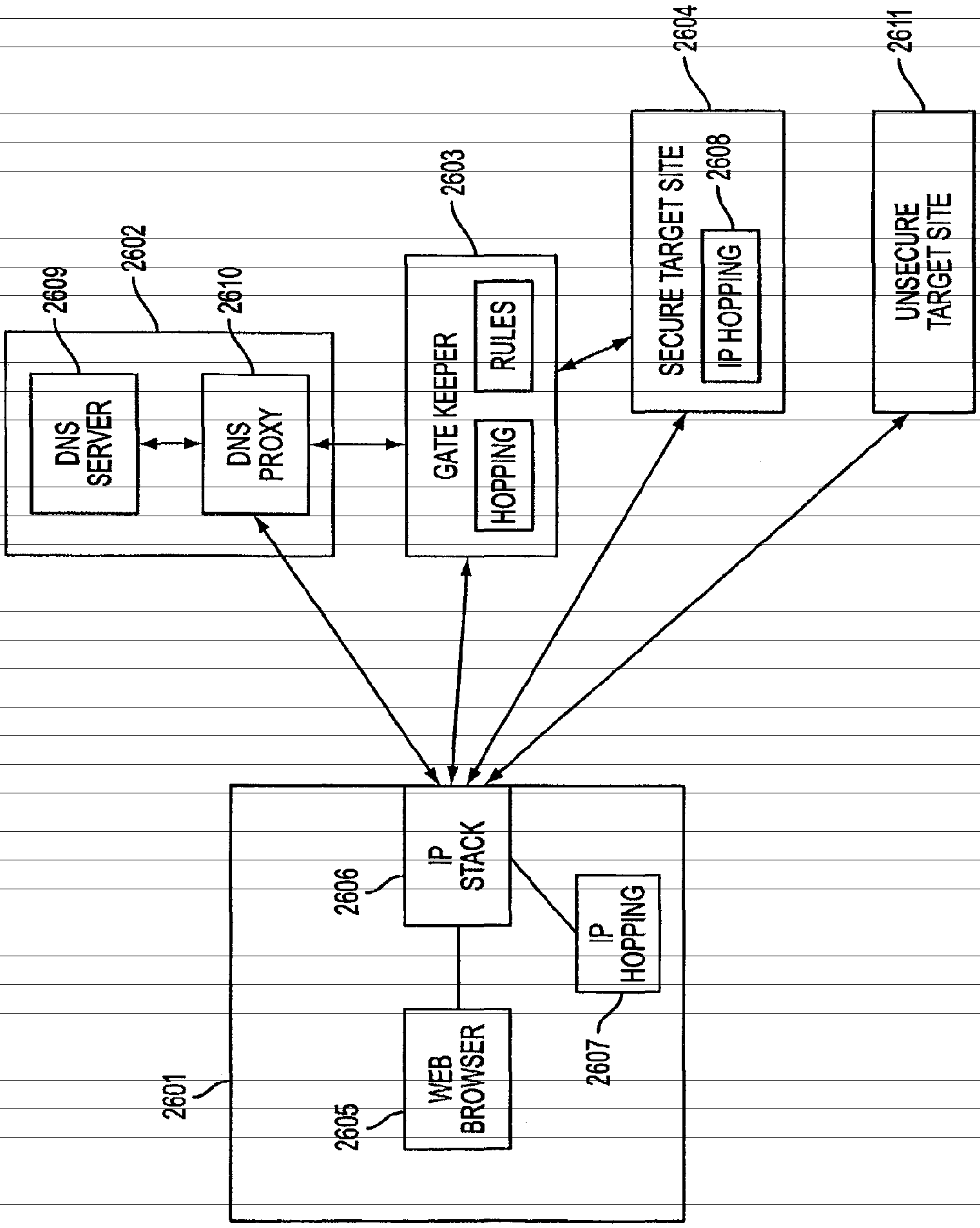

2602
2609
DNS SERVER
2610
DNS PROXY
2603
GATE KEEPER
HOPPING
RULES
2604
SECURE TARGET SITE
2608
IP HOPPING
2611
UNSECURE TARGET SITE
2601
2606
IP STACK
IP HOPPING
2607
2605
WEB BROWSER

FIG. 26

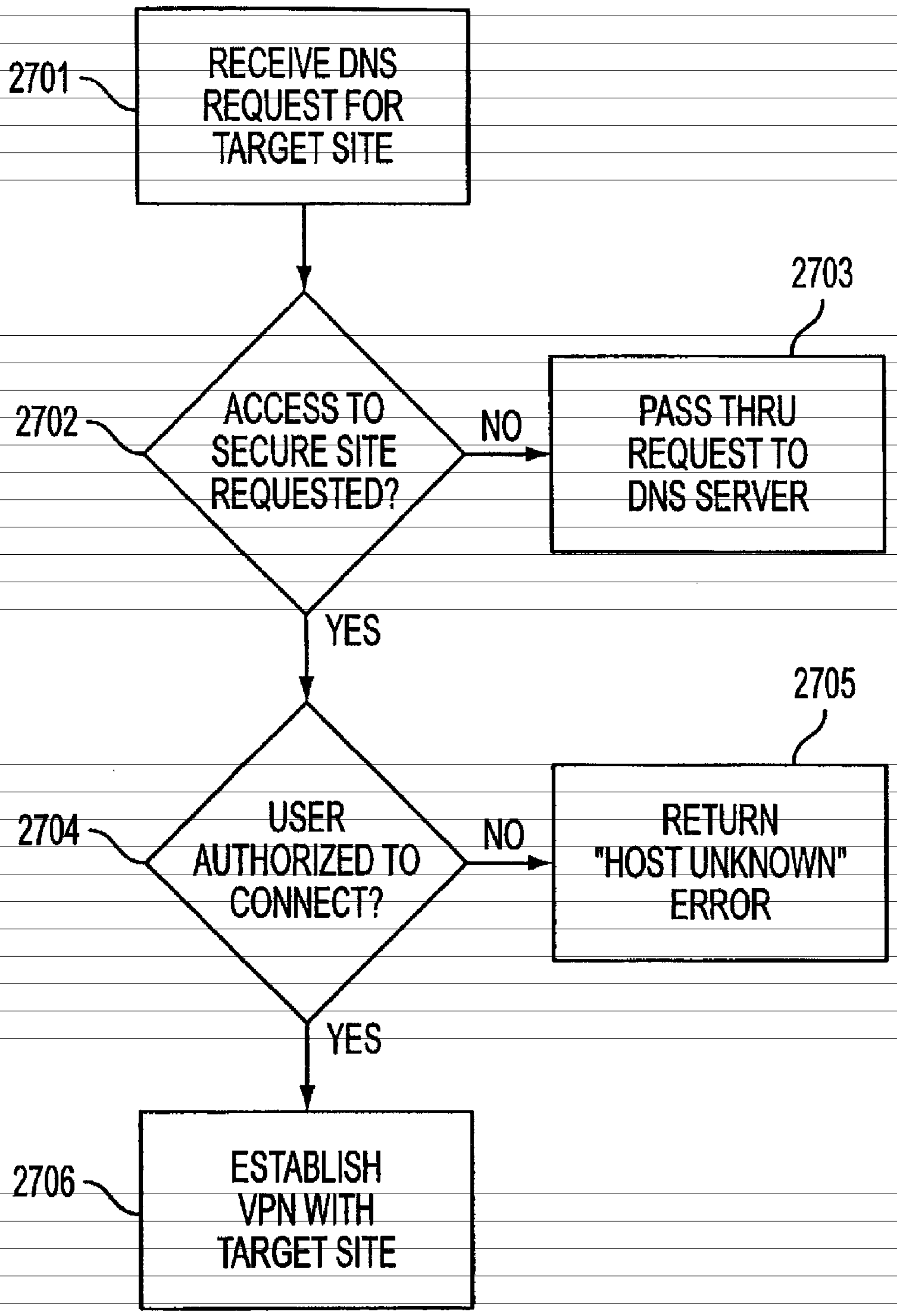

2701
RECEIVE DNS REQUEST FOR TARGET SITE
2702
ACCESS TO SECURE SITE REQUESTED?
NO
2703
PASS THRU REQUEST TO DNS SERVER
YES
2704
USER AUTHORIZED TO CONNECT?
NO
2705
RETURN "HOST UNKNOWN" ERROR
YES
2706
ESTABLISH VPN WITH TARGET SITE

FIG. 27

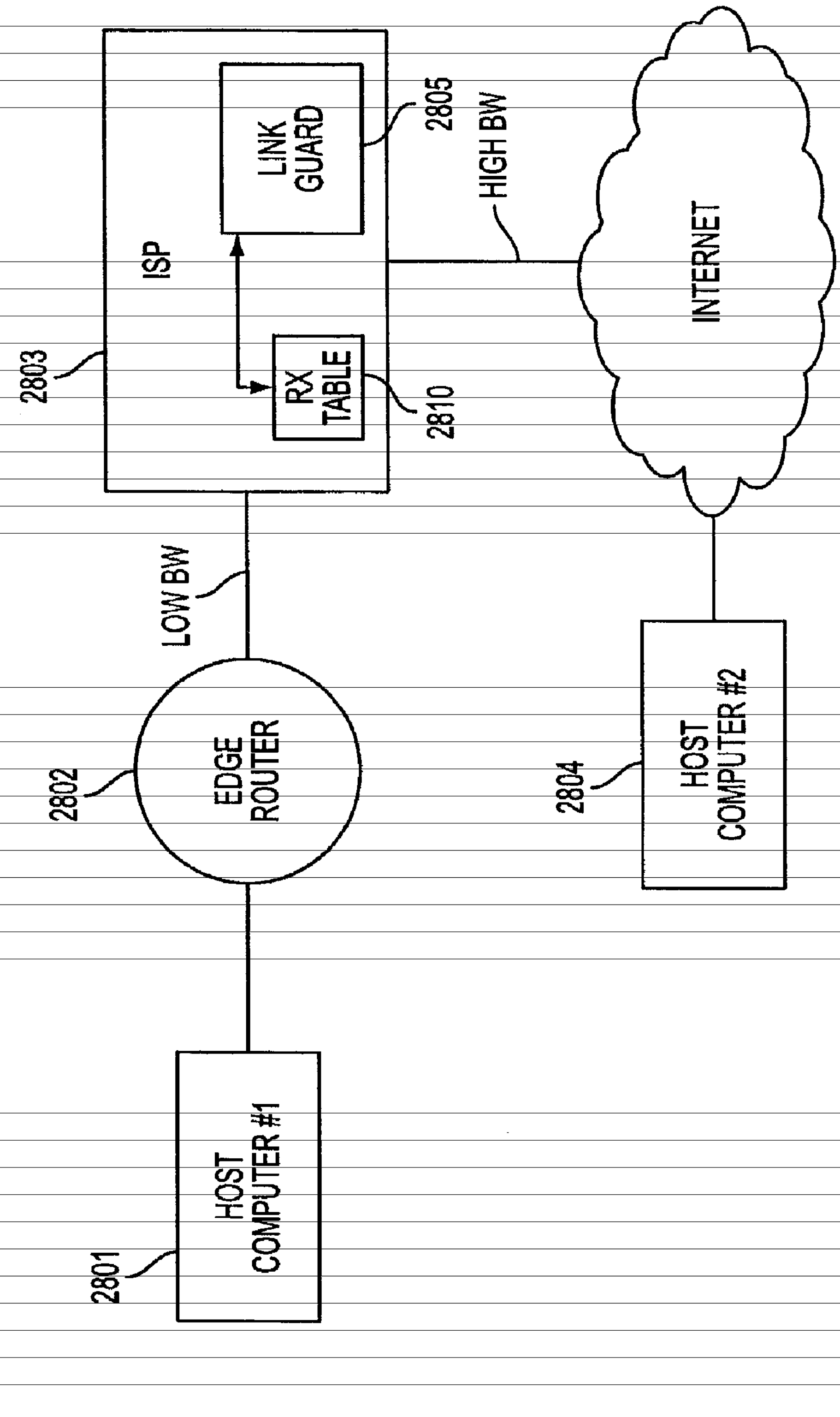
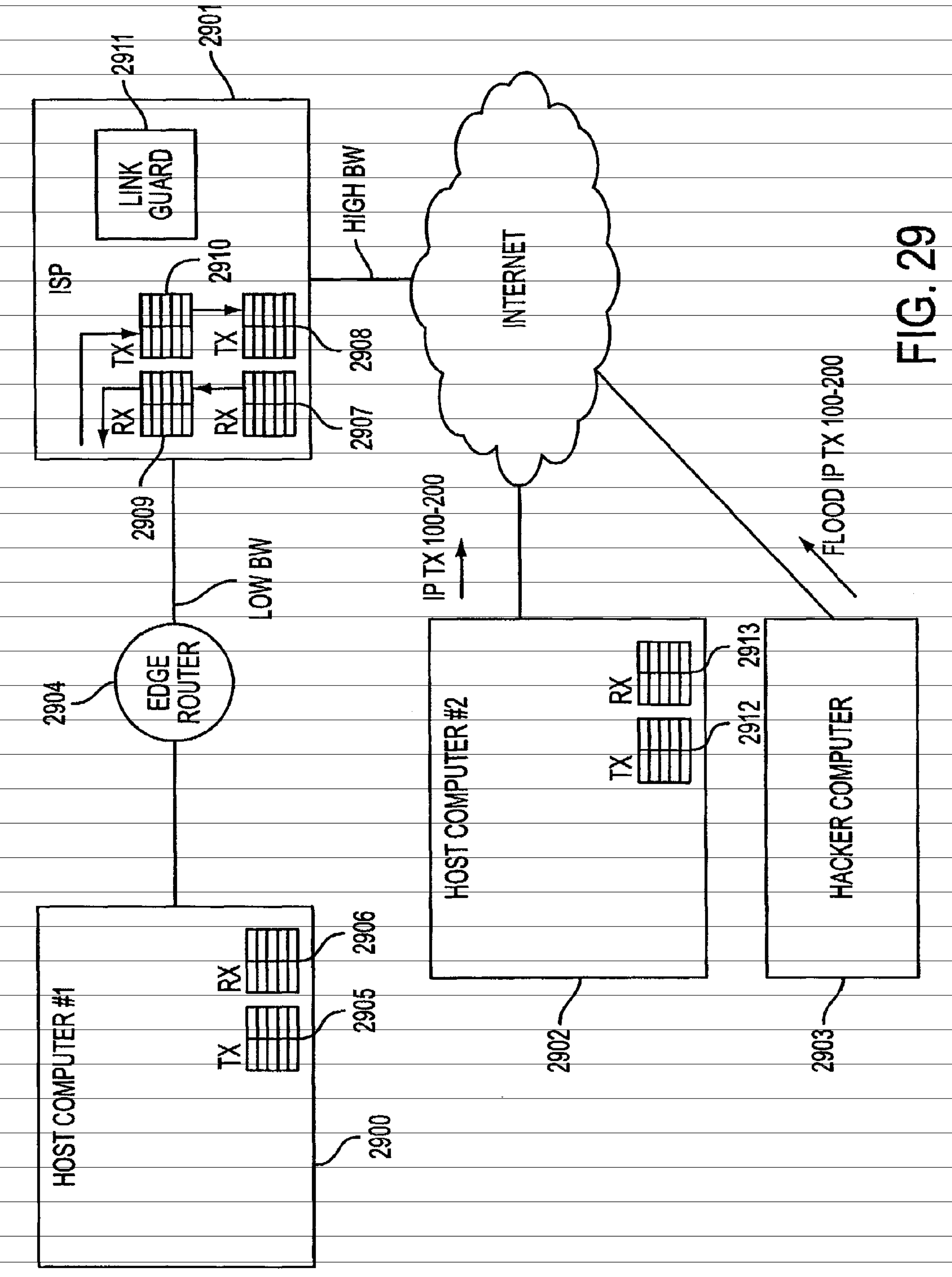

ISP
LINK GUARD
2805
HIGH BW
INTERNET
2803
RX TABLE
2810
LOW BW
EDGE ROUTER
2802
HOST COMPUTER #2
2804
HOST COMPUTER #1
2801

FIG. 28



ISP
2901
2911
LINK GUARD
2910
TX
TX
RX
RX
2908
2907
2909
HIGH BW
INTERNET
LOW BW
EDGE ROUTER
2904
IP TX 100-200
FLOOD IP TX 100-200
HOST COMPUTER #2
RX
TX
2913
2912
HACKER COMPUTER
2902
2903
HOST COMPUTER #1
RX
TX
2906
2905
2900

FIG. 29

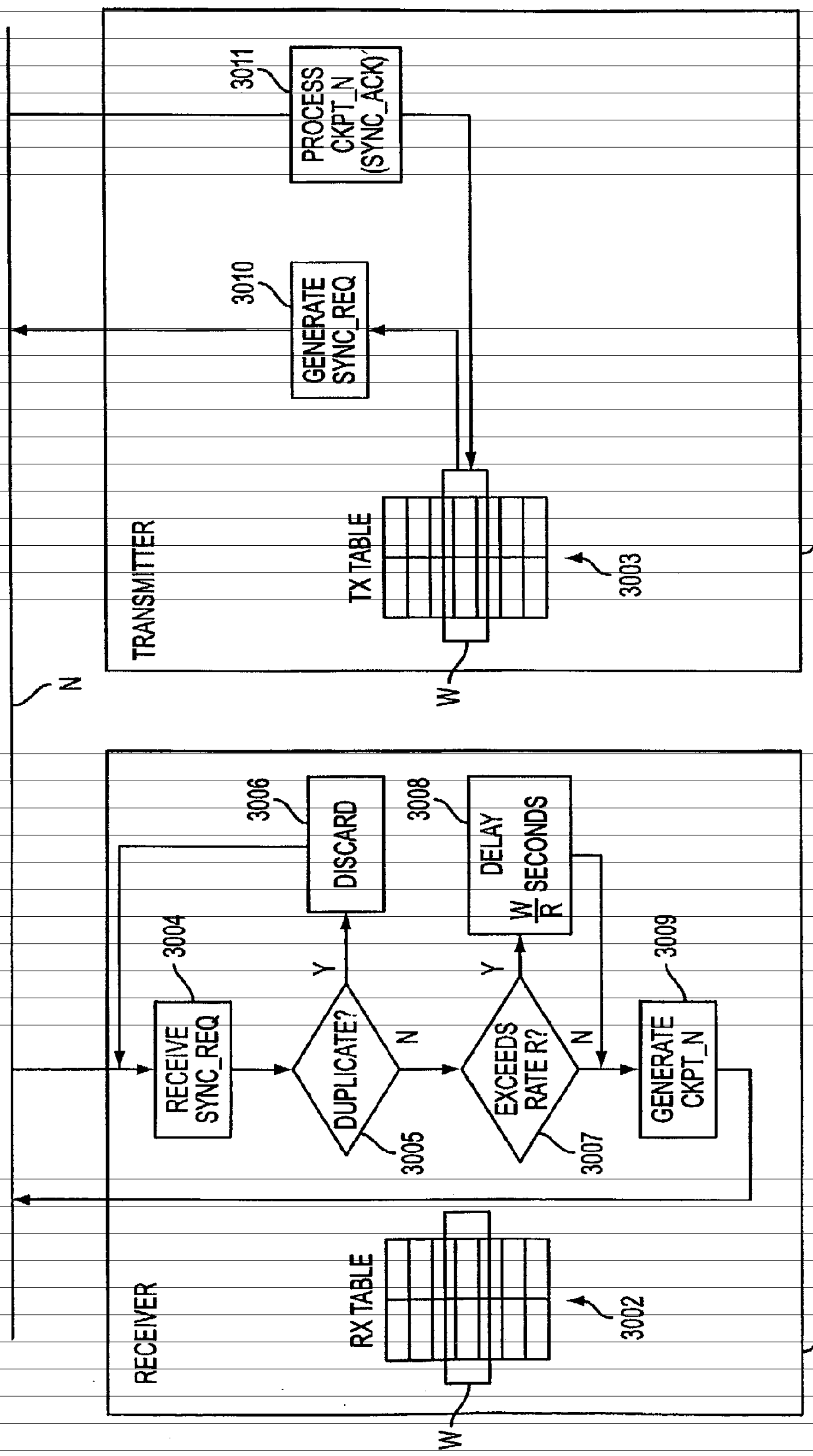

TRANSMITTER
3001
TX TABLE
3003
W
3010
GENERATE SYNC_REQ
3011
PROCESS CKPT_N (SYNC_ACK)
N
RECEIVER
3000
RX TABLE
3002
W
3004
RECEIVE SYNC_REQ
3005
DUPLICATE?
Y
N
3006
DISCARD
3007
EXCEEDS RATE R?
Y
N
3008
DELAY W/R SECONDS
3009
GENERATE CKPT_N

FIG. 30

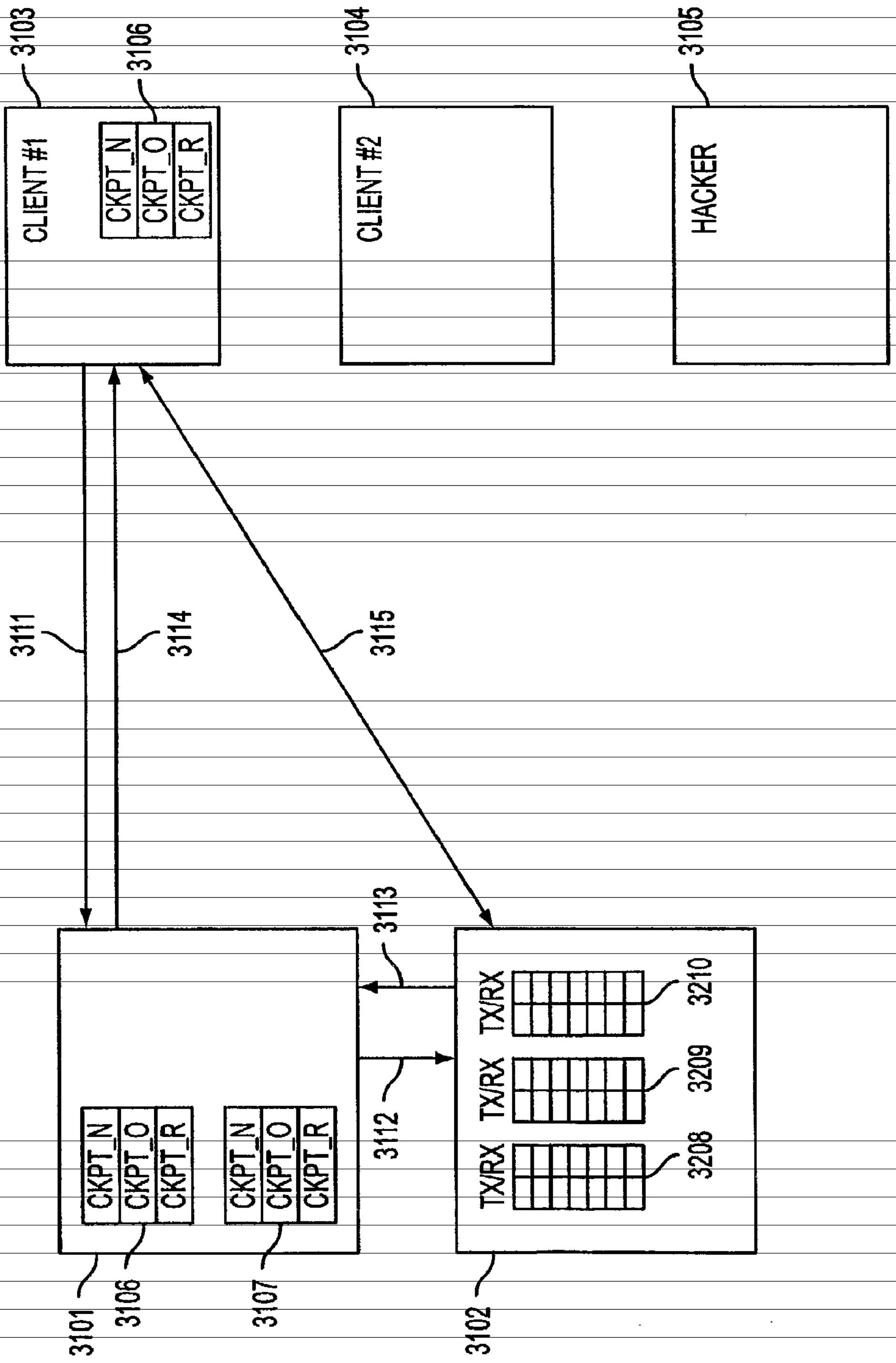

3103
CLIENT #1
3106
CKPT_N
CKPT_O
CKPT_R
3104
CLIENT #2
3105
HACKER
3111
3114
3115
3113
3112
3101
3106
CKPT_N
CKPT_O
CKPT_R
3107
CKPT_N
CKPT_O
CKPT_R
3102
TX/RX
3208
TX/RX
3209
TX/RX
3210

FIG. 31

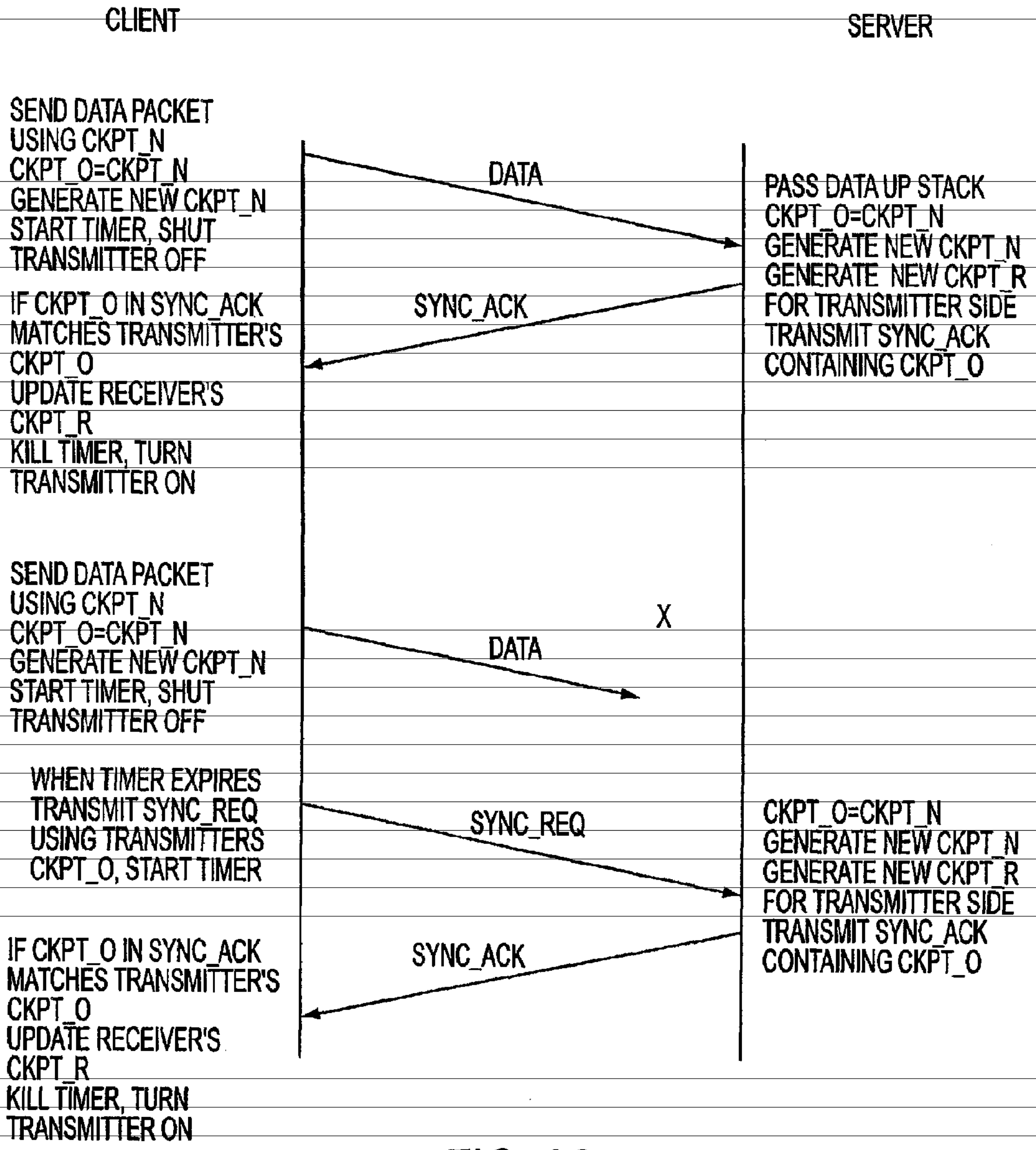

CLIENT
SERVER
SEND DATA PACKET USING CKPT_N CKPT_O=CKPT_N GENERATE NEW CKPT_N START TIMER, SHUT TRANSMITTER OFF
DATA
PASS DATA UP STACK CKPT_O=CKPT_N GENERATE NEW CKPT_N GENERATE NEW CKPT_R FOR TRANSMITTER SIDE TRANSMIT SYNC_ACK CONTAINING CKPT_O
SYNC_ACK
IF CKPT_O IN SYNC_ACK MATCHES TRANSMITTER'S CKPT_O UPDATE RECEIVER'S CKPT_R KILL TIMER, TURN TRANSMITTER ON
SEND DATA PACKET USING CKPT_N CKPT_O=CKPT_N GENERATE NEW CKPT_N START TIMER, SHUT TRANSMITTER OFF
X
DATA
WHEN TIMER EXPIRES TRANSMIT SYNC_REQ USING TRANSMITTERS CKPT_O, START TIMER
SYNC_REQ
CKPT_O=CKPT_N GENERATE NEW CKPT_N GENERATE NEW CKPT_R FOR TRANSMITTER SIDE TRANSMIT SYNC_ACK CONTAINING CKPT_O
SYNC_ACK
IF CKPT_O IN SYNC_ACK MATCHES TRANSMITTER'S CKPT_O UPDATE RECEIVER'S CKPT_R KILL TIMER, TURN TRANSMITTER ON

FIG. 32

ESTABLISHMENT OF A SECURE COMMUNICATION LINK BASED ON A DOMAIN NAME SERVICE (DNS) REQUEST

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a divisional application of 09/504,783 (filed Feb. 15, 2000), now U.S. Pat. No. 6,502,135, issued Dec. 31, 2002, which claims priority from and is a continuation-in-part of previously filed U.S. application Ser. No. 09/429,643 (filed Oct. 29, 1999) now U.S. Pat. No. 7,010,604. The subject matter of the '643 application, which is bodily incorporated herein, derives from provisional U.S. application No. 60/106,261 (filed Oct. 30, 1998) and 60/137,704 (filed Jun. 7, 1999).

GOVERNMENT CONTRACT RIGHTS

This invention was made with Government support under Contract No. 360000-1999-000000-QC-000-000 awarded by the Central Intelligence Agency. The Government has certain rights in the invention.

BACKGROUND OF THE INVENTION

A tremendous variety of methods have been proposed and implemented to provide security and anonymity for communications over the Internet. The variety stems, in part, from the different needs of different Internet users. A basic heuristic framework to aid in discussing these different security techniques is illustrated in FIG. 1. Two terminals, an originating terminal **100** and a destination terminal **110** are in communication over the Internet. It is desired for the communications to be secure, that is, immune to eavesdropping. For example, terminal **100** may transmit secret information to terminal **110** over the Internet **107**. Also, it may be desired to prevent an eavesdropper from discovering that terminal **100** is in communication with terminal **110**. For example, if terminal **100** is a user and terminal **110** hosts a web site, terminal **100**'s user may not want anyone in the intervening networks to know what web sites he is "visiting." Anonymity would thus be an issue, for example, for companies that want to keep their market research interests private and thus would prefer to prevent outsiders from knowing which web-sites or other Internet resources they are "visiting." These two security issues may be called data security and anonymity, respectively.

Data security is usually tackled using some form of data encryption. An encryption key **48** is known at both the originating and terminating terminals **100** and **110**. The keys may be private and public at the originating and destination terminals **100** and **110**, respectively or they may be symmetrical keys (the same key is used by both parties to encrypt and decrypt). Many encryption methods are known and usable in this context.

To hide traffic from a local administrator or ISP, a user can employ a local proxy server in communicating over an encrypted channel with an outside proxy such that the local administrator or ISP only sees the encrypted traffic. Proxy servers prevent destination servers from determining the identities of the originating clients. This system employs an intermediate server interposed between client and destination server. The destination server sees only the Internet Protocol (IP) address of the proxy server and not the originating client. The target server only sees the address of the outside proxy. This scheme relies on a trusted outside proxy server. Also, proxy schemes are vulnerable to traffic analysis methods of determining identities of transmitters and receivers. Another important limitation of proxy servers is that the server knows the identities of both calling and called parties. In many instances, an originating terminal, such as terminal A, would prefer to keep its identity concealed from the proxy, for example, if the proxy server is provided by an Internet service provider (ISP).

To defeat traffic analysis, a scheme called Chaum's mixes employs a proxy server that transmits and receives fixed length messages, including dummy messages. Multiple originating terminals are connected through a mix (a server) to multiple target servers. It is difficult to tell which of the originating terminals are communicating to which of the connected target servers, and the dummy messages confuse eavesdroppers' efforts to detect communicating pairs by analyzing traffic. A drawback is that there is a risk that the mix server could be compromised. One way to deal with this risk is to spread the trust among multiple mixes. If one mix is compromised, the identities of the originating and target terminals may remain concealed. This strategy requires a number of alternative mixes so that the intermediate servers interposed between the originating and target terminals are not determinable except by compromising more than one mix. The strategy wraps the message with multiple layers of encrypted addresses. The first mix in a sequence can decrypt only the outer layer of the message to reveal the next destination mix in sequence. The second mix can decrypt the message to reveal the next mix and so on. The target server receives the message and, optionally, a multi-layer encrypted payload containing return information to send data back in the same fashion. The only way to defeat such a mix scheme is to collude among mixes. If the packets are all fixed-length and intermixed with dummy packets, there is no way to do any kind of traffic analysis.

Still another anonymity technique, called 'crowds,' protects the identity of the originating terminal from the intermediate proxies by providing that originating terminals belong to groups of proxies called crowds. The crowd proxies are interposed between originating and target terminals. Each proxy through which the message is sent is randomly chosen by an upstream proxy. Each intermediate proxy can send the message either to another randomly chosen proxy in the "crowd" or to the destination. Thus, even crowd members cannot determine if a preceding proxy is the originator of the message or if it was simply passed from another proxy.

ZKS (Zero-Knowledge Systems) Anonymous IP Protocol allows users to select up to any of five different pseudonyms, while desktop software encrypts outgoing traffic and wraps it in User Datagram Protocol (UDP) packets. The first server in a 2+-hop system gets the UDP packets, strips off one layer of encryption to add another, then sends the traffic to the next server, which strips off yet another layer of encryption and adds a new one. The user is permitted to control the number of hops. At the final server, traffic is decrypted with an untraceable IP address. The technique is called onion-routing. This method can be defeated using traffic analysis. For a simple example, bursts of packets from a user during low-duty periods can reveal the identities of sender and receiver.

Firewalls attempt to protect LANs from unauthorized access and hostile exploitation or damage to computers connected to the LAN. Firewalls provide a server through which all access to the LAN must pass. Firewalls are centralized systems that require administrative overhead to maintain. They can be compromised by virtual-machine applications ("applets"). They instill a false sense of security that leads to security breaches for example by users sending sensitive information to servers outside the firewall or encouraging use of modems to sidestep the firewall security. Firewalls are not useful for distributed systems such as business travelers, extranets, small teams, etc.

SUMMARY OF THE INVENTION

A secure mechanism for communicating over the internet, including a protocol referred to as the Tunneled Agile Routing Protocol (TARP), uses a unique two-layer encryption format and special TARP routers. TARP routers are similar in function to regular IP routers. Each TARP router has one or more IP addresses and uses normal IP protocol to send IP packet messages ("packets" or "datagrams"). The IP packets exchanged between TARP terminals via TARP routers are actually encrypted packets whose true destination address is concealed except to TARP routers and servers. The normal or "clear" or "outside" IP header attached to TARP IP packets contains only the address of a next hop router or destination server. That is, instead of indicating a final destination in the destination field of the IP header, the TARP packet's IP header always points to a next-hop in a series of TARP router hops, or to the final destination. This means there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet since the destination could always be next-hop TARP router as well as the final destination.

Each TARP packet's true destination is concealed behind a layer of encryption generated using a link key. The link key is the encryption key used for encrypted communication between the hops intervening between an originating TARP terminal and a destination TARP terminal. Each TARP router can remove the outer layer of encryption to reveal the destination router for each TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal by the sender/receiver IP numbers in the cleartext IP header.

Once the outer layer of encryption is removed, the TARP router determines the final destination. Each TARP packet **140** undergoes a minimum number of hops to help foil traffic analysis. The hops may be chosen at random or by a fixed value. As a result, each TARP packet may make random trips among a number of geographically disparate routers before reaching its destination. Each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined. This feature is called agile routing. The fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. The associated advantages have to do with the inner layer of encryption discussed below. Agile routing is combined with another feature that furthers this purpose; a feature that ensures that any message is broken into multiple packets.

The IP address of a TARP router can be changed, a feature called IP agility. Each TARP router, independently or under direction from another TARP terminal or router, can change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs.

The message payload is hidden behind an inner layer of encryption in the TARP packet that can only be unlocked

using a session key. The session key is not available to any of the intervening TARP routers. The session key is used to decrypt the payloads of the TARP packets permitting the data stream to be reconstructed.

Communication may be made private using link and session keys, which in turn may be shared and used according to any desired method. For example, public/private keys or symmetric keys may be used.

To transmit a data stream, a TARP originating terminal constructs a series of TARP packets from a series of IP packets generated by a network (IP) layer process. (Note that the terms "network layer," "data link layer," "application layer," etc. used in this specification correspond to the Open Systems Interconnection (OSI) network terminology.) The payloads of these packets are assembled into a block and chain-block encrypted using the session key. This assumes, of course, that all the IP packets are destined for the same TARP terminal. The block is then interleaved and the interleaved encrypted block is broken into a series of payloads, one for each TARP packet to be generated. Special TARP headers IPT are then added to each payload using the IP headers from the data stream packets. The TARP headers can be identical to normal IP headers or customized in some way. They should contain a formula or data for deinterleaving the data at the destination TARP terminal, a time-to-live (TTL) parameter to indicate the number of hops still to be executed, a data type identifier which indicates whether the payload contains, for example, TCP or UDP data, the sender's TARP address, the destination TARP address, and an indicator as to whether the packet contains real or decoy data or a formula for filtering out decoy data if decoy data is spread in some way through the TARP payload data.

Note that although chain-block encryption is discussed here with reference to the session key, any encryption method may be used. Preferably, as in chain block encryption, a method should be used that makes unauthorized decryption difficult without an entire result of the encryption process. Thus, by separating the encrypted block among multiple packets and making it difficult for an interloper to obtain access to all of such packets, the contents of the communications are provided an extra layer of security.

Decoy or dummy data can be added to a stream to help foil traffic analysis by reducing the peak-to-average network load. It may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to portion, or entirety, of a message, and that portion or entirety then interleaved into a number of separate packets. Considering the agile IP routing of the packets, and the attendant difficulty of reconstructing an entire sequence of packets to form a single block-encrypted message element, decoy packets can significantly increase the difficulty of reconstructing an entire data stream.

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Because the encryption system described above is insertable between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the Network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicating that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. In addition, it may create a subprocess that maintains the original IP address and continues interacting with the attacker in some manner.

Decoy packets may be generated by each TARP terminal on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis.

In various other embodiments of the invention, a scalable version of the system may be constructed in which a plurality of IP addresses are preassigned to each pair of communicating nodes in the network. Each pair of nodes agrees upon an algorithm for "hopping" between IP addresses (both sending and receiving), such that an eavesdropper sees apparently continuously random IP address pairs (source and destination) for packets transmitted between the pair. Overlapping or "reusable" IP addresses may be allocated to different users on the same subnet, since each node merely verifies that a particular packet includes a valid source/destination pair from the agreed-upon algorithm. Source/destination pairs are preferably not reused between any two nodes during any given end-to-end session, though limited IP block sizes or lengthy sessions might require it.

Further improvements described in this continuation-in-part application include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is an illustration of secure communications over the Internet according to a prior art embodiment.

FIG. **2** is an illustration of secure communications over the Internet according to a an embodiment of the invention.

FIG. **3*a*** is an illustration of a process of forming a tunneled IP packet according to an embodiment of the invention.

FIG. **3*b*** is an illustration of a process of forming a tunneled IP packet according to another embodiment of the invention.

FIG. **4** is an illustration of an OSI layer location of processes that may be used to implement the invention.

FIG. **5** is a flow chart illustrating a process for routing a tunneled packet according to an embodiment of the invention.

FIG. **6** is a flow chart illustrating a process for forming a tunneled packet according to an embodiment of the invention.

FIG. **7** is a flow chart illustrating a process for receiving a tunneled packet according to an embodiment of the invention.

FIG. **8** shows how a secure session is established and synchronized between a client and a TARP router.

FIG. **9** shows an IP address hopping scheme between a client computer and TARP router using transmit and receive tables in each computer.

FIG. **10** shows physical link redundancy among three Internet Service Providers (ISPs) and a client computer.

FIG. **11** shows how multiple IP packets can be embedded into a single "frame" such as an Ethernet frame, and further shows the use of a discriminator field to camouflage true packet recipients.

FIG. **12A** shows a system that employs hopped hardware addresses, hopped IP addresses, and hopped discriminator fields.

FIG. **12B** shows several different approaches for hopping hardware addresses, IP addresses, and discriminator fields in combination.

FIG. **13** shows a technique for automatically re-establishing synchronization between sender and receiver through the use of a partially public sync value.

FIG. **14** shows a "checkpoint" scheme for regaining synchronization between a sender and recipient.

FIG. **15** shows further details of the checkpoint scheme of FIG. **14**.

FIG. **16** shows how two addresses can be decomposed into a plurality of segments for comparison with presence vectors.

FIG. **17** shows a storage array for a receiver's active addresses.

FIG. **18** shows the receiver's storage array after receiving a sync request.

FIG. **19** shows the receiver's storage array after new addresses have been generated.

DETAILED DESCRIPTION OF THE INVENTION

Referring to FIG. 2, a secure mechanism for communicating over the internet employs a number of special routers or servers, called TARP routers 122-127 that are similar to regular IP routers 128-132 in that each has one or more IP addresses and uses normal IP protocol to send normal-looking IP packet messages, called TARP packets 140. TARP packets 140 are identical to normal IP packet messages that are routed by regular IP routers 128-132 because each TARP packet 140 contains a destination address as in a normal IP packet. However, instead of indicating a final destination in the destination field of the IP header, the TARP packet's 140 IP header always points to a next-hop in a series of TARP router hops, or the final destination, TARP terminal 110. Because the header of the TARP packet contains only the next-hop destination, there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet 140 since the destination could always be the next-hop TARP router as well as the final destination, TARP terminal 110.

Each TARP packet's true destination is concealed behind an outer layer of encryption generated using a link key 146. The link key 146 is the encryption key used for encrypted communication between the end points (TARP terminals or TARP routers) of a single link in the chain of hops connecting the originating TARP terminal 100 and the destination TARP terminal 110. Each TARP router 122-127, using the link key 146 it uses to communicate with the previous hop in a chain, can use the link key to reveal the true destination of a TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal (which may indicate the link key used) by the sender field of the clear IP header. Alternatively, this identity may be hidden behind another layer of encryption in available bits in the

clear IP header. Each TARP router, upon receiving a TARP message, determines if the message is a TARP message by using authentication data in the TARP packet. This could be recorded in available bytes in the TARP packet's IP header. Alternatively, TARP packets could be authenticated by attempting to decrypt using the link key 146 and determining if the results are as expected. The former may have computational advantages because it does not involve a decryption process.

Once the outer layer of decryption is completed by a TARP router 122-127, the TARP router determines the final destination. The system is preferably designed to cause each TARP packet 140 to undergo a minimum number of hops to help foil traffic analysis. The time to live counter in the IP header of the TARP message may be used to indicate a number of TARP router hops yet to be completed. Each TARP router then would decrement the counter and determine from that whether it should forward the TARP packet 140 to another TARP router 122-127 or to the destination TARP terminal 110. If the time to live counter is zero or below zero after decrementing, for an example of usage, the TARP router receiving the TARP packet 140 may forward the TARP packet 140 to the destination TARP terminal 110. If the time to live counter is above zero after decrementing, for an example of usage, the TARP router receiving the TARP packet 140 may forward the TARP packet 140 to a TARP router 122-127 that the current TARP terminal chooses at random. As a result, each TARP packet 140 is routed through some minimum number of hops of TARP routers 122-127 which are chosen at random.

Thus, each TARP packet, irrespective of the traditional factors determining traffic in the Internet, makes random trips among a number of geographically disparate routers before reaching its destination and each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined as described above. This feature is called agile routing. For reasons that will become clear shortly, the fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. Agile routing is combined with another feature that furthers this purpose, a feature that ensures that any message is broken into multiple packets.

A TARP router receives a TARP packet when an IP address used by the TARP router coincides with the IP address in the TARP packet's IP header IP_C. The IP address of a TARP router, however, may not remain constant. To avoid and manage attacks, each TARP router, independently or under direction from another TARP terminal or router, may change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs. In reality, whenever a TARP router looks up the address of a destination in the encrypted header, it must convert a TARP address to a real IP address using its LUT.

While every TARP router receiving a TARP packet has the ability to determine the packet's final destination, the message payload is embedded behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the TARP routers 122-127 intervening between the originating 100 and destination 110 TARP terminals. The session key is used to decrypt the payloads of the TARP packets **140** permitting an entire message to be reconstructed.

In one embodiment, communication may be made private using link and session keys, which in turn may be shared and used according any desired method. For example, a public key or symmetric keys may be communicated between link or session endpoints using a public key method. Any of a variety of other mechanisms for securing data to ensure that only authorized computers can have access to the private information in the TARP packets **140** may be used as desired.

Referring to FIG. **3***a*, to construct a series of TARP packets, a data stream **300** of IP packets **207***a*, **207***b*, **207***c*, etc., such series of packets being formed by a network (IP) layer process, is broken into a series of small sized segments. In the present example, equal-sized segments **1**-**9** are defined and used to construct a set of interleaved data packets A, B, and C. Here it is assumed that the number of interleaved packets A, B, and C formed is three and that the number of IP packets **207***a*-**207***c* used to form the three interleaved packets A, B, and C is exactly three. Of course, the number of IP packets spread over a group of interleaved packets may be any convenient number as may be the number of interleaved packets over which the incoming data stream is spread. The latter, the number of interleaved packets over which the data stream is spread, is called the interleave window.

To create a packet, the transmitting software interleaves the normal IP packets **207***a* et. seq. to form a new set of interleaved payload data **320**. This payload data **320** is then encrypted using a session key to form a set of session-key-encrypted payload data **330**, each of which, A, B, and C, will form the payload of a TARP packet. Using the IP header data, from the original packets **207***a*-**207***c*, new TARP headers IPT are formed. The TARP headers IPT can be identical to normal IP headers or customized in some way. In a preferred embodiment, the TARP headers IPT are IP headers with added data providing the following information required for routing and reconstruction of messages, some of which data is ordinarily, or capable of being, contained in normal IP headers:

1. A window sequence number—an identifier that indicates where the packet belongs in the original message sequence.

2. An interleave sequence number—an identifier that indicates the interleaving sequence used to form the packet so that the packet can be deinterleaved along with other packets in the interleave window.

3. A time-to-live (TTL) datum—indicates the number of TARP-router-hops to be executed before the packet reaches its destination. Note that the TTL parameter may provide a datum to be used in a probabilistic formula for determining whether to route the packet to the destination or to another hop.

4. Data type identifier—indicates whether the payload contains, for example, TCP or UDP data.

5. Sender's address—indicates the sender's address in the TARP network.

6. Destination address—indicates the destination terminal's address in the TARP network.

7. Decoy/Real—an indicator of whether the packet contains real message data or dummy decoy data or a combination.

Obviously, the packets going into a single interleave window must include only packets with a common destination. Thus, it is assumed in the depicted example that the IP headers of IP packets **207***a*-**207***c* all contain the same destination address or at least will be received by the same terminal so that they can be deinterleaved. Note that dummy or decoy data or packets can be added to form a larger interleave window than would otherwise be required by the size of a given message. Decoy or dummy data can be added to a stream to help foil traffic analysis by leveling the load on the network. Thus, it may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to a portion, or the entirety, of a message, and that portion or entirety then interleaved into a number of separate packets.

Referring to FIG. **3***b*, in an alternative mode of TARP packet construction, a series of IP packets are accumulated to make up a predefined interleave window. The payloads of the packets are used to construct a single block **520** for chain block encryption using the session key. The payloads used to form the block are presumed to be destined for the same terminal. The block size may coincide with the interleave window as depicted in the example embodiment of FIG. **3***b*. After encryption, the encrypted block is broken into separate payloads and segments which are interleaved as in the embodiment of FIG. **3***a*. The resulting interleaved packets A, B, and C, are then packaged as TARP packets with TARP headers as in the Example of FIG. **3***a*. The remaining process is as shown in, and discussed with reference to, FIG. **3***a*.

Once the TARP packets **340** are formed, each entire TARP packet **340**, including the TARP header IP_T, is encrypted using the link key for communication with the first-hop-TARP router. The first hop TARP router is randomly chosen. A final unencrypted IP header IP_C is added to each encrypted TARP packet **340** to form a normal IP packet **360** that can be transmitted to a TARP router. Note that the process of constructing the TARP packet **360** does not have to be done in stages as described. The above description is just a useful heuristic for describing the final product, namely, the TARP packet.

Note that, TARP header IP_T could be a completely custom header configuration with no similarity to a normal IP header except that it contain the information identified above. This is so since this header is interpreted by only TARP routers.

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Referring to FIG. **4**, a TARP transceiver **405** can be an originating terminal **100**, a destination terminal **110**, or a TARP router **122**-**127**. In each TARP Transceiver **405**, a transmitting process is generated to receive normal packets from the Network (IP) layer and generate TARP packets for communication over the network. A receiving process is generated to receive normal IP packets containing TARP packets and generate from these normal IP packets which are "passed up" to the Network (IP) layer. Note that where the TARP Transceiver **405** is a router, the received TARP packets **140** are not processed into a stream of IP packets **415** because they need only be authenticated as proper TARP packets and then passed to another TARP router or a TARP destination terminal **110**. The intervening process, a "TARP Layer" **420**, could be combined with either the data link layer **430** or the Network layer **410**. In either case, it would intervene between the data link layer **430** so that the process would receive regular IP packets containing embedded TARP packets and "hand up" a series of reassembled IP packets to the Network layer **410**. As an example of combining the TARP layer **420** with the data link layer **430**, a program may augment the normal processes running a communications card, for example, an Ethernet card. Alternatively, the TARP layer processes may form part of a dynamically loadable module that is loaded and executed to support communications between the network and data link layers.

Because the encryption system described above can be inserted between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

Note that IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicates that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. To accomplish this, the TARP process will construct a TARP-formatted message, in the style of Internet Control Message Protocol (ICMP) datagrams as an example; this message will contain the machine's TARP address, its previous IP address, and its new IP address. The TARP layer will transmit this packet to at least one known TARP router; then upon receipt and validation of the message, the TARP router will update its LUT with the new IP address for the stated TARP address. The TARP router will then format a similar message, and broadcast it to the other TARP routers so that they may update their LUTs. Since the total number of TARP routers on any given subnet is expected to be relatively small, this process of updating the LUTs should be relatively fast. It may not, however, work as well when there is a relatively large number of TARP routers and/or a relatively large number of clients; this has motivated a refinement of this architecture to provide scalability; this refinement has led to a second embodiment, which is discussed below.

Upon detection of an attack, the TARP process may also create a subprocess that maintains the original IP address and continues interacting with the attacker. The latter may provide an opportunity to trace the attacker or study the attacker's methods (called "fishbowling" drawing upon the analogy of a small fish in a fish bowl that "thinks" it is in the ocean but is actually under captive observation). A history of the communication between the attacker and the abandoned (fishbowled) IP address can be recorded or transmitted for human analysis or further synthesized for purposes of responding in some way.

As mentioned above, decoy or dummy data or packets can be added to outgoing data streams by TARP terminals or routers. In addition to making it convenient to spread data over a larger number of separate packets, such decoy packets can also help to level the load on inactive portions of the Internet to help foil traffic analysis efforts.

Decoy packets may be generated by each TARP terminal **100**, **110** or each router **122-127** on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. That is, when a series of messages are received, the decoy packet generation rate may be increased. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis. The rate of reception of packets, decoy or otherwise, may be indicated to the decoy packet dropping and generating processes through perishable decoy and regular packet counters. (A perishable counter is one that resets or decrements its value in response to time so that it contains a high value when it is incremented in rapid succession and a small value when incremented either slowly or a small number of times in rapid succession.) Note that destination TARP terminal **110** may generate decoy packets equal in number and size to those TARP packets received to make it appear it is merely routing packets and is therefore not the destination terminal.

Referring to FIG. **5**, the following particular steps may be employed in the above-described method for routing TARP packets.

S0. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.

S2. The TARP packet may be probed in some way to authenticate the packet before attempting to decrypt it using the link key. That is, the router may determine that the packet is an authentic TARP packet by performing a selected operation on some data included with the clear IP header attached to the encrypted TARP packet contained in the payload. This makes it possible to avoid performing decryption on packets that are not authentic TARP packets.

S3. The TARP packet is decrypted to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.

S4. If the packet is a decoy packet, the perishable decoy counter is incremented.

S5. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the router may choose to throw it away. If the received packet is a decoy packet and it is determined that it should be thrown away (S6), control returns to step S0.

S7. The TTL parameter of the TARP header is decremented and it is determined if the TTL parameter is greater than zero.

S8. If the TTL parameter is greater than zero, a TARP address is randomly chosen from a list of TARP addresses maintained by the router and the link key and IP address corresponding to that TARP address memorized for use in creating a new IP packet containing the TARP packet.

S9. If the TTL parameter is zero or less, the link key and IP address corresponding to the TARP address of the destination are memorized for use in creating the new IP packet containing the TARP packet.

S10. The TARP packet is encrypted using the memorized link key.

S11. An IP header is added to the packet that contains the stored IP address, the encrypted TARP packet wrapped with an IP header, and the completed packet transmitted to the next hop or destination.

Referring to FIG. 6, the following particular steps may be employed in the above-described method for generating TARP packets.

S20. A background loop operation applies an algorithm that determines the generation of decoy IP packets. The loop is interrupted when a data stream containing IP packets is received for transmission.

S21. The received IP packets are grouped into a set consisting of messages with a constant IP destination address. The set is further broken down to coincide with a maximum size of an interleave window The set is encrypted, and interleaved into a set of payloads destined to become TARP packets.

S22. The TARP address corresponding to the IP address is determined from a lookup table and stored to generate the TARP header. An initial TTL count is generated and stored in the header. The TTL count may be random with minimum and maximum values or it may be fixed or determined by some other parameter.

S23. The window sequence numbers and interleave sequence numbers are recorded in the TARP headers of each packet.

S24. One TARP router address is randomly chosen for each TARP packet and the IP address corresponding to it stored for use in the clear IP header. The link key corresponding to this router is identified and used to encrypt TARP packets containing interleaved and encrypted data and TARP headers.

S25. A clear IP header with the first hop router's real IP address is generated and added to each of the encrypted TARP packets and the resulting packets.

Referring to FIG. 7, the following particular steps may be employed in the above-described method for receiving TARP packets.

S40. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.

S42. The TARP packet may be probed to authenticate the packet before attempting to decrypt it using the link key.

S43. The TARP packet is decrypted with the appropriate link key to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.

S44. If the packet is a decoy packet, the perishable decoy counter is incremented.

S45. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the receiver may choose to throw it away.

S46. The TARP packets are cached until all packets forming an interleave window are received.

S47. Once all packets of an interleave window are received, the packets are deinterleaved.

S48. The packets block of combined packets defining the interleave window is then decrypted using the session key.

S49. The decrypted block is then divided using the window sequence data and the IP_T headers are converted into normal IP_C headers. The window sequence numbers are integrated in the IP_C headers.

S50. The packets are then handed up to the IP layer processes.

1. SCALABILITY ENHANCEMENTS

The IP agility feature described above relies on the ability to transmit IP address changes to all TARP routers. The embodiments including this feature will be referred to as "boutique" embodiments due to potential limitations in scaling these features up for a large network, such as the Internet. (The "boutique" embodiments would, however, be robust for use in smaller networks, such as small virtual private networks, for example). One problem with the boutique embodiments is that if IP address changes are to occur frequently, the message traffic required to update all routers sufficiently quickly creates a serious burden on the Internet when the TARP router and/or client population gets large. The bandwidth burden added to the networks, for example in ICMP packets, that would be used to update all the TARP routers could overwhelm the Internet for a large scale implementation that approached the scale of the Internet. In other words, the boutique system's scalability is limited.

A system can be constructed which trades some of the features of the above embodiments to provide the benefits of IP agility without the additional messaging burden. This is accomplished by IP address-hopping according to shared algorithms that govern IP addresses used between links participating in communications sessions between nodes such as TARP nodes. (Note that the IP hopping technique is also applicable to the boutique embodiment.) The IP agility feature discussed with respect to the boutique system can be modified so that it becomes decentralized under this scalable regime and governed by the above-described shared algorithm. Other features of the boutique system may be combined with this new type of IP-agility.

The new embodiment has the advantage of providing IP agility governed by a local algorithm and set of IP addresses exchanged by each communicating pair of nodes. This local governance is session-independent in that it may govern communications between a pair of nodes, irrespective of the session or end points being transferred between the directly communicating pair of nodes.

In the scalable embodiments, blocks of IP addresses are allocated to each node in the network. (This scalability will increase in the future, when Internet Protocol addresses are increased to 128-bit fields, vastly increasing the number of distinctly addressable nodes). Each node can thus use any of the IP addresses assigned to that node to communicate with other nodes in the network. Indeed, each pair of communicating nodes can use a plurality of source IP addresses and destination IP addresses for communicating with each other.

Each communicating pair of nodes in a chain participating in any session stores two blocks of IP addresses, called netblocks, and an algorithm and randomization seed for selecting, from each netblock, the next pair of source/destination IP addresses that will be used to transmit the next message. In other words, the algorithm governs the sequential selection of IP-address pairs, one sender and one receiver IP address, from each netblock. The combination of algorithm, seed, and netblock (IP address block) will be called a "hopblock." A router issues separate transmit and receive hopblocks to its clients. The send address and the receive address of the IP header of each outgoing packet sent by the client are filled with the send and receive IP addresses generated by the algorithm. The algorithm is "clocked" (indexed) by a counter so that each time a pair is used, the algorithm turns out a new transmit pair for the next packet to be sent.

The router's receive hopblock is identical to the client's transmit hopblock. The router uses the receive hopblock to predict what the send and receive IP address pair for the next expected packet from that client will be. Since packets can be received out of order, it is not possible for the router to predict with certainty what IP address pair will be on the next sequential packet. To account for this problem, the router generates a range of predictions encompassing the number of possible transmitted packet send/receive addresses, of which the next packet received could leap ahead. Thus, if there is a vanishingly small probability that a given packet will arrive at the router ahead of 5 packets transmitted by the client before the given packet, then the router can generate a series of 6 send/receive IP address pairs (or "hop window") to compare with the next received packet. When a packet is received, it is marked in the hop window as such, so that a second packet with the same IP address pair will be discarded. If an out-of-sequence packet does not arrive within a predetermined timeout period, it can be requested for retransmission or simply discarded from the receive table, depending upon the protocol in use for that communications session, or possibly by convention.

When the router receives the client's packet, it compares the send and receive IP addresses of the packet with the next N predicted send and receive IP address pairs and rejects the packet if it is not a member of this set. Received packets that do not have the predicted source/destination IP addresses falling with the window are rejected, thus thwarting possible hackers. (With the number of possible combinations, even a fairly large window would be hard to fall into at random.) If it is a member of this set, the router accepts the packet and processes it further. This link-based IP-hopping strategy, referred to as "IHOP," is a network element that stands on its own and is not necessarily accompanied by elements of the boutique system described above. If the routing agility feature described in connection with the boutique embodiment is combined with this link-based IP-hopping strategy, the router's next step would be to decrypt the TARP header to determine the destination TARP router for the packet and determine what should be the next hop for the packet. The TARP router would then forward the packet to a random TARP router or the destination TARP router with which the source TARP router has a link-based IP hopping communication established.

FIG. **8** shows how a client computer **801** and a TARP router **811** can establish a secure session. When client **801** seeks to establish an IHOP session with TARP router **811**, the client **801** sends "secure synchronization" request ("SSYN") packet **821** to the TARP router **811**. This SYN packet **821** contains the client's **801** authentication token, and may be sent to the router **811** in an encrypted format. The source and destination IP numbers on the packet **821** are the client's **801** current fixed IP address, and a "known" fixed IP address for the router **811**. (For security purposes, it may be desirable to reject any packets from outside of the local network that are destined for the router's known fixed IP address.) Upon receipt and validation of the client's **801** SSYN packet **821**, the router **811** responds by sending an encrypted "secure synchronization acknowledgment" ("SSYN ACK") **822** to the client **801**. This SSYN ACK **822** will contain the transmit and receive hopblocks that the client **801** will use when communicating with the TARP router **811**. The client **801** will acknowledge the TARP router's **811** response packet **822** by generating an encrypted SSYN ACK ACK packet **823** which will be sent from the client's **801** fixed IP address and to the TARP router's **811** known fixed IP address. The client **801** will simultaneously generate a SSYN ACK ACK packet; this SSYN ACK packet, referred to as the Secure Session Initiation (SSI) packet **824**, will be sent with the first {sender, receiver} IP pair in the client's transmit table **921** (FIG. **9**), as specified in the transmit hopblock provided by the TARP router **811** in the SSYN ACK packet **822**. The TARP router **811** will respond to the SSI packet **824** with an SSI ACK packet **825**, which will be sent with the first {sender, receiver} IP pair in the TARP router's transmit table **923**. Once these packets have been successfully exchanged, the secure communications session is established, and all further secure communications between the client **801** and the TARP router **811** will be conducted via this secure session, as long as synchronization is maintained. If synchronization is lost, then the client **801** and TARP router **802** may re-establish the secure session by the procedure outlined in FIG. **8** and described above.

While the secure session is active, both the client **901** and TARP router **911** (FIG. **9**) will maintain their respective transmit tables **921**, **923** and receive tables **922**, **924**, as provided by the TARP router during session synchronization **822**. It is important that the sequence of IP pairs in the client's transmit table **921** be identical to those in the TARP router's receive table **924**; similarly, the sequence of IP pairs in the client's receive table **922** must be identical to those in the router's transmit table **923**. This is required for the session synchronization to be maintained. The client **901** need maintain only one transmit table **921** and one receive table **922** during the course of the secure session. Each sequential packet sent by the client **901** will employ the next {send, receive} IP address pair in the transmit table, regardless of TCP or UDP session. The TARP router **911** will expect each packet arriving from the client **901** to bear the next IP address pair shown in its receive table.

Since packets can arrive out of order, however, the router **911** can maintain a "look ahead" buffer in its receive table, and will mark previously-received IP pairs as invalid for future packets; any future packet containing an IP pair that is in the look-ahead buffer but is marked as previously received will be discarded. Communications from the TARP router **911** to the client **901** are maintained in an identical manner; in particular, the router **911** will select the next IP address pair from its transmit table **923** when constructing a packet to send to the client **901**, and the client **901** will maintain a look-ahead buffer of expected IP pairs on packets that it is receiving. Each TARP router will maintain separate pairs of transmit and receive tables for each client that is currently engaged in a secure session with or through that TARP router.

While clients receive their hopblocks from the first server linking them to the Internet, routers exchange hopblocks. When a router establishes a link-based IP-hopping communication regime with another router, each router of the pair exchanges its transmit hopblock. The transmit hopblock of each router becomes the receive hopblock of the other router. The communication between routers is governed as described by the example of a client sending a packet to the first router.

While the above strategy works fine in the IP milieu, many local networks that are connected to the Internet are Ethernet systems. In Ethernet, the IP addresses of the destination devices must be translated into hardware addresses, and vice versa, using known processes ("address resolution protocol," and "reverse address resolution protocol"). However, if the link-based IP-hopping strategy is employed, the correlation process would become explosive and burdensome. An alternative to the link-based IP hopping strategy may be employed within an Ethernet network. The solution is to provide that the node linking the Internet to the Ethernet (call it the border node) use the link-based IP-hopping communication regime to communicate with nodes outside the Ethernet LAN. Within the Ethernet LAN, each TARP node would have a single IP address which would be addressed in the conventional way. Instead of comparing the {sender, receiver} IP address pairs to authenticate a packet, the intra-LAN TARP node would use one of the IP header extension fields to do so. Thus, the border node uses an algorithm shared by the intra-LAN TARP node to generate a symbol that is stored in the free field in the IP header, and the intra-LAN TARP node generates a range of symbols based on its prediction of the next expected packet to be received from that particular source IP address. The packet is rejected if it does not fall into the set of predicted symbols (for example, numerical values) or is accepted if it does. Communications from the intra-LAN TARP node to the border node are accomplished in the same manner, though the algorithm will necessarily be different for security reasons. Thus, each of the communicating nodes will generate transmit and receive tables in a similar manner to that of FIG. 9; the intra-LAN TARP nodes transmit table will be identical to the border node's receive table, and the intra-LAN TARP node's receive table will be identical to the border node's transmit table.

The algorithm used for IP address-hopping can be any desired algorithm. For example, the algorithm can be a given pseudo-random number generator that generates numbers of the range covering the allowed IP addresses with a given seed. Alternatively, the session participants can assume a certain type of algorithm and specify simply a parameter for applying the algorithm. For example the assumed algorithm could be a particular pseudo-random number generator and the session participants could simply exchange seed values.

Note that there is no permanent physical distinction between the originating and destination terminal nodes. Either device at either end point can initiate a synchronization of the pair. Note also that the authentication/synchronization-request (and acknowledgment) and hopblock-exchange may all be served by a single message so that separate message exchanges may not be required.

As another extension to the stated architecture, multiple physical paths can be used by a client, in order to provide link redundancy and further thwart attempts at denial of service and traffic monitoring. As shown in FIG. 10, for example, client **1001** can establish three simultaneous sessions with each of three TARP routers provided by different ISPs **1011**, **1012**, **1013**. As an example, the client **1001** can use three different telephone lines **1021**, **1022**, **1023** to connect to the ISPs, or two telephone lines and a cable modem, etc. In this scheme, transmitted packets will be sent in a random fashion among the different physical paths. This architecture provides a high degree of communications redundancy, with improved immunity from denial-of-service attacks and traffic monitoring.

2. FURTHER EXTENSIONS

The following describes various extensions to the techniques, systems, and methods described above. As described above, the security of communications occurring between computers in a computer network (such as the Internet, an Ethernet, or others) can be enhanced by using seemingly random source and destination Internet Protocol (IP) addresses for data packets transmitted over the network. This feature prevents eavesdroppers from determining which computers in the network are communicating with each other while permitting the two communicating computers to easily recognize whether a given received data packet is legitimate or not. In one embodiment of the above-described systems, an IP header extension field is used to authenticate incoming packets on an Ethernet.

Various extensions to the previously described techniques described herein include: (1) use of hopped hardware or "MAC" addresses in broadcast type network; (2) a self-synchronization technique that permits a computer to automatically regain synchronization with a sender; (3) synchronization algorithms that allow transmitting and receiving computers to quickly re-establish synchronization in the event of lost packets or other events; and (4) a fast-packet rejection mechanism for rejecting invalid packets. Any or all of these extensions can be combined with the features described above in any of various ways.

A. Hardware Address Hopping

Internet protocol-based communications techniques on a LAN—or across any dedicated physical medium—typically embed the IP packets within lower-level packets, often referred to as "frames." As shown in FIG. 11, for example, a first Ethernet frame **1150** comprises a frame header **1101** and two embedded IP packets IP**1** and IP**2**, while a second Ethernet frame **1160** comprises a different frame header **1104** and a single IP packet IP**3**. Each frame header generally includes a source hardware address **1101A** and a destination hardware address **1101B**; other well-known fields in frame headers are omitted from FIG. **11** for clarity. Two hardware nodes communicating over a physical communication channel insert appropriate source and destination hardware addresses to indicate which nodes on the channel or network should receive the frame.

It may be possible for a nefarious listener to acquire information about the contents of a frame and/or its communicants by examining frames on a local network rather than (or in addition to) the IP packets themselves. This is especially true in broadcast media, such as Ethernet, where it is necessary to insert into the frame header the hardware address of the machine that generated the frame and the hardware address of the machine to which frame is being sent. All nodes on the network can potentially "see" all packets transmitted across the network. This can be a problem for secure communications, especially in cases where the communicants do not want for any third party to be able to identify who is engaging in the information exchange. One way to address this problem is to push the address-hopping scheme down to the hardware layer. In accordance with various embodiments of the invention, hardware addresses are "hopped" in a manner similar to that used to change IP addresses, such that a listener cannot determine which hardware node generated a particular message nor which node is the intended recipient.

FIG. 12A shows a system in which Media Access Control ("MAC") hardware addresses are "hopped" in order to increase security over a network such as an Ethernet. While the description refers to the exemplary case of an Ethernet environment, the inventive principles are equally applicable to other types of communications media. In the Ethernet case, the MAC address of the sender and receiver are inserted into the Ethernet frame and can be observed by anyone on the LAN who is within the broadcast range for that frame. For secure communications, it becomes desirable to generate frames with MAC addresses that are not attributable to any specific sender or receiver.

As shown in FIG. 12A, two computer nodes **1201** and **1202** communicate over a communication channel such as an Ethernet. Each node executes one or more application programs **1203** and **1218** that communicate by transmitting packets through communication software **1204** and **1217**, respectively. Examples of application programs include video conferencing, e-mail, word processing programs, telephony, and the like. Communication software **1204** and **1217** can comprise, for example, an OSI layered architecture or "stack" that standardizes various services provided at different levels of functionality.

The lowest levels of communication software **1204** and **1217** communicate with hardware components **1206** and **1214** respectively, each of which can include one or more registers **1207** and **1215** that allow the hardware to be reconfigured or controlled in accordance with various communication protocols. The hardware components (an Ethernet network interface card, for example) communicate with each other over the communication medium. Each hardware component is typically pre-assigned a fixed hardware address or MAC number that identifies the hardware component to other nodes on the network. One or more interface drivers control the operation of each card and can, for example, be configured to accept or reject packets from certain hardware addresses. As will be described in more detail below, various embodiments of the inventive principles provide for "hopping" different addresses using one or more algorithms and one or more moving windows that track a range of valid addresses to validate received packets. Packets transmitted according to one or more of the inventive principles will be generally referred to as "secure" packets or "secure communications" to differentiate them from ordinary data packets that are transmitted in the clear using ordinary, machine-correlated addresses.

One straightforward method of generating non-attributable MAC addresses is an extension of the IP hopping scheme. In this scenario, two machines on the same LAN that desire to communicate in a secure fashion exchange random-number generators and seeds, and create sequences of quasi-random MAC addresses for synchronized hopping. The implementation and synchronization issues are then similar to that of IP hopping.

This approach, however, runs the risk of using MAC addresses that are currently active on the LAN-which, in turn, could interrupt communications for those machines. Since an Ethernet MAC address is at present 48 bits in length, the chance of randomly misusing an active MAC address is actually quite small. However, if that figure is multiplied by a large number of nodes (as would be found on an extensive LAN), by a large number of frames (as might be the case with packet voice or streaming video), and by a large number of concurrent Virtual Private Networks (VPNs), then the chance that a non-secure machine's MAC address could be used in an

address-hopped frame can become non-trivial. In short, any scheme that runs even a small risk of interrupting communications for other machines on the LAN is bound to receive resistance from prospective system administrators. Nevertheless, it is technically feasible, and can be implemented without risk on a LAN on which there is a small number of machines, or if all of the machines on the LAN are engaging in MAC-hopped communications.

Synchronized MAC address hopping may incur some overhead in the course of session establishment, especially if there are multiple sessions or multiple nodes involved in the communications. A simpler method of randomizing MAC addresses is to allow each node to receive and process every incident frame on the network. Typically, each network interface driver will check the destination MAC address in the header of every incident frame to see if it matches that machine's MAC address; if there is no match, then the frame is discarded. In one embodiment, however, these checks can be disabled, and every incident packet is passed to the TARP stack for processing. This will be referred to as "promiscuous" mode, since every incident frame is processed. Promiscuous mode allows the sender to use completely random, unsynchronized MAC addresses, since the destination machine is guaranteed to process the frame. The decision as to whether the packet was truly intended for that machine is handled by the TARP stack, which checks the source and destination IP addresses for a match in its IP synchronization tables. If no match is found, the packet is discarded; if there is a match, the packet is unwrapped, the inner header is evaluated, and if the inner header indicates that the packet is destined for that machine then the packet is forwarded to the IP stack—otherwise it is discarded.

One disadvantage of purely-random MAC address hopping is its impact on processing overhead; that is, since every incident frame must be processed, the machine's CPU is engaged considerably more often than if the network interface driver is discriminating and rejecting packets unilaterally. A compromise approach is to select either a single fixed MAC address or a small number of MAC addresses (e.g., one for each virtual private network on an Ethernet) to use for MAC-hopped communications, regardless of the actual recipient for which the message is intended. In this mode, the network interface driver can check each incident frame against one (or a few) pre-established MAC addresses, thereby freeing the CPU from the task of physical-layer packet discrimination. This scheme does not betray any useful information to an interloper on the LAN; in particular, every secure packet can already be identified by a unique packet type in the outer header. However, since all machines engaged in secure communications would either be using the same MAC address, or be selecting from a small pool of predetermined MAC addresses, the association between a specific machine and a specific MAC address is effectively broken.

In this scheme, the CPU will be engaged more often than it would be in non-secure communications (or in synchronized MAC address hopping), since the network interface driver cannot always unilaterally discriminate between secure packets that are destined for that machine, and secure packets from other VPNs. However, the non-secure traffic is easily eliminated at the network interface, thereby reducing the amount of processing required of the CPU. There are boundary conditions where these statements would not hold, of course—e.g., if all of the traffic on the LAN is secure traffic, then the CPU would be engaged to the same degree as it is in the purely-random address hopping case; alternatively, if each VPN on the LAN uses a different MAC address, then the network interface can perfectly discriminate secure frames destined for the local machine from those constituting other VPNs. These are engineering tradeoffs that might be best handled by providing administrative options for the users when installing the software and/or establishing VPNs.

Even in this scenario, however, there still remains a slight risk of selecting MAC addresses that are being used by one or more nodes on the LAN. One solution to this problem is to formally assign one address or a range of addresses for use in MAC-hopped communications. This is typically done via an assigned numbers registration authority; e.g., in the case of Ethernet, MAC address ranges are assigned to vendors by the Institute of Electrical and Electronics Engineers (IEEE). A formally-assigned range of addresses would ensure that secure frames do not conflict with any properly-configured and properly-functioning machines on the LAN.

Reference will now be made to FIGS. **12**A and **12**B in order to describe the many combinations and features that follow the inventive principles. As explained above, two computer nodes **1201** and **1202** are assumed to be communicating over a network or communication medium such as an Ethernet. A communication protocol in each node (**1204** and **1217**, respectively) contains a modified element **1205** and **1216** that performs certain functions that deviate from the standard communication protocols. In particular, computer node **1201** implements a first "hop" algorithm **1208**X that selects seemingly random source and destination IP addresses (and, in one embodiment, seemingly random IP header discriminator fields) in order to transmit each packet to the other computer node. For example, node **1201** maintains a transmit table **1208** containing triplets of source (S), destination (D), and discriminator fields (DS) that are inserted into outgoing IP packet headers. The table is generated through the use of an appropriate algorithm (e.g., a random number generator that is seeded with an appropriate seed) that is known to the recipient node **1202**. As each new IP packet is formed, the next sequential entry out of the sender's transmit table **1208** is used to populate the IP source, IP destination, and IP header extension field (e.g., discriminator field). It will be appreciated that the transmit table need not be created in advance but could instead be created on-the-fly by executing the algorithm when each packet is formed.

At the receiving node **1202**, the same IP hop algorithm **1222**X is maintained and used to generate a receive table **1222** that lists valid triplets of source IP address, destination IP address, and discriminator field. This is shown by virtue of the first five entries of transmit table **1208** matching the second five entries of receive table **1222**. (The tables may be slightly offset at any particular time due to lost packets, misordered packets, or transmission delays). Additionally, node **1202** maintains a receive window **W3** that represents a list of valid IP source, IP destination, and discriminator fields that will be accepted when received as part of an incoming IP packet. As packets are received, window **W3** slides down the list of valid entries, such that the possible valid entries change over time. Two packets that arrive out of order but are nevertheless matched to entries within window **W3** will be accepted; those falling outside of window **W3** will be rejected as invalid. The length of window **W3** can be adjusted as necessary to reflect network delays or other factors.

Node **1202** maintains a similar transmit table **1221** for creating IP packets and frames destined for node **1201** using a potentially different hopping algorithm **1221**X, and node **1201** maintains a matching receive table **1209** using the same algorithm **1209**X. As node **1202** transmits packets to node **1201** using seemingly random IP source, IP destination, and/or discriminator fields, node **1201** matches the incoming

packet values to those falling within window **W1** maintained in its receive table. In effect, transmit table **1208** of node **1201** is synchronized (i.e., entries are selected in the same order) to receive table **1222** of receiving node **1202**. Similarly, transmit table **1221** of node **1202** is synchronized to receive table **1209** of node **1201**. It will be appreciated that although a common algorithm is shown for the source, destination and discriminator fields in FIG. **12**A (using, e.g., a different seed for each of the three fields), an entirely different algorithm could in fact be used to establish values for each of these fields. It will also be appreciated that one or two of the fields can be "hopped" rather than all three as illustrated.

In accordance with another aspect of the invention, hardware or "MAC" addresses are hopped instead of or in addition to IP addresses and/or the discriminator field in order to improve security in a local area or broadcast-type network. To that end, node **1201** further maintains a transmit table **1210** using a transmit algorithm **1210**X to generate source and destination hardware addresses that are inserted into frame headers (e.g., fields **1101**A and **1101**B in FIG. **11**) that are synchronized to a corresponding receive table **1224** at node **1202**. Similarly, node **1202** maintains a different transmit table **1223** containing source and destination hardware addresses that is synchronized with a corresponding receive table **1211** at node **1201**. In this manner, outgoing hardware frames appear to be originating from and going to completely random nodes on the network, even though each recipient can determine whether a given packet is intended for it or not. It will be appreciated that the hardware hopping feature can be implemented at a different level in the communications protocol than the IP hopping feature (e.g., in a card driver or in a hardware card itself to improve performance).

FIG. **12**B shows three different embodiments or modes that can be employed using the aforementioned principles. In a first mode referred to as "promiscuous" mode, a common hardware address (e.g., a fixed address for source and another for destination) or else a completely random hardware address is used by all nodes on the network, such that a particular packet cannot be attributed to any one node. Each node must initially accept all packets containing the common (or random) hardware address and inspect the IP addresses or discriminator field to determine whether the packet is intended for that node. In this regard, either the IP addresses or the discriminator field or both can be varied in accordance with an algorithm as described above. As explained previously, this may increase each node's overhead since additional processing is involved to determine whether a given packet has valid source and destination hardware addresses.

In a second mode referred to as "promiscuous per VPN" mode, a small set of fixed hardware addresses are used, with a fixed source/destination hardware address used for all nodes communicating over a virtual private network. For example, if there are six nodes on an Ethernet, and the network is to be split up into two private virtual networks such that nodes on one VPN can communicate with only the other two nodes on its own VPN, then two sets of hardware addresses could be used: one set for the first VPN and a second set for the second VPN. This would reduce the amount of overhead involved in checking for valid frames since only packets arriving from the designated VPN would need to be checked. IP addresses and one or more discriminator fields could still be hopped as before for secure communication within the VPN. Of course, this solution compromises the anonymity of the VPNs (i.e., an outsider can easily tell what traffic belongs in which VPN, though he cannot correlate it to a specific machine/person). It also requires the use of a discriminator field to mitigate the vulnerability to certain types of DoS attacks. (For example, without the discriminator field, an attacker on the LAN could stream frames containing the MAC addresses being used by the VPN; rejecting those frames could lead to excessive processing overhead. The discriminator field would provide a low-overhead means of rejecting the false packets.)

In a third mode referred to as "hardware hopping" mode, hardware addresses are varied as illustrated in FIG. 12A, such that hardware source and destination addresses are changed constantly in order to provide non-attributable addressing. Variations on these embodiments are of course possible, and the invention is not intended to be limited in any respect by these illustrative examples.

B. Extending the Address Space

Address hopping provides security and privacy. However, the level of protection is limited by the number of addresses in the blocks being hopped. A hopblock denotes a field or fields modulated on a packet-wise basis for the purpose of providing a VPN. For instance, if two nodes communicate with IP address hopping using hopblocks of 4 addresses (2 bits) each, there would be 16 possible address-pair combinations. A window of size 16 would result in most address pairs being accepted as valid most of the time. This limitation can be overcome by using a discriminator field in addition to or instead of the hopped address fields. The discriminator field would be hopped in exactly the same fashion as the address fields and it would be used to determine whether a packet should be processed by a receiver.

Suppose that two clients, each using four-bit hopblocks, would like the same level of protection afforded to clients communicating via IP hopping between two A blocks (24 address bits eligible for hopping). A discriminator field of 20 bits, used in conjunction with the 4 address bits eligible for hopping in the IP address field, provides this level of protection. A 24-bit discriminator field would provide a similar level of protection if the address fields were not hopped or ignored. Using a discriminator field offers the following advantages: (1) an arbitrarily high level of protection can be provided, and (2) address hopping is unnecessary to provide protection. This may be important in environments where address hopping would cause routing problems.

C. Synchronization Techniques

It is generally assumed that once a sending node and receiving node have exchanged algorithms and seeds (or similar information sufficient to generate quasi-random source and destination tables), subsequent communication between the two nodes will proceed smoothly. Realistically, however, two nodes may lose synchronization due to network delays or outages, or other problems. Consequently, it is desirable to provide means for re-establishing synchronization between nodes in a network that have lost synchronization.

One possible technique is to require that each node provide an acknowledgment upon successful receipt of each packet and, if no acknowledgment is received within a certain period of time, to re-send the unacknowledged packet. This approach, however, drives up overhead costs and may be prohibitive in high-throughput environments such as streaming video or audio, for example.

A different approach is to employ an automatic synchronizing technique that will be referred to herein as "self-synchronization." In this approach, synchronization information is embedded into each packet, thereby enabling the receiver to re-synchronize itself upon receipt of a single packet if it determines that is has lost synchronization with the sender. (If communications are already in progress, and the receiver determines that it is still in sync with the sender, then there is no need to re-synchronize.) A receiver could detect that it was out of synchronization by, for example, employing a "dead-man" timer that expires after a certain period of time, wherein the timer is reset with each valid packet. A time stamp could be hashed into the public sync field (see below) to preclude packet-retry attacks.

In one embodiment, a "sync field" is added to the header of each packet sent out by the sender. This sync field could appear in the clear or as part of an encrypted portion of the packet. Assuming that a sender and receiver have selected a random-number generator (RNG) and seed value, this combination of RNG and seed can be used to generate a random-number sequence (RNS). The RNS is then used to generate a sequence of source/destination IP pairs (and, if desired, discriminator fields and hardware source and destination addresses), as described above. It is not necessary, however, to generate the entire sequence (or the first N−1 values) in order to generate the Nth random number in the sequence; if the sequence index N is known, the random value corresponding to that index can be directly generated (see below). Different RNGs (and seeds) with different fundamental periods could be used to generate the source and destination IP sequences, but the basic concepts would still apply. For the sake of simplicity, the following discussion will assume that IP source and destination address pairs (only) are hopped using a single RNG sequencing mechanism.

In accordance with a "self-synchronization" feature, a sync field in each packet header provides an index (i.e., a sequence number) into the RNS that is being used to generate IP pairs. Plugging this index into the RNG that is being used to generate the RNS yields a specific random number value, which in turn yields a specific IP pair. That is, an IP pair can be generated directly from knowledge of the RNG, seed, and index number; it is not necessary, in this scheme, to generate the entire sequence of random numbers that precede the sequence value associated with the index number provided.

Since the communicants have presumably previously exchanged RNGs and seeds, the only new information that must be provided in order to generate an IP pair is the sequence number. If this number is provided by the sender in the packet header, then the receiver need only plug this number into the RNG in order to generate an IP pair—and thus verify that the IP pair appearing in the header of the packet is valid. In this scheme, if the sender and receiver lose synchronization, the receiver can immediately re-synchronize upon receipt of a single packet by simply comparing the IP pair in the packet header to the IP pair generated from the index number. Thus, synchronized communications can be resumed upon receipt of a single packet, making this scheme ideal for multicast communications. Taken to the extreme, it could obviate the need for synchronization tables entirely; that is, the sender and receiver could simply rely on the index number in the sync field to validate the IP pair on each packet, and thereby eliminate the tables entirely.

The aforementioned scheme may have some inherent security issues associated with it—namely, the placement of the sync field. If the field is placed in the outer header, then an interloper could observe the values of the field and their relationship to the IP stream. This could potentially compromise the algorithm that is being used to generate the IP-address sequence, which would compromise the security of the communications. If, however, the value is placed in the inner header, then the sender must decrypt the inner header before it can extract the sync value and validate the IP pair;

this opens up the receiver to certain types of denial-of-service (DoS) attacks, such as packet replay. That is, if the receiver must decrypt a packet before it can validate the IP pair, then it could potentially be forced to expend a significant amount of processing on decryption if an attacker simply retransmits previously valid packets. Other attack methodologies are possible in this scenario.

A possible compromise between algorithm security and processing speed is to split up the sync value between an inner (encrypted) and outer (unencrypted) header. That is, if the sync value is sufficiently long, it could potentially be split into a rapidly-changing part that can be viewed in the clear, and a fixed (or very slowly changing) part that must be protected. The part that can be viewed in the clear will be called the "public sync" portion and the part that must be protected will be called the "private sync" portion.

Both the public sync and private sync portions are needed to generate the complete sync value. The private portion, however, can be selected such that it is fixed or will change only occasionally. Thus, the private sync value can be stored by the recipient, thereby obviating the need to decrypt the header in order to retrieve it. If the sender and receiver have previously agreed upon the frequency with which the private part of the sync will change, then the receiver can selectively decrypt a single header in order to extract the new private sync if the communications gap that has led to lost synchronization has exceeded the lifetime of the previous private sync. This should not represent a burdensome amount of decryption, and thus should not open up the receiver to denial-of-service attack simply based on the need to occasionally decrypt a single header.

One implementation of this is to use a hashing function with a one-to-one mapping to generate the private and public sync portions from the sync value. This implementation is shown in FIG. **13**, where (for example) a first ISP **1302** is the sender and a second ISP **1303** is the receiver. (Other alternatives are possible from FIG. **13**.) A transmitted packet comprises a public or "outer" header **1305** that is not encrypted, and a private or "inner" header **1306** that is encrypted using for example a link key. Outer header **1305** includes a public sync portion while inner header **1306** contains the private sync portion. A receiving node decrypts the inner header using a decryption function **1307** in order to extract the private sync portion. This step is necessary only if the lifetime of the currently buffered private sync has expired. (If the currently-buffered private sync is still valid, then it is simply extracted from memory and "added" (which could be an inverse hash) to the public sync, as shown in step **1308**.) The public and decrypted private sync portions are combined in function **1308** in order to generate the combined sync **1309**. The combined sync (**1309**) is then fed into the RNG (**1310**) and compared to the IP address pair (**1311**) to validate or reject the packet.

An important consideration in this architecture is the concept of "future" and "past" where the public sync values are concerned. Though the sync values, themselves, should be random to prevent spoofing attacks, it may be important that the receiver be able to quickly identify a sync value that has already been sent—even if the packet containing that sync value was never actually received by the receiver. One solution is to hash a time stamp or sequence number into the public sync portion, which could be quickly extracted, checked, and discarded, thereby validating the public sync portion itself.

In one embodiment, packets can be checked by comparing the source/destination IP pair generated by the sync field with the pair appearing in the packet header. If (1) they match, (2) the time stamp is valid, and (3) the dead-man timer has expired, then re-synchronization occurs; otherwise, the packet is rejected. If enough processing power is available, the dead-man timer and synchronization tables can be avoided altogether, and the receiver would simply resynchronize (e.g., validate) on every packet.

The foregoing scheme may require large-integer (e.g., 160-bit) math, which may affect its implementation. Without such large-integer registers, processing throughput would be affected, thus potentially affecting security from a denial-of-service standpoint. Nevertheless, as large-integer math processing features become more prevalent, the costs of implementing such a feature will be reduced.

D. Other Synchronization Schemes

As explained above, if W or more consecutive packets are lost between a transmitter and receiver in a VPN (where W is the window size), the receiver's window will not have been updated and the transmitter will be transmitting packets not in the receiver's window. The sender and receiver will not recover synchronization until perhaps the random pairs in the window are repeated by chance. Therefore, there is a need to keep a transmitter and receiver in synchronization whenever possible and to re-establish synchronization whenever it is lost.

A "checkpoint" scheme can be used to regain synchronization between a sender and a receiver that have fallen out of synchronization. In this scheme, a checkpoint message comprising a random IP address pair is used for communicating synchronization information. In one embodiment, two messages are used to communicate synchronization information between a sender and a recipient:

1. SYNC_REQ is a message used by the sender to indicate that it wants to synchronize; and
2. SYNC_ACK is a message used by the receiver to inform the transmitter that it has been synchronized.

According to one variation of this approach, both the transmitter and receiver maintain three checkpoints (see FIG. **14**):

1. In the transmitter, ckpt_o ("checkpoint old") is the IP pair that was used to re-send the last SYNC_REQ packet to the receiver. In the receiver, ckpt_o ("checkpoint old") is the IP pair that receives repeated SYNC_REQ packets from the transmitter.
2. In the transmitter, ckpt_n ("checkpoint new") is the IP pair that will be used to send the next SYNC_REQ packet to the receiver. In the receiver, ckpt_n ("checkpoint new") is the IP pair that receives a new SYNC_REQ packet from the transmitter and which causes the receiver's window to be re-aligned, ckpt_o set to ckpt_n, a new ckpt_n to be generated and a new ckpt_r to be generated.
3. In the transmitter, ckpt_r is the IP pair that will be used to send the next SYNC_ACK packet to the receiver. In the receiver, ckpt_r is the IP pair that receives a new SYNC_ACK packet from the transmitter and which causes a new ckpt_n to be generated. Since SYNC_ACK is transmitted from the receiver ISP to the sender ISP, the transmitter ckpt_r refers to the ckpt_r of the receiver and the receiver ckpt_r refers to the ckpt_r of the transmitter (see FIG. **14**).

When a transmitter initiates synchronization, the IP pair it will use to transmit the next data packet is set to a predetermined value and when a receiver first receives a SYNC_REQ, the receiver window is updated to be centered on the transmitter's next IP pair. This is the primary mechanism for checkpoint synchronization.

Synchronization can be initiated by a packet counter (e.g., after every N packets transmitted, initiate a synchronization) or by a timer (every S seconds, initiate a synchronization) or a combination of both. See FIG. 15. From the transmitter's perspective, this technique operates as follows: (1) Each transmitter periodically transmits a "sync request" message to the receiver to make sure that it is in sync. (2) If the receiver is still in sync, it sends back a "sync ack" message. (If this works, no further action is necessary). (3) If no "sync ack" has been received within a period of time, the transmitter retransmits the sync request again. If the transmitter reaches the next checkpoint without receiving a "sync ack" response, then synchronization is broken, and the transmitter should stop transmitting. The transmitter will continue to send sync_reqs until it receives a sync_ack, at which point transmission is reestablished.

From the receiver's perspective, the scheme operates as follows: (1) when it receives a "sync request" request from the transmitter, it advances its window to the next checkpoint position (even skipping pairs if necessary), and sends a "sync ack" message to the transmitter. If sync was never lost, then the "jump ahead" really just advances to the next available pair of addresses in the table (i.e., normal advancement).

If an interloper intercepts the "sync request" messages and tries to interfere with communication by sending new ones, it will be ignored if the synchronization has been established or it it will actually help to re-establish synchronization.

A window is realigned whenever a re-synchronization occurs. This realignment entails updating the receiver's window to straddle the address pairs used by the packet transmitted immediately after the transmission of the SYNC_REQ packet. Normally, the transmitter and receiver are in synchronization with one another. However, when network events occur, the receiver's window may have to be advanced by many steps during resynchronization. In this case, it is desirable to move the window ahead without having to step through the intervening random numbers sequentially. (This feature is also desirable for the auto-sync approach discussed above).

E. Random Number Generator with a Jump-Ahead capability

An attractive method for generating randomly hopped addresses is to use identical random number generators in the transmitter and receiver and advance them as packets are transmitted and received. There are many random number generation algorithms that could be used. Each one has strengths and weaknesses for address hopping applications.

Linear congruential random number generators (LCRs) are fast, simple and well characterized random number generators that can be made to jump ahead n steps efficiently. An LCR generates random numbers $X_1, X_2, X_3 \ldots X_k$ starting with seed X_0 using a recurrence

$$X_i=(a X_{i-1}+b) \bmod c \tag{1}$$

where a, b and c define a particular LCR. Another expression for X_i,

$$X_i=((a^i(X_0+b)-b)/(a-1)) \bmod c \tag{2}$$

enables the jump-ahead capability. The factor a^i can grow very large even for modest i if left unfettered. Therefore some special properties of the modulo operation can be used to control the size and processing time required to compute (2). (2) can be rewritten as:

$$X_i=(a^i(X_0(a-1)+b)-b)/(a-1) \bmod c \tag{3}$$

It can be shown that:

$$(a^i(X_0(a-1)+b)-b)/(a-1) \bmod c=((a^i \bmod((a-1)c)(X_0(a-1)+b)-b)/(a-1)) \bmod c \tag{4}$$

$(X_0(a-1)+b)$ can be stored as $(X_0(a-1)+b) \bmod c$, b as b mod c and compute $a^i \bmod ((a-1)c)$(this requires O(log(i)) steps).

A practical implementation of this algorithm would jump a fixed distance, n, between synchronizations; this is tantamount to synchronizing every n packets. The window would commence n IP pairs from the start of the previous window. Using X_j^w, the random number at the jth checkpoint, as X_0 and n as i, a node can store $a^n \bmod((a-1)c)$ once per LCR and set

$$X_{j+1}^w=X_{n(j+1)}=((a^n \bmod((a+1)c)(X_j^w(a-1)+b)-b)/(a-1)) \bmod c, \tag{5}$$

to generate the random number for the j+1th synchronization. Using this construction, a node could jump ahead an arbitrary (but fixed) distance between synchronizations in a constant amount of time (independent of n).

Pseudo-random number generators, in general, and LCRs, in particular, will eventually repeat their cycles. This repetition may present vulnerability in the IP hopping scheme. An adversary would simply have to wait for a repeat to predict future sequences. One way of coping with this vulnerability is to create a random number generator with a known long cycle. A random sequence can be replaced by a new random number generator before it repeats. LCRs can be constructed with known long cycles. This is not currently true of many random number generators.

Random number generators can be cryptographically insecure. An adversary can derive the RNG parameters by examining the output or part of the output. This is true of LCGs. This vulnerability can be mitigated by incorporating an encryptor, designed to scramble the output as part of the random number generator. The random number generator prevents an adversary from mounting an attack—e.g., a known plaintext attack—against the encryptor.

F. Random Number Generator Example

Consider a RNG where a=31, b=4 and c=15. For this case equation (1) becomes:

$$X_i=(31 X_{i-1}+4) \bmod 15. \tag{6}$$

If one sets $X_0=1$, equation (6) will produce the sequence 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 0, 4, 8, 12. This sequence will repeat indefinitely. For a jump ahead of 3 numbers in this sequence $a^n=31^3=29791$, $c^*(a-1)=15^*30=450$ and $a^n \bmod ((a-1)c)=31^3 \bmod(15^*30)=29791 \bmod(450)=91$. Equation (5) becomes:

$$(91(X_i30+4)-4)/30) \bmod 15 \tag{7}$$

Table 1 shows the jump ahead calculations from (7). The calculations start at 5 and jump ahead 3.

TABLE 1

I	X_i	$(X_i30 + 4)$	$91 (X_i30 + 4) - 4$	$((91 (X_i30 + 4) - 4) / 30$	X_{i+3}
1	5	154	14010	467	2
4	2	64	5820	194	14
7	14	424	38580	1286	11
10	11	334	30390	1013	8
13	8	244	22200	740	5

G. Fast Packet Filter

Address hopping VPNs must rapidly determine whether a packet has a valid header and thus requires further processing, or has an invalid header (a hostile packet) and should be immediately rejected. Such rapid determinations will be referred to as "fast packet filtering." This capability protects the VPN from attacks by an adversary who streams hostile packets at the receiver at a high rate of speed in the hope of saturating the receiver's processor (a so-called "denial of service" attack). Fast packet filtering is an important feature for implementing VPNs on shared media such as Ethernet.

Assuming that all participants in a VPN share an unassigned "A" block of addresses, one possibility is to use an experimental "A" block that will never be assigned to any machine that is not address hopping on the shared medium. "A" blocks have a 24 bits of address that can be hopped as opposed to the 8 bits in "C" blocks. In this case a hopblock will be the "A" block. The use of the experimental "A" block is a likely option on an Ethernet because:

1. The addresses have no validity outside of the Ethernet and will not be routed out to a valid outside destination by a gateway.
2. There are 2^{24} (~16 million) addresses that can be hopped within each "A" block. This yields >280 trillion possible address pairs making it very unlikely that an adversary would guess a valid address. It also provides acceptably low probability of collision between separate VPNs (all VPNs on a shared medium independently generate random address pairs from the same "A" block).
3. The packets will not be received by someone on the Ethernet who is not on a VPN (unless the machine is in promiscuous mode) minimizing impact on non-VPN computers.

The Ethernet example will be used to describe one implementation of fast packet filtering. The ideal algorithm would quickly examine a packet header, determine whether the packet is hostile, and reject any hostile packets or determine which active IP pair the packet header matches. The problem is a classical associative memory problem. A variety of techniques have been developed to solve this problem (hashing, B-trees etc). Each of these approaches has its strengths and weaknesses. For instance, hash tables can be made to operate quite fast in a statistical sense, but can occasionally degenerate into a much slower algorithm. This slowness can persist for a period of time. Since there is a need to discard hostile packets quickly at all times, hashing would be unacceptable.

H. Presence Vector Alorithm

A presence vector is a bit vector of length 2^n that can be indexed by n-bit numbers (each ranging from 0 to 2^n-1). One can indicate the presence of k n-bit numbers (not necessarily unique), by setting the bits in the presence vector indexed by each number to 1. Otherwise, the bits in the presence vector are 0. An n-bit number, x, is one of the k numbers if and only if the x^{th} bit of the presence vector is 1. A fast packet filter can be implemented by indexing the presence vector and looking for a 1, which will be referred to as the "test."

For example, suppose one wanted to represent the number 135 using a presence vector. The 135^{th} bit of the vector would be set. Consequently, one could very quickly determine whether an address of 135 was valid by checking only one bit: the 135^{th} bit. The presence vectors could be created in advance corresponding to the table entries for the IP addresses. In effect, the incoming addresses can be used as indices into a long vector, making comparisons very fast. As each RNG generates a new address, the presence vector is updated to reflect the information. As the window moves, the presence vector is updated to zero out addresses that are no longer valid.

There is a trade-off between efficiency of the test and the amount of memory required for storing the presence vector(s). For instance, if one were to use the 48 bits of hopping addresses as an index, the presence vector would have to be 35 terabytes. Clearly, this is too large for practical purposes. Instead, the 48 bits can be divided into several smaller fields. For instance, one could subdivide the 48 bits into four 12-bit fields (see FIG. 16). This reduces the storage requirement to 2048 bytes at the expense of occasionally having to process a hostile packet. In effect, instead of one long presence vector, the decomposed address portions must match all four shorter presence vectors before further processing is allowed. (If the first part of the address portion doesn't match the first presence vector, there is no need to check the remaining three presence vectors).

A presence vector will have a 1 in the y^{th} bit if and only if one or more addresses with a corresponding field of y are active. An address is active only if each presence vector indexed by the appropriate sub-field of the address is 1.

Consider a window of 32 active addresses and 3 checkpoints. A hostile packet will be rejected by the indexing of one presence vector more than 99% of the time. A hostile packet will be rejected by the indexing of all 4 presence vectors more than 99.9999995% of the time. On average, hostile packets will be rejected in less than 1.02 presence vector index operations.

The small percentage of hostile packets that pass the fast packet filter will be rejected when matching pairs are not found in the active window or are active checkpoints. Hostile packets that serendipitously match a header will be rejected when the VPN software attempts to decrypt the header. However, these cases will be extremely rare. There are many other ways this method can be configured to arbitrate the space/speed tradeoffs.

I. Further Synchronization Enhancements

A slightly modified form of the synchronization techniques described above can be employed. The basic principles of the previously described checkpoint synchronization scheme remain unchanged. The actions resulting from the reception of the checkpoints are, however, slightly different. In this variation, the receiver will maintain between OoO ("Out of Order") and 2×WINDOW_SIZE+OoO active addresses (1≦OoO≦WINDOW_SIZE and WINDOW_SIZE≧1). OoO and WINDOW_SIZE are engineerable parameters, where OoO is the minimum number of addresses needed to accommodate lost packets due to events in the network or out of order arrivals and WINDOW_SIZE is the number of packets transmitted before a SYNC_REQ is issued. FIG. 17 depicts a storage array for a receiver's active addresses.

The receiver starts with the first 2×WINDOW_SIZE addresses loaded and active (ready to receive data). As packets are received, the corresponding entries are marked as "used" and are no longer eligible to receive packets. The transmitter maintains a packet counter, initially set to 0, containing the number of data packets transmitted since the last initial transmission of a SYNC_REQ for which SYNC_ACK has been received. When the transmitter packet counter equals WINDOW_SIZE, the transmitter generates a SYNC_REQ and does its initial transmission. When the receiver receives a SYNC_REQ corresponding to its current CKPT_N, it generates the next WINDOW_SIZE addresses and starts loading them in order starting at the first location after the last active address wrapping around to the beginning of the array after the end of the array has been reached. The receiver's array might look like FIG. 18 when a SYNC_REQ has been received. In this case a couple of packets have been either lost or will be received out of order when the SYNC_REQ is received.

FIG. 19 shows the receiver's array after the new addresses have been generated. If the transmitter does not receive a SYNC_ACK, it will re-issue the SYNC_REQ at regular intervals. When the transmitter receives a SYNC_ACK, the packet counter is decremented by WINDOW_SIZE. If the packet counter reaches 2×WINDOW_SIZE −OoO then the transmitter ceases sending data packets until the appropriate SYNC_ACK is finally received. The transmitter then resumes sending data packets. Future behavior is essentially a repetition of this initial cycle. The advantages of this approach are:

1. There is no need for an efficient jump ahead in the random number generator,
2. No packet is ever transmitted that does not have a corresponding entry in the receiver side
3. No timer based re-synchronization is necessary. This is a consequence of 2.
4. The receiver will always have the ability to accept data messages transmitted within OoO messages of the most recently transmitted message.

J. Distributed Transmission Path Variant

Another embodiment incorporating various inventive principles is shown in FIG. **20**. In this embodiment, a message transmission system includes a first computer **2001** in communication with a second computer **2002** through a network **2011** of intermediary computers. In one variant of this embodiment, the network includes two edge routers **2003** and **2004** each of which is linked to a plurality of Internet Service Providers (ISPs) **2005** through **2010**. Each ISP is coupled to a plurality of other ISPs in an arrangement as shown in FIG. **20**, which is a representative configuration only and is not intended to be limiting. Each connection between ISPs is labeled in FIG. **20** to indicate a specific physical transmission path (e.g., AD is a physical path that links ISP A (element **2005**) to ISP D (element **2008**)). Packets arriving at each edge router are selectively transmitted to one of the ISPs to which the router is attached on the basis of a randomly or quasi-randomly selected basis.

As shown in FIG. **21**, computer **2001** or edge router **2003** incorporates a plurality of link transmission tables **2100** that identify, for each potential transmission path through the network, valid sets of IP addresses that can be used to transmit the packet. For example, AD table **2101** contains a plurality of IP source/destination pairs that are randomly or quasi-randomly generated. When a packet is to be transmitted from first computer **2001** to second computer **2002**, one of the link tables is randomly (or quasi-randomly) selected, and the next valid source/destination address pair from that table is used to transmit the packet through the network. If path AD is randomly selected, for example, the next source/destination IP address pair (which is pre-determined to transmit between ISP A (element **2005**) and ISP B (element **2008**)) is used to transmit the packet. If one of the transmission paths becomes degraded or inoperative, that link table can be set to a "down" condition as shown in table **2105**, thus preventing addresses from being selected from that table. Other transmission paths would be unaffected by this broken link.

3. CONTINUATION-IN-PART IMPROVEMENTS

The following describes various improvements and features that can be applied to the embodiments described above. The improvements include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities. Each is discussed separately below.

A. Load Balancer

Various embodiments described above include a system in which a transmitting node and a receiving node are coupled through a plurality of transmission paths, and wherein successive packets are distributed quasi-randomly over the plurality of paths. See, for example, FIGS. **20** and **21** and accompanying description. The improvement extends this basic concept to encompass distributing packets across different paths in such a manner that the loads on the paths are generally balanced according to transmission link quality.

In one embodiment, a system includes a transmitting node and a receiving node that are linked via a plurality of transmission paths having potentially varying transmission quality. Successive packets are transmitted over the paths based on a weight value distribution function for each path. The rate that packets will be transmitted over a given path can be different for each path. The relative "health" of each transmission path is monitored in order to identify paths that have become degraded. In one embodiment, the health of each path is monitored in the transmitter by comparing the number of packets transmitted to the number of packet acknowledgements received. Each transmission path may comprise a physically separate path (e.g., via dial-up phone line, computer network, router, bridge, or the like), or may comprise logically separate paths contained within a broadband communication medium (e.g., separate channels in an FDM, TDM, CDMA, or other type of modulated or unmodulated transmission link).

When the transmission quality of a path falls below a predetermined threshold and there are other paths that can transmit packets, the transmitter changes the weight value used for that path, making it less likely that a given packet will be transmitted over that path. The weight will preferably be set no lower than a minimum value that keeps nominal traffic on the path. The weights of the other available paths are altered to compensate for the change in the affected path. When the quality of a path degrades to where the transmitter is turned off by the synchronization function (i.e., no packets are arriving at the destination), the weight is set to zero. If all transmitters are turned off, no packets are sent.

Conventional TCP/IP protocols include a "throttling" feature that reduces the transmission rate of packets when it is determined that delays or errors are occurring in transmission. In this respect, timers are sometimes used to determine whether packets have been received. These conventional techniques for limiting transmission of packets, however, do not involve multiple transmission paths between two nodes wherein transmission across a particular path relative to the others is changed based on link quality.

According to certain embodiments, in order to damp oscillations that might otherwise occur if weight distributions are changed drastically (e.g., according to a step function), a linear or an exponential decay formula can be applied to gradually decrease the weight value over time that a degrading path will be used. Similarly, if the health of a degraded path improves, the weight value for that path is gradually increased.

Transmission link health can be evaluated by comparing the number of packets that are acknowledged within the transmission window (see embodiments discussed above) to the number of packets transmitted within that window and by the state of the transmitter (i.e., on or off). In other words, rather than accumulating general transmission statistics over time for a path, one specific implementation uses the "windowing" concepts described above to evaluate transmission path health.

The same scheme can be used to shift virtual circuit paths from an "unhealthy" path to a "healthy" one, and to select a path for a new virtual circuit.

FIG. 22A shows a flowchart for adjusting weight values associated with a plurality of transmission links. It is assumed that software executing in one or more computer nodes executes the steps shown in FIG. 22A. It is also assumed that the software can be stored on a computer-readable medium such as a magnetic or optical disk for execution by a computer.

Beginning in step **2201**, the transmission quality of a given transmission path is measured. As described above, this measurement can be based on a comparison between the number of packets transmitted over a particular link to the number of packet acknowledgements received over the link (e.g., per unit time, or in absolute terms). Alternatively, the quality can be evaluated by comparing the number of packets that are acknowledged within the transmission window to the number of packets that were transmitted within that window. In yet another variation, the number of missed synchronization messages can be used to indicate link quality. Many other variations are of course possible.

In step **2202**, a check is made to determine whether more than one transmitter (e.g., transmission path) is turned on. If not, the process is terminated and resumes at step **2201**.

In step **2203**, the link quality is compared to a given threshold (e.g., 50%, or any arbitrary number). If the quality falls below the threshold, then in step **2207** a check is made to determine whether the weight is above a minimum level (e.g., 1%). If not, then in step **2209** the weight is set to the minimum level and processing resumes at step **2201**. If the weight is above the minimum level, then in step **2208** the weight is gradually decreased for the path, then in step **2206** the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are increased).

If in step **2203** the quality of the path was greater than or equal to the threshold, then in step **2204** a check is made to determine whether the weight is less than a steady-state value for that path. If so, then in step **2205** the weight is increased toward the steady-state value, and in step **2206** the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are decreased). If in step **2204** the weight is not less than the steady-state value, then processing resumes at step **2201** without adjusting the weights.

The weights can be adjusted incrementally according to various functions, preferably by changing the value gradually. In one embodiment, a linearly decreasing function is used to adjust the weights; according to another embodiment, an exponential decay function is used. Gradually changing the weights helps to damp oscillators that might otherwise occur if the probabilities were abruptly.

Although not explicitly shown in FIG. **22A** the process can be performed only periodically (e.g., according to a time schedule), or it can be continuously run, such as in a background mode of operation. In one embodiment, the combined weights of all potential paths should add up to unity (e.g., when the weighting for one path is decreased, the corresponding weights that the other paths will be selected will increase).

Adjustments to weight values for other paths can be prorated. For example, a decrease of 10% in weight value for one path could result in an evenly distributed increase in the weights for the remaining paths. Alternatively, weightings could be adjusted according to a weighted formula as desired (e.g., favoring healthy paths over less healthy paths). In yet another variation, the difference in weight value can be amortized over the remaining links in a manner that is proportional to their traffic weighting.

FIG. **22B** shows steps that can be executed to shut down transmission links where a transmitter turns off. In step **2210**, a transmitter shut-down event occurs. In step **2211**, a test is made to determine whether at least one transmitter is still turned on. If not, then in step **2215** all packets are dropped until a transmitter turns on. If in step **2211** at least one transmitter is turned on, then in step **2212** the weight for the path is set to zero, and the weights for the remaining paths are adjusted accordingly.

FIG. **23** shows a computer node **2301** employing various principles of the above-described embodiments. It is assumed that two computer nodes of the type shown in FIG. **23** communicate over a plurality of separate physical transmission paths. As shown in FIG. **23**, four transmission paths X1 through X4 are defined for communicating between the two nodes. Each node includes a packet transmitter **2302** that operates in accordance with a transmit table **2308** as described above. (The packet transmitter could also operate without using the IP-hopping features described above, but the following description assumes that some form of hopping is employed in conjunction with the path selection mechanism.). The computer node also includes a packet receiver **2303** that operates in accordance with a receive table **2309**, including a moving window W that moves as valid packets are received. Invalid packets having source and destination addresses that do not fall within window W are rejected.

As each packet is readied for transmission, source and destination IP addresses (or other discriminator values) are selected from transmit table **2308** according to any of the various algorithms described above, and packets containing these source/destination address pairs, which correspond to the node to which the four transmission paths are linked, are generated to a transmission path switch **2307**. Switch **2307**, which can comprise a software function, selects from one of the available transmission paths according to a weight distribution table **2306**. For example, if the weight for path **X1** is 0.2, then every fifth packet will be transmitted on path **X1**. A similar regime holds true for the other paths as shown. Initially, each link's weight value can be set such that it is proportional to its bandwidth, which will be referred to as its "steady-state" value.

Packet receiver **2303** generates an output to a link quality measurement function **2304** that operates as described above to determine the quality of each transmission path. (The input to packet receiver **2303** for receiving incoming packets is omitted for clarity). Link quality measurement function **2304** compares the link quality to a threshold for each transmission link and, if necessary, generates an output to weight adjustment function **2305**. If a weight adjustment is required, then the weights in table **2306** are adjusted accordingly, preferably according to a gradual (e.g., linearly or exponentially declining) function. In one embodiment, the weight values for all available paths are initially set to the same value, and only when paths degrade in quality are the weights changed to reflect differences.

Link quality measurement function **2304** can be made to operate as part of a synchronizer function as described above. That is, if resynchronization occurs and the receiver detects that synchronization has been lost (e.g., resulting in the synchronization window W being advanced out of sequence), that fact can be used to drive link quality measurement function **2304**. According to one embodiment, load balancing is performed using information garnered during the normal synchronization, augmented slightly to communicate link health from the receiver to the transmitter. The receiver maintains a count, MESS_R(W), of the messages received in synchronization window W. When it receives a synchronization request (SYNC_REQ) corresponding to the end of window W, the receiver includes counter MESS_R in the resulting synchronization acknowledgement (SYNC_ACK) sent back to the transmitter. This allows the transmitter to compare messages sent to messages received in order to asses the health of the link.

If synchronization is completely lost, weight adjustment function **2305** decreases the weight value on the affected path to zero. When synchronization is regained, the weight value for the affected path is gradually increased to its original value. Alternatively, link quality can be measured by evaluating the length of time required for the receiver to acknowledge a synchronization request. In one embodiment, separate transmit and receive tables are used for each transmission path.

When the transmitter receives a SYNC_ACK, the MESS_R is compared with the number of messages transmitted in a window (MESS_T). When the transmitter receives a SYNC_ACK, the traffic probabilities will be examined and adjusted if necessary. MESS_R is compared with the number of messages transmitted in a window (MESS_T). There are two possibilities:

1. If MESS_R is less than a threshold value, THRESH, then the link will be deemed to be unhealthy. If the transmitter was turned off, the transmitter is turned on and the weight P for that link will be set to a minimum value MIN. This will keep a trickle of traffic on the link for monitoring purposes until it recovers. If the transmitter was turned on, the weight P for that link will be set to:

$$P' = \alpha \times MIN + (1-\alpha) \times P \tag{1}$$

Equation 1 will exponentially damp the traffic weight value to MIN during sustained periods of degraded service.

2. If MESS_R for a link is greater than or equal to THRESH, the link will be deemed healthy. If the weight P for that link is greater than or equal to the steady state value S for that link, then P is left unaltered. If the weight P for that link is less than THRESH then P will be set to:

$$P' = \beta \times S + (1-\beta) \times P \tag{2}$$

where β is a parameter such that 0<=β<=1 that determines the damping rate of P.

Equation 2 will increase the traffic weight to S during sustained periods of acceptable service in a damped exponential fashion.

A detailed example will now be provided with reference to FIG. **24**. As shown in FIG. **24**, a first computer **2401** communicates with a second computer **2402** through two routers **2403** and **2404**. Each router is coupled to the other router

through three transmission links. As described above, these may be physically diverse links or logical links (including virtual private networks).

Suppose that a first link L1 can sustain a transmission bandwidth of 100 Mb/s and has a window size of 32; link L2 can sustain 75 Mb/s and has a window size of 24; and link L3 can sustain 25 Mb/s and has a window size of 8. The combined links can thus sustain 200Mb/s. The steady state traffic weights are 0.5 for link L1; 0.375 for link L2, and 0.125 for link L3. MIN=1Mb/s, THRESH=0.8 MESS_T for each link, α=0.75 and β=0.5. These traffic weights will remain stable until a link stops for synchronization or reports a number of packets received less than its THRESH. Consider the following sequence of events:

1. Link L1 receives a SYNC_ACK containing a MESS_R of 24, indicating that only 75% of the MESS_T (32) messages transmitted in the last window were successfully received. Link 1 would be below THRESH (0.8). Consequently, link L1's traffic weight value would be reduced to 0.12825, while link L2's traffic weight value would be increased to 0.65812 and link L3's traffic weight value would be increased to 0.217938.
2. Link L2 and L3 remained healthy and link L1 stopped to synchronize. Then link L1's traffic weight value would be set to 0, link L2's traffic weight value would be set to 0.75, and link L33's traffic weight value would be set to 0.25.
3. Link L1 finally received a SYNC_ACK containing a MESS_R of 0 indicating that none of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be below THRESH. Link L1's traffic weight value would be increased to 0.005, link L2's traffic weight value would be decreased to 0.74625, and link L3's traffic weight value would be decreased to 0.24875.
4. Link L1 received a SYNC_ACK containing a MESS_R of 32 indicating that 100% of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.2525, while link L2's traffic weight value would be decreased to 0.560625 and link L3's traffic weight value would be decreased to 0.186875.
5. Link L1 received a SYNC_ACK containing a MESS_R of 32 indicating that 100% of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.37625; link L2's traffic weight value would be decreased to 0.4678125, and link L3's traffic weight value would be decreased to 0.1559375.
6. Link L1 remains healthy and the traffic probabilities approach their steady state traffic probabilities.

B. Use of a DNS Proxy to Transparently Create Virtual Private Networks

A second improvement concerns the automatic creation of a virtual private network (VPN) in response to a domain-name server look-up function.

Conventional Domain Name Servers (DNSs) provide a look-up function that returns the IP address of a requested computer or host. For example, when a computer user types in the web name "Yahoo.com," the user's web browser transmits a request to a DNS, which converts the name into a four-part IP address that is returned to the user's browser and then used by the browser to contact the destination web site.

This conventional scheme is shown in FIG. **25**. A user's computer **2501** includes a client application **2504** (for example, a web browser) and an IP protocol stack **2505**. When the user enters the name of a destination host, a request DNS REQ is made (through IP protocol stack **2505**) to a DNS **2502** to look up the IP address associated with the name. The DNS returns the IP address DNS RESP to client application **2504**, which is then able to use the IP address to communicate with the host **2503** through separate transactions such as PAGE REQ and PAGE RESP.

In the conventional architecture shown in FIG. **25**, nefarious listeners on the Internet could intercept the DNS REQ and DNS RESP packets and thus learn what IP addresses the user was contacting. For example, if a user wanted to set up a secure communication path with a web site having the name "Target.com," when the user's browser contacted a DNS to find the IP address for that web site, the true IP address of that web site would be revealed over the Internet as part of the DNS inquiry. This would hamper anonymous communications on the Internet.

One conventional scheme that provides secure virtual private networks over the Internet provides the DNS server with the public keys of the machines that the DNS server has the addresses for. This allows hosts to retrieve automatically the public keys of a host that the host is to communicate with so that the host can set up a VPN without having the user enter the public key of the destination host. One implementation of this standard is presently being developed as part of the FreeS/WAN project(RFC **2535**).

The conventional scheme suffers from certain drawbacks. For example, any user can perform a DNS request. Moreover, DNS requests resolve to the same value for all users.

According to certain aspects of the invention, a specialized DNS server traps DNS requests and, if the request is from a special type of user (e.g., one for which secure communication services are defined), the server does not return the true IP address of the target node, but instead automatically sets up a virtual private network between the target node and the user. The VPN is preferably implemented using the IP address "hopping" features of the basic invention described above, such that the true identity of the two nodes cannot be determined even if packets during the communication are intercepted. For DNS requests that are determined to not require secure services (e.g., an unregistered user), the DNS server transparently "passes through" the request to provide a normal look-up function and return the IP address of the target web server, provided that the requesting host has permissions to resolve unsecured sites. Different users who make an identical DNS request could be provided with different results.

FIG. **26** shows a system employing various principles summarized above. A user's computer **2601** includes a conventional client (e.g., a web browser) **2605** and an IP protocol stack **2606** that preferably operates in accordance with an IP hopping function **2607** as outlined above. A modified DNS server **2602** includes a conventional DNS server function **2609** and a DNS proxy **2610**. A gatekeeper server **2603** is interposed between the modified DNS server and a secure target site **2704**. An "unsecure" target site **2611** is also accessible via conventional IP protocols.

According to one embodiment, DNS proxy **2610** intercepts all DNS lookup functions from client **2605** and determines whether access to a secure site has been requested. If access to a secure site has been requested (as determined, for example, by a domain name extension, or by reference to an internal table of such sites), DNS proxy **2610** determines whether the user has sufficient security privileges to access the site. If so, DNS proxy **2610** transmits a message to gatekeeper **2603** requesting that a virtual private network be created between user computer **2601** and secure target site **2604**. In one embodiment, gatekeeper **2603** creates "hopblocks" to be used by computer **2601** and secure target site **2604** for secure communication. Then, gatekeeper **2603** communicates these to user computer **2601**. Thereafter, DNS proxy **2610** returns to user computer **2601** the resolved address passed to it by the gatekeeper (this address could be different from the actual target computer) **2604**, preferably using a secure administrative VPN. The address that is returned need not be the actual address of the destination computer.

Had the user requested lookup of a non-secure web site such as site **2611**, DNS proxy would merely pass through to conventional DNS server **2609** the look-up request, which would be handled in a conventional manner, returning the IP address of non-secure web site **2611**. If the user had requested lookup of a secure web site but lacked credentials to create such a connection, DNS proxy **2610** would return a "host unknown" error to the user. In this manner, different users requesting access to the same DNS name could be provided with different look-up results.

Gatekeeper **2603** can be implemented on a separate computer (as shown in FIG. **26**) or as a function within modified DNS server **2602**. In general, it is anticipated that gatekeeper **2703** facilitates the allocation and exchange of information needed to communicate securely, such as using "hopped" IP addresses. Secure hosts such as site **2604** are assumed to be equipped with a secure communication function such as an IP hopping function **2608**.

It will be appreciated that the functions of DNS proxy **2610** and DNS server **2609** can be combined into a single server for convenience. Moreover, although element **2602** is shown as combining the functions of two servers, the two servers can be made to operate independently.

FIG. **27** shows steps that can be executed by DNS proxy server **2610** to handle requests for DNS look-up for secure hosts. In step **2701**, a DNS look-up request is received for a target host. In step **2702**, a check is made to determine whether access to a secure host was requested. If not, then in step **2703** the DNS request is passed to conventional DNS server **2609**, which looks up the IP address of the target site and returns it to the user's application for further processing.

In step **2702**, if access to a secure host was requested, then in step **2704** a further check is made to determine whether the user is authorized to connect to the secure host. Such a check can be made with reference to an internally stored list of authorized IP addresses, or can be made by communicating with gatekeeper **2603** (e.g., over an "administrative" VPN that is secure). It will be appreciated that different levels of security can also be provided for different categories of hosts. For example, some sites may be designated as having a certain security level, and the security level of the user requesting access must match that security level. The user's security level can also be determined by transmitting a request message back to the user's computer requiring that it prove that it has sufficient privileges.

If the user is not authorized to access the secure site, then a "host unknown" message is returned (step **2705**). If the user has sufficient security privileges, then in step **2706** a secure VPN is established between the user's computer and the secure target site. As described above, this is preferably done by allocating a hopping regime that will be carried out between the user's computer and the secure target site, and is preferably performed transparently to the user (i.e., the user need not be involved in creating the secure link). As described in various embodiments of this application, any of various fields can be "hopped" (e.g., IP source/destination addresses; a field in the header; etc.) in order to communicate securely.

Some or all of the security functions can be embedded in gatekeeper **2603**, such that it handles all requests to connect to secure sites. In this embodiment, DNS proxy **2610** communicates with gatekeeper **2603** to determine (preferably over a secure administrative VPN) whether the user has access to a particular web site. Various scenarios for implementing these features are described by way of example below:

Scenario #1: Client has permission to access target computer, and gatekeeper has a rule to make a VPN for the client. In this scenario, the client's DNS request would be received by the DNS proxy server **2610**, which would forward the request to gatekeeper **2603**. The gatekeeper would establish a VPN between the client and the requested target. The gatekeeper would provide the address of the destination to the DNS proxy, which would then return the resolved name as a result. The resolved address can be transmitted back to the client in a secure administrative VPN.

Scenario #2: Client does not have permission to access target computer. In this scenario, the client's DNS request would be received by the DNS proxy server **2610**, which would forward the request to gatekeeper **2603**. The gatekeeper would reject the request, informing DNS proxy server **2610** that it was unable to find the target computer. The DNS proxy **2610** would then return a "host unknown" error message to the client.

Scenario #3: Client has permission to connect using a normal non-VPN link, and the gatekeeper does not have a rule to set up a VPN for the client to the target site. In this scenario, the client's DNS request is received by DNS proxy server **2610**, which would check its rules and determine that no VPN is needed. Gatekeeper **2603** would then inform the DNS proxy server to forward the request to conventional DNS server **2609**, which would resolve the request and return the result to the DNS proxy server and then back to the client.

Scenario #4: Client does not have permission to establish a normal/non-VPN link, and the gatekeeper does not have a rule to make a VPN for the client to the target site. In this scenario, the DNS proxy server would receive the client's DNS request and forward it to gatekeeper **2603**. Gatekeeper **2603** would determine that no special VPN was needed, but that the client is not authorized to communicate with non-VPN members. The gatekeeper would reject the request, causing DNS proxy server **2610** to return an error message to the client.

C. Large Link to Small Link Bandwidth Management

One feature of the basic architecture is the ability to prevent so-called "denial of service" attacks that can occur if a computer hacker floods a known Internet node with packets, thus preventing the node from communicating with other nodes. Because IP addresses or other fields are "hopped" and packets arriving with invalid addresses are quickly discarded, Internet nodes are protected against flooding targeted at a single IP address.

In a system in which a computer is coupled through a link having a limited bandwidth (e.g., an edge router) to a node that can support a much higher-bandwidth link (e.g., an Internet Service Provider), a potential weakness could be exploited by a determined hacker. Referring to FIG. **28**, suppose that a first host computer **2801** is communicating with a second host computer **2804** using the IP address hopping principles described above. The first host computer is coupled through an edge router **2802** to an Internet Service Provider (ISP) **2803** through a low bandwidth link (LOW BW), and is in turn coupled to second host computer **2804** through parts of the Internet through a high bandwidth link (HIGH BW). In this architecture, the ISP is able to support a high bandwidth to the internet, but a much lower bandwidth to the edge router **2802**.

Suppose that a computer hacker is able to transmit a large quantity of dummy packets addressed to first host computer **2801** across high bandwidth link HIGH BW. Normally, host computer **2801** would be able to quickly reject the packets since they would not fall within the acceptance window permitted by the IP address hopping scheme. However, because the packets must travel across low bandwidth link LOW BW, the packets overwhelm the lower bandwidth link before they are received by host computer **2801**. Consequently, the link to host computer **2801** is effectively flooded before the packets can be discarded.

According to one inventive improvement, a "link guard" function **2805** is inserted into the high-bandwidth node (e.g., ISP **2803**) that quickly discards packets destined for a low-bandwidth target node if they are not valid packets. Each packet destined for a low-bandwidth node is cryptographically authenticated to determine whether it belongs to a VPN. If it is not a valid VPN packet, the packet is discarded at the high-bandwidth node. If the packet is authenticated as belonging to a VPN, the packet is passed with high preference. If the packet is a valid non-VPN packet, it is passed with a lower quality of service (e.g., lower priority).

In one embodiment, the ISP distinguishes between VPN and non-VPN packets using the protocol of the packet. In the case of IPSEC [rfc **2401**], the packets have IP protocols **420** and **421**. In the case of the TARP VPN, the packets will have an IP protocol that is not yet defined. The ISP's link guard, **2805**, maintains a table of valid VPNs which it uses to validate whether VPN packets are cryptographically valid.

According to one embodiment, packets that do not fall within any hop windows used by nodes on the low-bandwidth link are rejected, or are sent with a lower quality of service. One approach for doing this is to provide a copy of the IP hopping tables used by the low-bandwidth nodes to the high-bandwidth node, such that both the high-bandwidth and low-bandwidth nodes track hopped packets (e.g., the high-bandwidth node moves its hopping window as valid packets are received). In such a scenario, the high-bandwidth node discards packets that do not fall within the hopping window before they are transmitted over the low-bandwidth link. Thus, for example, ISP **2903** maintains a copy **2910** of the receive table used by host computer **2901**. Incoming packets that do not fall within this receive table are discarded. According to a different embodiment, link guard **2805** validates each VPN packet using a keyed hashed message authentication code (HMAC) [rfc **2104**]. According to another embodiment, separate VPNs (using, for example, hopblocks) can be established for communicating between the low-bandwidth node and the high-bandwidth node (i.e., packets arriving at the high-bandwidth node are converted into different packets before being transmitted to the low-bandwidth node).

As shown in FIG. **29**, for example, suppose that a first host computer **2900** is communicating with a second host computer **2902** over the Internet, and the path includes a high bandwidth link HIGH BW to an ISP **2901** and a low bandwidth link LOW BW through an edge router **2904**. In accordance with the basic architecture described above, first host computer **2900** and second host computer **2902** would exchange hopblocks (or a hopblock algorithm) and would be able to create matching transmit and receive tables **2905**, **2906**, **2912** and **2913**. Then in accordance with the basic architecture, the two computers would transmit packets having seemingly random IP source and destination addresses, and each would move a corresponding hopping window in its receive table as valid packets were received.

Suppose that a nefarious computer hacker **2903** was able to deduce that packets having a certain range of IP addresses (e.g., addresses 100 to 200 for the sake of simplicity) are being transmitted to ISP **2901**, and that these packets are being forwarded over a low-bandwidth link. Hacker computer **2903** could thus "flood" packets having addresses falling into the range 100 to 200, expecting that they would be forwarded along low bandwidth link LOW BW, thus causing the low bandwidth link to become overwhelmed. The fast packet reject mechanism in first host computer **3000** would be of little use in rejecting these packets, since the low bandwidth link was effectively jammed before the packets could be rejected. In accordance with one aspect of the improvement, however, VPN link guard **2911** would prevent the attack from impacting the performance of VPN traffic because the packets would either be rejected as invalid VPN packets or given a lower quality of service than VPN traffic over the lower bandwidth link. A denial-of-service flood attack could, however, still disrupt non-VPN traffic.

According to one embodiment of the improvement, ISP **2901** maintains a separate VPN with first host computer **2900**, and thus translates packets arriving at the ISP into packets having a different IP header before they are transmitted to host computer **2900**. The cryptographic keys used to authenticate VPN packets at the link guard **2911** and the cryptographic keys used to encrypt and decrypt the VPN packets at host **2902** and host **2901** can be different, so that link guard **2911** does not have access to the private host data; it only has the capability to authenticate those packets.

According to yet a third embodiment, the low-bandwidth node can transmit a special message to the high-bandwidth node instructing it to shut down all transmissions on a particular IP address, such that only hopped packets will pass through to the low-bandwidth node. This embodiment would prevent a hacker from flooding packets using a single IP address. According to yet a fourth embodiment, the high-bandwidth node can be configured to discard packets transmitted to the low-bandwidth node if the transmission rate exceeds a certain predetermined threshold for any given IP address; this would allow hopped packets to go through. In this respect, link guard **2911** can be used to detect that the rate of packets on a given IP address are exceeding a threshold rate; further packets addressed to that same IP address would be dropped or transmitted at a lower priority (e.g., delayed).

D. Traffic Limiter

In a system in which multiple nodes are communicating using "hopping" technology, a treasonous insider could internally flood the system with packets. In order to prevent this possibility, one inventive improvement involves setting up "contracts" between nodes in the system, such that a receiver can impose a bandwidth limitation on each packet sender. One technique for doing this is to delay acceptance of a checkpoint synchronization request from a sender until a certain time period (e.g., one minute) has elapsed. Each receiver can effectively control the rate at which its hopping window moves by delaying "SYNC ACK" responses to "SYNC_REQ" messages.

A simple modification to the checkpoint synchronizer will serve to protect a receiver from accidental or deliberate overload from an internally treasonous client. This modification is based on the observation that a receiver will not update its

tables until a SYNC_REQ is received on hopped address CKPT_N. It is a simple matter of deferring the generation of a new CKPT_N until an appropriate interval after previous checkpoints.

Suppose a receiver wished to restrict reception from a transmitter to 100 packets a second, and that checkpoint synchronization messages were triggered every 50 packets. A compliant transmitter would not issue new SYNC_REQ messages more often than every 0.5 seconds. The receiver could delay a non-compliant transmitter from synchronizing by delaying the issuance of CKPT_N for 0.5 second after the last SYNC_REQ was accepted.

In general, if M receivers need to restrict N transmitters issuing new SYNC_REQ messages after every W messages to sending R messages a second in aggregate, each receiver could defer issuing a new CKPT_N until M×N×W/R seconds have elapsed since the last SYNC_REQ has been received and accepted. If the transmitter exceeds this rate between a pair of checkpoints, it will issue the new checkpoint before the receiver is ready to receive it, and the SYNC_REQ will be discarded by the receiver. After this, the transmitter will re-issue the SYNC_REQ every **T1** seconds until it receives a SYNC_ACK. The receiver will eventually update CKPT_N and the SYNC_REQ will be acknowledged. If the transmission rate greatly exceeds the allowed rate, the transmitter will stop until it is compliant. If the transmitter exceeds the allowed rate by a little, it will eventually stop after several rounds of delayed synchronization until it is in compliance. Hacking the transmitter's code to not shut off only permits the transmitter to lose the acceptance window. In this case it can recover the window and proceed only after it is compliant again.

Two practical issues should be considered when implementing the above scheme:

1. The receiver rate should be slightly higher than the permitted rate in order to allow for statistical fluctuations in traffic arrival times and non-uniform load balancing.

2. Since a transmitter will rightfully continue to transmit for a period after a SYNC_REQ is transmitted, the algorithm above can artificially reduce the transmitter's bandwidth. If events prevent a compliant transmitter from synchronizing for a period (e.g. the network dropping a SYNC_REQ or a SYNC_ACK) a SYNC_REQ will be accepted later than expected. After this, the transmitter will transmit fewer than expected messages before encountering the next checkpoint. The new checkpoint will not have been activated and the transmitter will have to retransmit the SYNC_REQ. This will appear to the receiver as if the transmitter is not compliant. Therefore, the next checkpoint will be accepted late from the transmitter's perspective. This has the effect of reducing the transmitter's allowed packet rate until the transmitter transmits at a packet rate below the agreed upon rate for a period of time.

To guard against this, the receiver should keep track of the times that the last C SYNC_REQs were received and accepted and use the minimum of M×N×W/R seconds after the last SYNC_REQ has been received and accepted, 2×M×N×W/R seconds after next to the last SYNC_REQ has been received and accepted, C×M×N×W/R seconds after $(C-1)^{th}$ to the last SYNC_REQ has been received, as the time to activate CKPT_N. This prevents the receiver from inappropriately limiting the transmitter's packet rate if at least one out of the last C SYNC_REQs was processed on the first attempt.

FIG. **30** shows a system employing the above-described principles. In FIG. **30**, two computers **3000** and **3001** are assumed to be communicating over a network N in accordance with the "hopping" principles described above (e.g., hopped IP addresses, discriminator values, etc.). For the sake of simplicity, computer **3000** will be referred to as the receiving computer and computer **3001** will be referred to as the transmitting computer, although full duplex operation is of course contemplated. Moreover, although only a single transmitter is shown, multiple transmitters can transmit to receiver **3000**.

As described above, receiving computer **3000** maintains a receive table **3002** including a window W that defines valid IP address pairs that will be accepted when appearing in incoming data packets. Transmitting computer **3001** maintains a transmit table **3003** from which the next IP address pairs will be selected when transmitting a packet to receiving computer **3000**. (For the sake of illustration, window W is also illustrated with reference to transmit table **3003**). As transmitting computer moves through its table, it will eventually generate a SYNC_REQ message as illustrated in function **3010**. This is a request to receiver **3000** to synchronize the receive table **3002**, from which transmitter **3001** expects a response in the form of a CKPT_N (included as part of a SYNC_ACK message). If transmitting computer **3001** transmits more messages than its allotment, it will prematurely generate the SYNC_REQ message. (If it has been altered to remove the SYNC_REQ message generation altogether, it will fall out of synchronization since receiver **3000** will quickly reject packets that fall outside of window W, and the extra packets generated by transmitter **3001** will be discarded).

In accordance with the improvements described above, receiving computer **3000** performs certain steps when a SYNC_REQ message is received, as illustrated in FIG. **30**. In step **3004**, receiving computer **3000** receives the SYNC_REQ message. In step **3005**, a check is made to determine whether the request is a duplicate. If so, it is discarded in step **3006**. In step **3007**, a check is made to determine whether the SYNC_REQ received from transmitter **3001** was received at a rate that exceeds the allowable rate R (i.e., the period between the time of the last SYNC_REQ message). The value R can be a constant, or it can be made to fluctuate as desired. If the rate exceeds R, then in step **3008** the next activation of the next CKPT_N hopping table entry is delayed by W/R seconds after the last SYNC_REQ has been accepted.

Otherwise, if the rate has not been exceeded, then in step **3109** the next CKPT_N value is calculated and inserted into the receiver's hopping table prior to the next SYNC_REQ from the transmitter **3101**. Transmitter **3101** then processes the SYNC_REQ in the normal manner.

E. Signaling Synchronizer

In a system in which a large number of users communicate with a central node using secure hopping technology, a large amount of memory must be set aside for hopping tables and their supporting data structures. For example, if one million subscribers to a web site occasionally communicate with the web site, the site must maintain one million hopping tables, thus using up valuable computer resources, even though only a small percentage of the users may actually be using the system at any one time. A desirable solution would be a system that permits a certain maximum number of simultaneous links to be maintained, but which would "recognize" millions of registered users at any one time. In other words, out of a population of a million registered users, a few thousand at a time could simultaneously communicate with a central server, without requiring that the server maintain one million hopping tables of appreciable size.

One solution is to partition the central node into two nodes: a signaling server that performs session initiation for user log-on and log-off (and requires only minimally sized tables), and a transport server that contains larger hopping tables for the users. The signaling server listens for the millions of known users and performs a fast-packet reject of other (bogus) packets. When a packet is received from a known user, the signaling server activates a virtual private link (VPL) between the user and the transport server, where hopping tables are allocated and maintained. When the user logs onto the signaling server, the user's computer is provided with hop tables for communicating with the transport server, thus activating the VPL. The VPLs can be torn down when they become inactive for a time period, or they can be torn down upon user log-out. Communication with the signaling server to allow user log-on and log-off can be accomplished using a specialized version of the checkpoint scheme described above.

FIG. **31** shows a system employing certain of the above-described principles. In FIG. **31**, a signaling server **3101** and a transport server **3102** communicate over a link. Signaling server **3101** contains a large number of small tables **3106** and **3107** that contain enough information to authenticate a communication request with one or more clients **3103** and **3104**. As described in more detail below, these small tables may advantageously be constructed as a special case of the synchronizing checkpoint tables described previously. Transport server **3102**, which is preferably a separate computer in communication with signaling server **3101**, contains a smaller number of larger hopping tables **3108**, **3109**, and **3110** that can be allocated to create a VPN with one of the client computers.

According to one embodiment, a client that has previously registered with the system (e.g., via a system administration function, a user registration procedure, or some other method) transmits a request for information from a computer (e.g., a web site). In one variation, the request is made using a "hopped" packet, such that signaling server **3101** will quickly reject invalid packets from unauthorized computers such as hacker computer **3105**. An "administrative" VPN can be established between all of the clients and the signaling server in order to ensure that a hacker cannot flood signaling server **3101** with bogus packets. Details of this scheme are provided below.

Signaling server **3101** receives the request **3111** and uses it to determine that client **3103** is a validly registered user. Next, signaling server **3101** issues a request to transport server **3102** to allocate a hopping table (or hopping algorithm or other regime) for the purpose of creating a VPN with client **3103**. The allocated hopping parameters are returned to signaling server **3101** (path **3113**), which then supplies the hopping parameters to client **3103** via path **3114**, preferably in encrypted form.

Thereafter, client **3103** communicates with transport server **3102** using the normal hopping techniques described above. It will be appreciated that although signaling server **3101** and transport server **3102** are illustrated as being two separate computers, they could of course be combined into a single computer and their functions performed on the single computer. Alternatively, it is possible to partition the functions shown in FIG. **31** differently from as shown without departing from the inventive principles.

One advantage of the above-described architecture is that signaling server **3101** need only maintain a small amount of information on a large number of potential users, yet it retains the capability of quickly rejecting packets from unauthorized users such as hacker computer **3105**. Larger data tables needed to perform the hopping and synchronization functions are instead maintained in a transport server **3102**, and a smaller number of these tables are needed since they are only allocated for "active" links. After a VPN has become inactive for a certain time period (e.g., one hour), the VPN can be automatically torn down by transport server **3102** or signaling server **3101**.

A more detailed description will now be provided regarding how a special case of the checkpoint synchronization feature can be used to implement the signaling scheme described above.

The signaling synchronizer may be required to support many (millions) of standing, low bandwidth connections. It therefore should minimize per-VPL memory usage while providing the security offered by hopping technology. In order to reduce memory usage in the signaling server, the data hopping tables can be completely eliminated and data can be carried as part of the SYNC_REQ message. The table used by the server side (receiver) and client side (transmitter) is shown schematically as element **3106** in FIG. **31**.

The meaning and behaviors of CKPT_N, CKPT_O and CKPT_R remain the same from the previous description, except that CKPT_N can receive a combined data and SYNC_REQ message or a SYNC_REQ message without the data.

The protocol is a straightforward extension of the earlier synchronizer. Assume that a client transmitter is on and the tables are synchronized. The initial tables can be generated "out of band." For example, a client can log into a web server to establish an account over the Internet. The client will receive keys etc encrypted over the Internet. Meanwhile, the server will set up the signaling VPN on the signaling server.

Assuming that a client application wishes to send a packet to the server on the client's standing signaling VPL:

1. The client sends the message marked as a data message on the inner header using the transmitter's CKPT_N address. It turns the transmitter off and starts a timer T1 noting CKPT_O. Messages can be one of three types: DATA, SYNC_REQ and SYNC_ACK. In the normal algorithm, some potential problems can be prevented by identifying each message type as part of the encrypted inner header field. In this algorithm, it is important to distinguish a data packet and a SYNC_REQ in the signaling synchronizer since the data and the SYNC_REQ come in on the same address.
2. When the server receives a data message on its CKPT_N, it verifies the message and passes it up the stack. The message can be verified by checking message type and and other information (i.e user credentials) contained in the inner header It replaces its CKPT_O with CKPT_N and generates the next CKPT_N. It updates its transmitter side

CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.

3. When the client side receiver receives a SYNC_ACK on its CKPT_R with a payload matching its transmitter side CKPT_O and the transmitter is off, the transmitter is turned on and the receiver side CKPT_R is updated. If the SYNC_ACK's payload does not match the transmitter side CKPT_O or the transmitter is on, the SYNC_ACK is simply discarded.
4. T1 expires: If the transmitter is off and the client's transmitter side CKPT_O matches the CKPT_O associated with the timer, it starts timer T1 noting CKPT_O again, and a SYNC_REQ is sent using the transmitter's CKPT_O address. Otherwise, no action is taken.
5. When the server receives a SYNC_REQ on its CKPT_N, it replaces its CKPT_O with CKPT_N and generates the next CKPT_N. It updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.
6. When the server receives a SYNC_REQ on its CKPT_O, it updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.

FIG. **32** shows message flows to highlight the protocol. Reading from top to bottom, the client sends data to the server using its transmitter side CKPT_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT_N into CKPT_O and updates CKPT_N. This message is successfully received and a passed up the stack. It also synchronizes the receiver i.e, the server loads CKPT_N into CKPT_O and generates a new CKPT_N, it generates a new CKPT_R in the server side transmitter and transmits a SYNC_ACK containing the server side receiver's CKPT_O the server. The SYNC_ACK is successfully received at the client. The client side receiver's CKPT_R is updated, the transmitter is turned on and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

Next, the client sends data to the server using its transmitter side CKPT_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT_N into CKPT_O and updates CKPT_N. This message is lost. The client side timer expires and as a result a SYNC_REQ is transmitted on the client side transmitter's CKPT_O (this will keep happening until the SYNC_ACK has been received at the client). The SYNC_REQ is successfully received at the server. It synchronizes the receiver i.e, the server loads CKPT_N into CKPT_O and generates a new CKPT_N, it generates an new CKPT_R in the server side transmitter and transmits a SYNC_ACK containing the server side receiver's CKPT_O the server. The SYNC_ACK is successfully received at the client. The client side receiver's CKPT_R is updated, the transmitter is turned off and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

There are numerous other scenarios that follow this flow. For example, the SYNC_ACK could be lost. The transmitter would continue to re-send the SYNC_REQ until the receiver synchronizes and responds.

The above-described procedures allow a client to be authenticated at signaling server **3201** while maintaining the ability of signaling server **3201** to quickly reject invalid packets, such as might be generated by hacker computer **3205**. In various embodiments, the signaling synchronizer is really a derivative of the synchronizer. It provides the same protection as the hopping protocol, and it does so for a large number of low bandwidth connections.

We claim:

1. A data processing device, comprising memory storing a domain name server (DNS) proxy module that intercepts DNS requests sent by a client and, for each intercepted DNS request, performs the steps of:

(i) determining whether the intercepted DNS request corresponds to a secure server;

(ii) when the intercepted DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a nonsecure computer, and

(iii) when the intercepted DNS request corresponds to a secure server, automatically initiating an encrypted channel between the client and the secure server.

2. The data processing device of claim 1, wherein step (iii) comprises the steps of:
(a) determining whether the client is authorized to access the secure server; and
(b) when the client is authorized to access the secure server, sending a request to the secure server to establish an encrypted channel between the secure server and the client.

3. The data processing device of claim 2, wherein step (iii) further comprises the step of:
(c) when the client is not authorized to access the secure server, returning a host unknown error message to the client.

4. The data processing device of claim 3, wherein the client comprises a web browser into which a user enters a URL resulting in the DNS request.

5. The data processing device of claim 1, wherein automatically initiating the encrypted channel between the client and the secure sewer comprises establishing an IF address hopping scheme between the client and the secure server.

6. The data processing device of claim 1, wherein automatically initiating the encrypted channel between the client and the secure server avoids sending a true IP address of the secure server to the client.

7. A computer readable medium storing a domain name server (DNS) proxy module comprised of computer readable instructions that, when executed, cause a data processing device to perform the steps of:
(i) intercepting a DNS request sent by a client;
(ii) determining whether the intercepted DNS request corresponds to a secure server;
(iii) when the intercepted DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a nonsecure computer; and
(iv) when the intercepted DNS request corresponds to a secure server, automatically initiating an encrypted channel between the client and the secure server.

8. The computer readable medium of claim 7, wherein step (iv) comprises the steps of
(a) determining whether the client is authorized to access the secure server, and
(b) when the client is authorized to access the secure server, sending a request to the secure sewer to establish an encrypted channel between the secure sewer and the client.

9. The computer readable medium of claim 8, wherein step (iv) further comprises the step of:
(c) when the client is not authorized to access the secure server, returning a host unknown error message to the client.

10. The computer readable medium of claim 9, wherein the client comprises a web browser into which a user enters a URL resulting in the DNS request.

11. The computer readable medium of claim 7, wherein automatically initiating the encrypted channel between the client and the secure sewer comprises establishing an IP address hopping scheme between the client and the secure server.

12. The computer readable medium of claim 7, wherein automatically initiating the encrypted channel between the client and the secure server avoids sending a true IP address of the secure server to the client.

13. A computer readable medium storing a domain name server (DNS) module comprised of computer readable instructions that, when executed, cause a data processing device to perform the steps of:
(i) determining whether a DNS request sent by a client corresponds to a secure server;
(ii) when the DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a nonsecure computer; and
(iii) when the intercepted DNS request corresponds to a secure server, automatically creating a secure channel between the client and the secure server.

14. The computer readable medium of claim 13, wherein step (iii) comprises the steps of
(a) determining whether the client is authorized to access the secure server; and
(b) when the client is authorized to access the secure server, sending a request to the secure server to establish a secure channel between the secure server and the client.

15. The computer readable medium of claim 14, wherein step (iii) further comprises the step of:
(c) when the client is not authorized to access the secure server, returning a host unknown error message to the client.

16. The computer readable medium of claim 15, wherein the client comprises a web browser into which a user enters a URL resulting in the DNS request.

* * * * *

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

April 12, 2011

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF:

U.S. PATENT: *7,921,211*

ISSUE DATE: *April 05, 2011*

By Authority of the

Under Secretary of Commerce for Intellectual Property and Director of the United States Patent and Trademark Office

W. Montgomery

W. MONTGOMERY

Certifying Officer

Plaintiffs' VirnetX Exhibit

VirnetX, Inc. v. Apple, Inc.

PX010

C.A. 6:10-cv-0417

VX00036869

US007921211B2

(12) **United States Patent**
Larson et al.

(10) **Patent No.: US 7,921,211 B2**
(45) **Date of Patent: *Apr. 5, 2011**

(54) **AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES**

(75) Inventors: **Victor Larson**, Fairfax, VA (US); **Robert Dunham Short, III**, Leesburg, VA (US); **Edmund Colby Munger**, Crownsville, MD (US); **Michael Williamson**, South Riding, VA (US)

(73) Assignee: **VirnetX, Inc.**, Scotts Valley, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 701 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **11/840,560**

(22) Filed: **Aug. 17, 2007**

(65) **Prior Publication Data**

US 2008/0040792 A1 Feb. 14, 2008

Related U.S. Application Data

(63) Continuation of application No. 10/714,849, filed on Nov. 18, 2003, now Pat. No. 7,418,504, which is a continuation of application No. 09/558,210, filed on Apr. 26, 2000, now abandoned, which is a continuation-in-part of application No. 09/504,783, filed on Feb. 15, 2000, now Pat. No. 6,502,135, which is a continuation-in-part of application No. 09/429,643, filed on Oct. 29, 1999, now Pat. No. 7,010,604.

(60) Provisional application No. 60/106,261, filed on Oct. 30, 1998, provisional application No. 60/137,704, filed on Jun. 7, 1999.

(51) **Int. Cl.**
G06F 15/173 (2006.01)

(52) **U.S. Cl.** **709/226**

(58) **Field of Classification Search** 709/226, 709/221; 726/15

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2,895,502	A	7/1959	Roper et al.
5,303,302	A	4/1994	Burrows
5,311,593	A	5/1994	Carmi

(Continued)

FOREIGN PATENT DOCUMENTS

EP	0838930	4/1988

(Continued)

OTHER PUBLICATIONS

Baumgartner et al, "Differentiated Services: A New Approach for Quality of Service in the Internet," International Conference on High Performance Networking, 255-273 (1998).

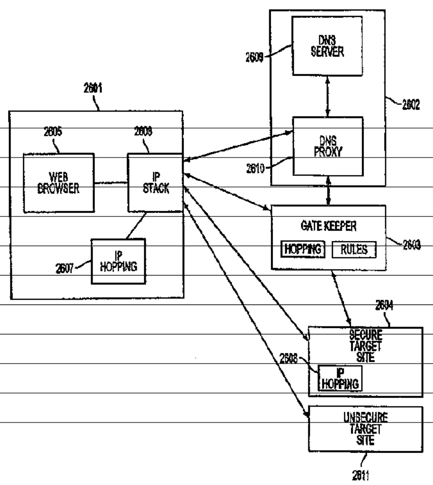
(Continued)

Primary Examiner — Krisna Lim
(74) *Attorney, Agent, or Firm* — McDermott Will & Emery LLP

(57) **ABSTRACT**

A secure domain name service for a computer network is disclosed that includes a portal connected to a computer network, such as the Internet, and a domain name database connected to the computer network through the portal. The portal authenticates a query for a secure computer network address, and the domain name database stores secure computer network addresses for the computer network. Each secure computer network address is based on a non-standard top-level domain name, such as .scom, .sorg, .snet, .snet, .sedu, .smil and .sint.

60 Claims, 40 Drawing Sheets



U.S. PATENT DOCUMENTS

5,384,848 A		1/1995	Kikuchi	
5,511,122 A		4/1996	Atkinson	
5,629,984 A		5/1997	McManis	
5,764,906 A		6/1998	Edelstein et al.	
5,771,239 A		6/1998	Moroney et al.	
5,805,803 A		9/1998	Birrell et al.	
5,822,434 A		10/1998	Caronni et al.	
5,864,666 A	*	1/1999	Shrader	726/15
5,870,610 A		2/1999	Beyda et al.	
5,898,830 A		4/1999	Wesinger, Jr. et al.	
5,950,195 A		9/1999	Stockwell et al.	
6,052,788 A		4/2000	Wesinger et al.	
6,055,574 A		4/2000	Smorodinsky et al.	
6,061,346 A		5/2000	Nordman	
6,079,020 A		6/2000	Liu	
6,081,900 A	*	6/2000	Subramaniam et al.	726/19
6,101,182 A		8/2000	Sistanizadeh et al.	
6,119,171 A		9/2000	Alkhatib	
6,173,399 B1		1/2001	Gilbrech	
6,199,112 B1		3/2001	Wilson	
6,202,081 B1		3/2001	Naudus	
6,223,287 B1		4/2001	Douglas et al.	
6,226,748 B1		5/2001	Bots et al.	
6,226,751 B1		5/2001	Arrow et al.	
6,246,670 B1		6/2001	Karlsson et al.	
6,262,987 B1		7/2001	Mogul	
6,298,341 B1		10/2001	Mann et al.	
6,314,463 B1		11/2001	Abbott et al.	
6,333,272 B1		12/2001	McMillin et al.	
6,338,082 B1		1/2002	Schneider	
6,502,135 B1		12/2002	Munger et al.	
6,557,037 B1		4/2003	Provino	
6,687,746 B1		2/2004	Shuster et al.	
6,701,437 B1		3/2004	Hoke et al.	
6,752,166 B2		6/2004	Lull et al.	
6,757,740 B1		6/2004	Parkh et al.	
6,937,597 B1		8/2005	Rosenberg et al.	
7,039,713 B1		5/2006	Van Gunter et al.	
7,072,964 B1		7/2006	Whittle et al.	
7,167,904 B1		1/2007	Devarajan et al.	
7,188,175 B1		3/2007	McKeeth	
7,353,841 B2		4/2008	Kono et al.	
7,461,334 B1		12/2008	Lu et al.	
7,490,151 B2		2/2009	Munger et al.	
7,493,403 B2		2/2009	Shull et al.	
2001/0049741 A1		12/2001	Skene et al.	
2004/0199493 A1		10/2004	Ruiz et al.	
2004/0199520 A1		10/2004	Ruiz et al.	
2004/0199608 A1		10/2004	Rechterman et al.	
2004/0199620 A1		10/2004	Ruiz et al.	
2007/0208869 A1		9/2007	Adelman et al.	
2007/0214284 A1		9/2007	King et al.	
2007/0266141 A1		11/2007	Norton	
2008/0235507 A1		9/2008	Ishikawa et al.	

FOREIGN PATENT DOCUMENTS

EP	0814589	12/1997
GB	2317792	4/1998
GB	2334181	8/1999
GB	2340702	2/2000
JP	62-214744	9/1987
JP	04-363941	12/1992
JP	09-018492	1/1997
JP	10-070531	3/1998
WO	WO98/27783	6/1998
WO	WO99/11019	3/1999
WO	WO 00/17775	3/2000
WO	WO 00/70458	11/2000
WO	WO 01/16766	3/2001

OTHER PUBLICATIONS

Chapman et al., "Domain Name System (DNS)," 278-296 (1995).

Davila et al., "Implementation of Virtual Private Networks at the Transport Layer," M. Mambo, Y. Zheng (Eds), Information Security (Second International) Workshop, ISW' 99. Lecture Notes in Computer Science (LNCS), vol. 1729; 85-102 (1999).

De Raadt et al., "Cryptography in OpenBSD," 10 pages (1999).

Eastlake, "Domain Name System Security Extensions," Internet Citation, Retrieved from the Internet: URL:ftp://ftp.inet.no/pub/ietf/internet-drafts/draft-ietf-dnssec-secext2-05.txt (1998).

Gunter et al., "An Architecture for Managing QoS-Enabled VRNs Over the Internet," Proceedings 24th Conference on Local Computer Networks. LCN' 99 IEEE Comput. Soc Los Alamitos, CA, pp. 122-131 (1999).

Shimizu, "Special Feature: Mastering the Internet with Windows 2000", Internet Magazine, 63:296-307 (2000).

Stallings, "Cryptography and Network Security," Principals and Practice, 2nd Edition, pp. 399-440 (1999).

Takata, "U.S. Vendors Take Serious Action to Act Against Crackers—A Tracking Tool and a Highly Safe DNS Software are Released", Nikkei Communications, 257:87(1997).

Wells, Email (Lancasterb1be@mail.msn.com), Subject: "Security Icon," (1998).

Fasbender, A., et al., Variable and Scalable Security: Protection of Location Information in Mobile IP, IEEE VTS, 46th, 1996, 5 pp.

DNS-related correspondence dated Sep. 7, 1993 to Sep. 20, 1993. (Pre KX, KX Records).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Dec. 2, 1996). (RFC 2543 Internet Draft 1).

Aventail Corp., "AutoSOCKS v. 2.1 Datasheet," available at http://www.archive.org/web/19970212013409/www.aventail.com/prod/autosk2ds.html (1997). (AutoSOCKS, Aventail).

Aventail Corp., "Socks Version 5," Aventail Whitepaper, available at http://web.archive.org/web/19970620030312/www.aventail.com/educate/whitepaper/soc kswp.html (1997). (Socks, Aventail).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Mar. 27, 1997). (RFC 2543 Internet Draft 2).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jul. 31, 1997). (RFC 2543 Internet Draft 3).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Nov. 11, 1997). (RFC 2543 Internet Draft 4).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (May 14, 1998). (RFC 2543 Internet Draft 5).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jun. 17, 1998). (RFC 2543 Internet Draft 6).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jul. 16, 1998). (RFC 2543 Internet Draft 7).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Aug. 7, 1998). (RFC 2543 Internet Draft 8).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Sep. 18, 1998). (RFC 2543 Internet Draft 9).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Nov. 12, 1998). (RFC 2543 Internet Draft 10).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Dec. 15, 1998). (RFC 2543 Internet Draft 11).

Aventail Corp., "Aventail Connect 3.1/2.6Administrator's Guide," (1999). (Aventail Administrator 3.1, Aventail).

Aventail Corp., "Aventail Connect 3.1/2.6 User's Guide," (1999). (Aventail User 3.1, Aventail).

Aventail Corp., "Aventail ExtraWeb Server v3.2 Administrator's Guide," (1999). (Aventail ExtraWeb 3.2, Aventail).

Check Point Software Technologies Ltd. (1999) (Check Point, Checkpoint FW).

M. Handley, H. Schulzrinne, E. Schooler, Internet Engineering Task Force, Internet Draft, (Jan. 15, 1999). (RFC 2543 Internet Draft 12).

Goncalves, et al. *Check Point FireWall—1 Administration Guide*, McGraw-Hill Companies (2000). (Goncalves, Checkpoint FW).

Assured Digital Products. (Assured Digital).

F-Secure, *F-Secure Evaluation Kit* (May 1999) (FSECURE 00000003) (Evaluation Kit 3).

F-Secure, *F-Secure Evaluation Kit* (Sep. 1998) (FSECURE 00000009) (Evaluation Kit 9).

IRE, Inc., *SafeNet/Soft-PK Version 4* (Mar. 28, 2000) (Soft-PK Version 4).

IRE/SafeNet Inc., *VPN Technologies Overview* (Mar. 28, 2000) (Safenet VPN Overview).

IRE, Inc., *SafeNet/VPN Policy Manager Quick Start Guide Version 1* (1999) (SafeNet VPN Policy Manager).

Information Assurance/NAI Labs, *Dynamic Virtual Private Networks Presentation v.3* (2000).

U.S. Appl. No. 60/134,547, filed May 17, 1999, Victor Sheymov.
U.S. Appl. No. 60/151,563, filed Aug. 31, 1999, Bryan Whittles.
U.S. Appl. No. 09/399,753, filed Sep. 22, 1998, Graig Miller et al.
Microsoft Corporation's Fourth Amended Invalidity Contentions dated Jan. 5, 2009, *VirnetX Inc. and Science Applications International Corp.* v. *Microsoft Corporation.*
Appendix A of the Microsoft Corporation's Fourth Amended Invalidity Contentions dated Jan. 5, 2009.
Concordance Table for the References Cited in Tables on pp. 6-15, 71-80 and 116-124 of the Microsoft Corporation's Fourth Amended Invalidity Contentions dated Jan. 5, 2009.
1. P. Mockapetris, "DNS Encoding of Network Names and Other Types," Network Working Group, RFC 1101 (Apr. 1989) (RFC1101, DNS SRV).
R. Atkinson, "An Internetwork Authentication Architecture," Naval Research Laboratory, Center for High Assurance Computing Systems (Aug. 5, 1993). (Atkinson NRL, KX Records).
Henning Schulzrinne, *Personal Mobility for Multimedia Services In The Internet*, Proceedings of the Interactive Distributed Multimedia Systems and Services European Workshop at 143 (1996). (Schulzrinne 96).
Microsoft Corp., *Microsoft Virtual Private Networking: Using Point-to-Point Tunneling Protocol for Low-Cost, Secure, Remote Access Across the Internet* (1996) (printed from 1998 PDC DVD-ROM). (Point to Point, Microsoft Prior Art VPN Technology).
"Safe Surfing: How to Build a Secure World Wide Web Connection," IBM Technical Support Organization, (Mar. 1996). (Safe Surfing, Website Art).
Goldschlag, et al., "Hiding Routing Information," Workshop on Information Hiding, Cambridge, UK (May 1996). (Goldschlag II, Onion Routing).
"IPSec Minutes From Montreal", IPSEC Working Group Meeting Notes, http://www.sandleman.ca/ipsec/1996/08/msg00018.html (Jun. 1996). (IPSec Minutes, FreeS/WAN).
J. M. Galvin, "Public Key Distribution with Secure DNS," Proceedings of the Sixth USENIX UNIX Security Symposium, San Jose, California, Jul. 1996. (Galvin, DNSSEC).
J. Gilmore, et al. "Re: Key Management, anyone? (DNS Keying)," IPSec Working Group Mailing List Archives (Aug. 1996). (Gilmore DNS, FreeS/WAN).
H. Orman, et al. "Re: 'Re: DNS? was Re: Key Management, anyone?" IETF IPSec Working Group Mailing List Archive (Aug. 1996-Sep. 1996). (Orman DNS, FreeS/WAN).
Arnt Gulbrandsen & Paul Vixie, *A DNS RR for specifying the location of services (DNS SRV)*, IETF RFC 2052 (Oct. 1996). (RFC 2052, DNS SRV).
Freier, et al. "The SSL Protocol Version 3.0," Transport Layer Security Working Group (Nov. 18, 1996). (SSL, Underlying Security Technology).
M.G. Reed, et al. "Proxies for Anonymous Routing," 12th Annual Computer Security Applications Conference, San Diego, CA, Dec. 9-13, 1996. (Reed, Onion Routing).
Kenneth F. Alden & Edward P. Wobber, *The AltaVista Tunnel: Using the Internet to Extend Corporate Networks*, Digital Technical Journal (1997) (Alden, AltaVista.
Automative Industry Action Group, "ANX Release 1 Document Publication," AIAG (1997). (AIAG, ANX).
Automative Industry Action Group, "ANX Release 1 Draft Document Publication," AIAG Publications (1997). (AIAG Release, ANX).
Aventail Corp. "Aventail VPN Data Sheet," available at http://www.archive.org/web/19970212013043/www.aventail.com/prod/vpndata.html (1997).(Data Sheet, Aventail).
Aventail Corp., "Directed VPN Vs. Tunnel," available at http://web.archive.org/web/19970620030312/www.aventail.com/educate/directvpn.html (1997). (Directed VPN, Aventail).
Aventail Corp., "Managing Corporate Access to the Internet," Aventail AutoSOCKS White Paper *available at* http://web.archive.org/web/19970620030312/www.aventail.com/educate/whitepaper/ipmwp.html (1997). (Corporate Access, Aventail).
Aventail Corp., "VPN Server V2.0 Administration Guide," (1997). (VPN, Aventail).
Goldschlag, et al. "*Privacy on the Internet*," Naval Research Laboratory, Center for High Assurance Computer Systems (1997). (Goldschlag I, Onion Routing).
Microsoft Corp., *Installing Configuring and Using PPTP with Microsoft Clients and Servers* (1997). (Using PPTP, Microsoft Prior Art VPN Technology).
Microsoft Corp., *IP Security for Microsoft Windows NT Server 5.0* (1997) (printed from 1998 PDC DVD-ROM). (IP Security, Microsoft Prior Art VPN Technology).
Microsoft Corp., *Microsoft Windows NT Active Directory: An Introduction to the Next Generation Directory Services* (1997) (printed from 1998 PDC DVD-ROM). (Directory, Microsoft Prior Art VPN Technology).
Microsoft Corp., *Routing and Remote Access Service for Windows NT Server NewOpportunities Today and Looking Ahead* (1997) (printed from 1998 PDC DVD-ROM).(Routing, Microsoft Prior Art VPN Technology).
Microsoft Corp., *Understanding Point-to-Point Tunneling Protocol PPTP* (1997) (printed from 1998 PDC DVD-ROM). (Understanding PPTP, Microsoft Prior Art VPN Technology).
J. Mark Smith et.al., *Protecting a Private Network: The AltaVista Firewall*, Digital Technical Journal (1997). (Smith, AltaVista).
Naganand Doraswamy *Implementation of Virtual Private Networks (VPNs) with IPSecurity*, <draft-ietf-ipsec-vpn-00.txt> (Mar. 12, 1997). (Doraswamy).
Aventail Corp., "Aventail, and Cybersafe to Provide Secure Authentication For Internet and Intranet Communication," Press Release, Apr. 3, 1997. (Secure Authentication, Aventail).
D. Wagner, et al. "Analysis of the SSL 3.0 Protocol," (Apr. 15, 1997). (Analysis, Underlying Security Technologies).
Automotive Industry Action Group, "ANXO Certification Authority Service and Directory Service Definition for ANX Release 1," AIAG Telecommunications Project Team and Bellcore (May 9, 1997). (AIAG Defintion, ANX).
Automotive Industry Action Group, "ANXO Certification Process and ANX Registration Process Definition for ANX Release 1," AIAG Telecommunications Project Team and Bellcore (May 9, 1997). (AIAG Certification, ANX).
Aventail Corp., "Aventail Announces the First VPN Solution to Assure Interoperability Across Emerging Security Protocols," Jun. 2, 1997. (First VPN, Aventail).
Syverson, et al. "Private Web Browsing," Naval Research Laboratory, Center for High 8 Assurance Computer Systems (Jun. 2, 1997). (Syverson, Onion Routing).
Bellcore, "Metrics, Criteria, and Measurement Technique Requirements for ANX Release 1," AIAG Telecommunications Project Team and Bellcore (Jun. 16, 1997). (AIAG Requirements, ANX).
R. Atkinson, "Key Exchange Delegation Record for the DNS," Network Working Group, RFC 2230 (Nov. 1997). (RFC 2230, KX Records).
1998 Microsoft Professional Developers Conference DVD ("1998 PDC DVD-ROM") (including screenshots captured therefrom and produced as MSFTVX 00018827-00018832). (Conference, Microsoft Prior Art VPN Technology).
Microsoft Corp., *Virtual Private Networking An Overview* (1998) (printed from 1998 PDC DVD-ROM) (Overview, Microsoft Prior Art VPN Technology).
Microsoft Corp., *Windows NT 5.0 Beta Has Public Premiere at Seattle Mini-Camp Seminar attendees get first look at the performance and capabilities of Windows NT 5.0* (1998) (available at hap //www.microsoft.com/presspass/features/1998/10-19nt5.mspxpftrue).(NT Beta, Microsoft Prior Art VPN Technology).
"What ports does SSL use" available at stason.org/TULARC/security/ssl-talk/3-4-What-ports-does-ssl-use.html (1998). (Ports, DNS SRV).
Aventail Corp., "Aventail VPN V2.6 Includes Support for More Than Ten Authentication Methods Making Extranet VPN Development Secure and Simple," Press Release, Jan. 19, 1998. (VPN V2.6, Aventail).
R. G. Moskowitz, "Network Address Translation Issues with IPsec," Internet Draft, Internet Engineering Task Force, Feb. 6, 1998. (Moskowitz).

H. Schulzrinne, et al, "Internet Telephony Gateway Location," Proceedings of IEEE INfocom '98, The Conference on Computer Communications, vol. 2 (Mar. 29-Apr. 2, 1998). (Gateway, Schulzrinne).
C. Huitema, 45 al. "Simple Gateway Control Protocol," Version 1.0 (May 5, 1998). (SGCP).
DISA "Secret Internet Protocol Router Network," SIPRNET Program Management Office (D3113) DISN Networks, DISN Transmission Services (May 8, 1998). (DISA, SIPRNET).
D. McDonald, et al. "PF_KEY Key Management API, Version 2," Network Working Group, RFC 2367 (Jul. 1998). (RFC 2367).
Microsoft Corp., *Company Focuses on Quality and Customer Feedback* (Aug. 18, 1998). (Focus, Microsoft Prior Art VPN Technology).
Atkinson, et al. "Security Architecture for the Internet Protocol," Network Working Group, RFC 2401 (Nov. 1998). (RFC 2401, Underlying Security Technologies).
Donald Eastlake, *Domain Name System Security Extensions*, IETF DNS Security Working Group (Dec. 1998). (DNSSEC-7).
Kaufman et al, "Implementing IPsec," (Copyright 1999). (Implementing IPSEC, VPN References).
Network Solutions, Inc. "Enabling SSL," NSI Registry (1999). (Enabling SSL, Underlying Security Technologies).
C. Scott, et al. *Virtual Private Networks*, O'Reilly and Associates, Inc.; 2nd ed. (Jan. 1999). (Scott VPNs).
Goldschlag, et al., "Onion Routing for Anonymous and Private Internet Connections," Naval Research Laboratory, Center for High Assurance Computer Systems (Jan. 28, 1999). (Goldschlag III, Onion Routing).
H. Schulzrinne, "Internet Telephony: architecture and protocols—an IETF perspective," Computer Networks, vol. 31, No. 3 (Feb. 1999). (Telephony, Schulzrinne).
M. Handley, et al. "SIP: Session Initiation Protocol," Network Working Group, RFC 2543 and Internet Drafts (Dec. 1996-Mar. 1999). (Handley, RFC 2543).
FreeS/WAN Project, *Linux FreeS/WAN Compatibility Guide* (Mar. 4, 1999). (FreeS/WAN Compatibility Guide, FreeS/WAN).
Telcordia Technologies, "ANX Release 1 Document Corrections," AIAG (May 11, 1999). (Telcordia, ANX).
Ken Hornstein & Jeffrey Altman, *Distributing Kerberos KDC and Realm Information with DNS* <draft-eitf-cat-krb-dns-locate-oo.txt> (Jun. 21, 1999). (Hornstein, DNS SRV).
Bhattacharya et. al. "An LDAP Schema for Configuration and Administration of IPSec Based Virtual Private Networks (VPNs)", IETF Internet Draft (Oct. 1999). (Bhattcharya LDAP VPN).
B. Patel, et al. "DHCP Configuration of IPSEC Tunnel Mode," IPSEC Working Group, Internet Draft 02 (Oct. 15, 1999). (Patel).
"Building a Microsoft VPN: A Comprehensive Collection of Microsoft Resources," FirstVPN, (Jan. 2000). (FirstVPN Microsoft).
Gulbrandsen, Vixie, & Esibov, *A DNS RR for specifying the location of services (DNS SRV)*, IETF RFC 2782 (Feb. 2000). (RFC 2782, DNS SRV).
Mitre Organization, "Technical Description," Collaborative Operations in Joint Expeditionary Force Experiment (JEFX) 99 (Feb. 2000). (MITRE, SIPRNET).
H. Schulzrinne, et al. "Application-Layer Mobility Using SIP," Mobile Computing and Communications Review, vol. 4, No. 3. pp. 47-57 (Jul. 2000). (Application, SIP).
Kindred et al, "Dynamic VPN Communities: Implementation and Experience," DARPA Information Survivability Conference and Exposition II (Jun. 2001). (DARPA, VPN Systems).
ANX 101: Basic ANX Service Outline. (Outline, ANX).
ANX 201: Advanced ANX Service. (Advanced, ANX).
Appendix A: Certificate Profile for ANX IPsec Certificates. (Appendix, ANX).
Aventail Corp., "Aventail AutoSOCKS the Client Key to Network Security," Aventail Corporation White Paper. (Network Security, Aventail).
Cindy Moran, "DISN Data Networks: Secret Internet Protocol Router Network (SIPRNet)." (Moran, SIPRNET).
Data Fellows F-Secure VPN+ (F-Secure VPN+).
Interim Operational Systems Doctrine for the Remote Access Security Program (RASP) Secret Dial-In Solution. (RASP, SIPRNET).
Onion Routing, "Investigation of Route Selection Algorithms," available at http://www.onion-router.net/Archives/Route/index.html. (Route Selection, Onion Routing).
Secure Computing, "Bullet-Proofing an Army Net," Washington Technology. (Secure, SIPRNET).
Sparta "Dynamic Virtual Private Network." (Sparta, VPN Systems).
Standard Operation Procedure for Using the 1910 Secure Modems. (Standard, SIPRNET).
Publically available emails relating to FreeS/WAN (MSFTVX00018833-MSFTVX00019206). (FreeS/WAN emails, FreeS/WAN).
Kaufman et al., "Implementing IPsec," (Copyright 1999) (Implementing IPsec).
Network Associates *Gauntlet Firewall For Unix User's Guide Version 5.0* (1999). (Gauntlet User's Guide—Unix, Firewall Products).
Network Associates *Gauntlet Firewall for Windows NT Getting Started Guide Version 5.0* (1999) (Gauntlet Getting Started Guide—NT, Firewall Products).
Network Associates *Gauntlet Firewall for Unix Getting Started Guide Version 5.0* (1999) (Gauntlet Unix Getting Started Guide, Firewall Products).
Network Associates *Release Notes Gauntlet Firewall for Unix 5.0* (Mar. 19, 1999) (Gauntlet Unix Release Notes, Firewall Products).
Network Associates *Gauntlet Firewall For Windows NT Administrator's Guide Version 5.0* (1999) (Gauntlet NT Administrator's Guide, Firewall Products).
Trusted Information Systems, Inc. *Gauntlet Internet Firewall Firewall-to-Firewall Encryption Guide Version 3.1* (1996) (Gauntlet Firewall-to-Firewall, Firewall Products).
Network Associates *Gauntlet Firewall Global Virtual Private Network User's Guide for Windows NT Version 5.0* (1999) (Gauntlet NT GVPN, GVPN).
Network Associates *Gauntlet Firewall For UNIX Global Virtual Private Network User's Guide Version 5.0* (1999) (Gauntlet Unix GVPN, GVPN).
Dan Sterne *Dynamic Virtual Private Networks* (May 23, 2000) (Sterne DVPN, DVPN).
Darrell Kindred *Dynamic Virtual Private Networks (DVPN)* (Dec. 21, 1999) (Kindred DVPN, DVPN).
Dan Sterne et.al. *TIS Dynamic Security Perimeter Research Project Demonstration* (Mar. 9, 1998) (Dynamic Security Perimeter, DVPN).
Darrell Kindred *Dynamic Virtual Private Networks Capability Description* (Jan. 5, 2000) (Kindred DVPN Capability, DVPN) 11.
Oct. 7, and 28, 1997 email from Domenic J. Turchi Jr. (SPARTA00001712-1714, 1808-1811) (Turchi DVPN email, DVPN).
James Just & Dan Sterne *Security Quickstart Task Update* (Feb. 5, 1997) (Security Quickstart, DVPN).
Virtual Private Network Demonstration dated Mar. 21, 1998 (SPARTA00001844-54) (DVPN Demonstration, DVPN).
GTE Internetworking & BBN Technologies *DARPA Information Assurance Program Integrated Feasibility Demonstration (IFD) 1.1 Plan* (Mar. 10, 1998) (IFD 1.1, DVPN).
Microsoft Corp. Windows NT Server Product Documentation: Administration Guide—Connection Point Services, available at http://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/cpsops.mspx (Connection Point Services) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows. Accordingly, upon information and belief, this reference is prior art to the patents-insuit.).
Microsoft Corp. Windows NT Server Product Documentation: Administration Kit Guide—Connection Manager, available at http://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/cmak.mspx (Connection Manager) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).
Microsoft Corp. Autodial Heuristics, available at http://support.microsoft.com/kb/164249 (Autodial Heuristics) (Although undated, this reference refers to the operation of prior art versions of Microsoft

Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).
Microsoft Corp., Cariplo: Distributed Component Object Model, (1996) available at http://msdn2.microsoft.com/en-us/library/ms809332(printer).aspx (Cariplo I).
Marc Levy, COM Internet Services (Apr. 23, 1999), available at http://msdn2.microsoft.com/en-us/library/ms809302(printer).aspx (Levy).
Markus Horstmann and Mary Kirtland, DCOM Architecture (Jul. 23, 1997), available at http://msdn2.microsoft.com/en-us/library/ms809311(printer).aspx (Horstmann).
Microsoft Corp., DCOM: A Business Overview (Apr. 1997), available at http://msdn2.microsoft.com/en-us/library/ms809320(printer).aspx (DCOM Business Overview I).
Microsoft Corp., DCOM Technical Overview (Nov. 1996), available at http://msdn2.microsoft.com/en-us/library/ms809340(printer).aspx (DCOM Technical Overview I).
Microsoft Corp., DCOM Architecture White Paper (1998) available in PDC DVD-ROM (DCOM Architecture).
Microsoft Corp, DCOM—The Distributed Component Object Model, A Business Overview White Paper (Microsoft 1997) available in PDC DVD-ROM (DCOM Business Overview II).
Microsoft Corp., DCOM—Cariplo Home Banking Over The Internet White Paper (Microsoft 1996) available in PDC DVD-ROM (Cariplo II).
Microsoft Corp., DCOM Solutions in Action White Paper (Microsoft 1996) available in PDC DVD-ROM (DCOM Solutions in Action).
Microsoft Corp., DCOM Technical Overview White Paper (Microsoft 1996) available 12 in PDC DVD-ROM (DCOM Technical Overview II).
Scott Suhy & Glenn Wood, DNS and Microsoft Windows NT 4.0, (1996) available at http://msdn2.microsoft.com/en-us/library/ms810277(printer).aspx (Suhy).
Aaron Skonnard, *Essential WinInet* 313-423 (Addison Wesley Longman 1998) (Essential WinInet).
Microsoft Corp. Installing, Configuring, and Using PPTP with Microsoft Clients and Servers, (1998) available at http://msdn2.microsoft.com/enus/library/ms811078(printer).aspx (Using PPTP).
Microsoft Corp., Internet Connection Services for MS RAS, Standard Edition, http://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/bcgstart.mspx (Internet Connection Services I).
Microsoft Corp., Internet Connection Services for RAS, Commercial Edition, available athttp://www.microsoft.com/technet/archive/winntas/proddocs/inetconctservice/bcgstrtc.mspx (Internet Connection Services II).
Microsoft Corp., Internet Explorer 5 Corporate Deployment Guide—Appendix B:Enabling Connections with the Connection Manager Administration Kit, available at http://www.microsoft.com/technet/prodtechnol/ ie/deploy/deploy5/appendb.mspx (IE5 Corporate Development).
Mark Minasi, *Mastering Windows NT Server 4* 1359-1442 (6th ed., Jan. 15, 1999)(Mastering Windows NT Server).
Hands On, Self-Paced Training for Supporting Version 4.0 371-473 (Microsoft Press 1998) (Hands On).
Microsoft Corp., MS Point-to-Point Tunneling Protocol (Windows NT 4.0), available at http://www.microsoft.com/technet/archive/winntas/maintain/featusability/pptpwp3.mspx (MS PPTP).
Kenneth Gregg, et al., *Microsoft Windows NT Server Administrator's Bible* 173-206, 883-911, 974-1076 (IDG Books Worldwide 1999) (Gregg).
Microsoft Corp., Remote Access (Windows), available at http://msdn2.microsoft.com/en-us/library/bb545687(VS.85,printer).aspx (Remote Access).
Microsoft Corp., Understanding PPTP (Windows NT 4.0), available at http://www.microsoft.com/technet/archive/winntas/plan/pptpudst.mspx (Understanding PPTP NT 4) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).
Microsoft Corp., Windows NT 4.0: Virtual Private Networking, available at http://www.microsoft.com/technet/archive/winntas/ deploy/confeat/vpntwk.mspx (NT4 VPN) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).
Anthony Northrup, *NT Network Plumbing: Routers, Proxies, and Web Services* 299-399 (IDG Books Worldwide 1998) (Network Plumbing).
Microsoft Corp., Chapter 1—Introduction to Windows NT Routing with Routing and Remote Access Service, Available at http://www.microsoft.com/technet/archive/winntas/proddocs/ rras40/rrasch01.mspx (Intro to RRAS) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.) 13.
Microsoft Corp., Windows NT Server Product Documentation: Chapter 5—Planning for Large-Scale Configurations, available at http://www.microsoft.com/technet/archive/winntas/proddocs/rras40/rrasch05.mspx (Large-Scale Configurations) (Although undated, this reference refers to the operation of prior art versions of Microsoft Windows such as Windows NT 4.0. Accordingly, upon information and belief, this reference is prior art to the patents-in-suit.).
F-Secure, *F-Secure NameSurfer* (May 1999) (from FSECURE 00000003) (NameSurfer 3).
F-Secure, *F-Secure VPN Administrator's Guide* (May 1999) (from FSECURE 00000003) (F-Secure VPN 3).
F-Secure, *F-Secure SSH User's & Administrator's Guide* (May 1999) (from FSECURE 00000003) (SSH Guide 3).
F-Secure, *F-Secure SSH2.0 for Windows NT and 95* (May 1999) (from FSECURE 00000003) (SSH 2.0 Guide 3).
F-Secure, *F-Secure VPN+ Administrator's Guide* (May 1999) (from Fsecure 00000003) (VPN+ Guide 3).
F-Secure, *F-Secure VPN+ 4.1* (1999) (from Fsecure 00000006) (VPN+ 4.1 Guide 6).
F-Secure, *F-Secure SSH* (1996) (from Fsecure 00000006) (F-Secure SSH 6).
F-Secure, *F-Secure SSH 2.0 for Windows NT and 95* (1998) (from Fsecure 00000006) (F-Secure SSH 2.0 Guide 6).
F-Secure, *F-Secure SSH User's & Administrator's Guide* (Sep. 1998) (from Fsecure 00000009) (SSH Guide 9).
F-Secure, *F-Secure SSH 2.0 for Windows NT and 95* (Sep. 1998) (from Fsecure 00000009) (F-Secure SSH 2.0 Guide 9).
F-Secure, *F-Secure VPN+* (Sep. 1998) (from Fsecure 00000009) (VPN+ Guide 9).
F-Secure, *F-Secure Management Tools, Administrator's Guide* (1999) (from Fsecure 00000003) (F-Secure Management Tools).
F-Secure, *F-Secure Desktop, User's Guide* (1997) (from Fsecure 00000009) (F-Secure Desktop User's Guide).
SafeNet, Inc., *VPN Policy Manager* (Jan. 2000) (VPN Policy Manager).
F-Secure, F-Secure VPN+ for Windows NT 4.0 (1998) (from Fsecure 00000009) (FSecure VPN+).
IRE, Inc., *SafeNet / Security Center Technical Reference Addendum* (Jun. 22, 1999) (Safenet Addendum).
IRE, Inc., *System Description for VPN Policy Manager and SafeNet/SoftPK* (Mar. 30, 2000) (VPN Policy Manager System Description).
IRE, Inc., About SafeNet / VPN Policy Manager (1999) (About Safenet VPN Policy Manager).
Trusted Information Systems, Inc., *Gauntlet Internet Firewall, Firewall Product Functional Summary* (Jul. 22, 1996) (Gauntlet Functional Summary).
Trusted Information Systems, Inc., *Running the Gauntlet Internet Firewall, An Administrator's Guide to Gauntlet Version 3.0* (May 31, 1995) (Running the Gauntlet Internet Firewall).
Ted Harwood, *Windows NT Terminal Server and Citrix Metaframe* (New Riders 1999) (Windows NT Harwood) 79.
Todd W. Matehrs and Shawn P. Genoway, *Windows NT Thing Client Solutions: Implemetning Terminal Server and Citrix MetaFrame* (Macmillan Technial Publishing 1999) (Windows NT Mathers).
Bernard Aboba et al., *Securing L2TP using IPSEC* (Feb. 2, 1999).
Finding Your Way Through the VPN Maze (1999) ("PGP").
Linux FreeS/WAN Overview (1999) (Linux FreeS/WAN) Overview).
TimeStep, *The Business Case for Secure VPNs* (1998) ("TimeStep").

WatchGuard Technologies, Inc., *WatchGuard Firebox System Powerpoint* (2000).
WatchGuard Technologies, Inc., *MSS Firewall Specifications* (1999).
WatchGuard Technologies, Inc., *Request for Information, Security Services* (2000).
WatchGuard Technologies, Inc., *Protecting the Internet Distributed Enterprise, White Paper* (Feb. 2000).
WatchGuard Technologies, Inc., *WatchGuard LiveSecurity for MSS Powerpoint* (Feb. 14, 2000).
WatchGuard Technologies, Inc., *MSS Version 2.5, Add-On for WatchGuard SOHO Releaset Notes* (Jul. 21, 2000).
Air Force Research Laboratory, *Statement of Work for Information Assurance System Architecture and Integration*, PR No. N-8-6106 (Contract No. F30602-98-C-0012) (Jan. 29, 1998).
GTE Internetworking & BBN Technologies *DARPA Information Assurance Program Integrated Feasibility Demonstration (IFD) 1.2 Report, Rev. 1.0* (Sep. 21, 1998).
BBN Information Assurance Contract, *TIS Labs Monthly Status Report* (Mar. 16-Apr. 30, 1998).
DARPA, *Dynamic Virtual Private Network (VPN) Powerpoint*.
GTE Internetworking, *Contractor's Program Progress Report* (Mar. 16-Apr. 30, 1998).
Darrell Kindred, *Dynamic Virtual Private Networks (DVPN) Countermeasure Characterization* (Jan. 30, 2001).
Virtual Private Networking Countermeasure Characterization (Mar. 30, 2000).
Virtual Private Network Demonstration (Mar. 21, 1998).
Information Assurance/NAI Labs, *Dynamic Virtual Private Networks (VPNs) and Integrated Security Management* (2000).
Information Assurance/NAI Labs, *Create/Add DVPN Enclave* (2000).
NAI Labs, *IFE 3.1 Integration Demo* (2000).
Information Assurance, *Science Fair Agenda* (2000).
Darrell Kindred et al., *Proposed Threads for IFE 3.1* (Jan. 13, 2000).
IFE 3.1 Technology Dependencies (2000).
IFE 3.1 Topology (Feb. 9, 2000).
Information Assurance, *Information Assurance Integration: IFE 3.1, Hypothesis & Thread Development* (Jan. 10-11, 2000).
Information Assurance/NAI Labs, *Dynamic Virtual Private Networks Presentation* (2000).
Information Assurance/NAI Labs, *Dynamic Virtual Private Networks Presentation v.2* (2000).
Information Assurance/NAI Labs, *Dynamic Virtual Private Networks Presentation v.3* (2000).
T. Braun et al., *Virtual Private Network Architecture*, Charging and Accounting Technology for the Internet (Aug. 1, 1999) (VPNA).
Network Associates Products—*PGP Total Network Security Suite, Dynamic Virtual Private Networks* (1999).
Microsoft Corporation, Microsoft Proxy Server 2.0 (1997) (Proxy Server 2.0, Microsoft Prior Art VPN Technology).
David Johnson et. al., *A Guide To Microsoft Proxy Server 2.0* (1999) (Johnson, Microsoft Prior Art VPN Technology).
Microsoft Corporation, *Setting Server Parameters* (1997 (copied from Proxy Server 2.0 CD labeled MSFTVX00157288) (Setting Server Parameters, Microsoft Prior Art VPN Technology).
Kevin Schuler, *Microsoft Proxy Server 2* (1998) (Schuler, Microsoft Prior Art VPN Technology).
Erik Rozell et. al., *MCSE Proxy Server 2 Study Guide* (1998) (Rozell, Microsoft Prior 15 Art VPN Technology).
M. Shane Stigler & Mark A Linsenbardt, *IIS 4 and Proxy Server 2* (1999) (Stigler, Microsoft Prior Art VPN Technology).
David G. Schaer, *MCSE Test Success: Proxy Server 2* (1998) (Schaer, Microsoft Prior Art VPN Technology).
John Savill, *The Windows NT and Windows 2000 Answer Book* (1999) (Savill, Microsoft Prior Art VPN Technology).
Network Associates *Gauntlet Firewall Global Virtual Private Network User's Guide for Windows NT Version 5.0* (1999) (Gauntlet NT GVPN, GVPN).
File History for U.S. Appl. No. 09/653,201, Applicant(s): Whittle Bryan, et al., filed Aug. 31, 2000.
AutoSOCKS v2.1, Datasheet, http://web.archive.org/web/19970212013409/www.aventail.com/prod/autoskds.html.
Ran Atkinson, *Use of DNS to Distribute Keys*, Sep. 7, 1993, http://ops.ietf.org/lists/namedroppers/namedroppers.199x/msg00945.html.
FirstVPN Enterprise Networks, Overview.
Chapter 1: Introduction to Firewall Technology, Administration Guide; Dec. 19, 2007, http://www.books24x7.com/book/id_762/viewer_r.asp?bookid=762&chunked=41065062.
The TLS Protocol Version 1.0; Jan. 1999; p. 65 of 71.
Elizabeth D. Zwicky, et al., Building Internet Firewalls, 2nd Ed.
Virtual Private Networks—Assured Digital Incorporated—ADI 4500; http://web.archive.org/web/19990224050035/www.assured-digital.com/products/prodvpn/adia4500.htm.
Accessware—The Third Wave in Network Security, Conclave from Internet Dynamics; http://web.archive.org/web/11980210013830/interdyn.com/Accessware.html.
Extended System Press Release, Sep. 2, 1997; *Extended VPN Uses The Internet to Create Virtual Private Networks*, www.extendedsystems.com.
Socks Version 5; Executive Summary; http://web.archive.org/web/199970620031945/www.aventail.com/educate/whitepaper/sockswp.html.
Internet Dynamics First to Ship Integrated Security Solutions for Enterprise Intranets and Extranets; Sep. 15, 1997; http://web.archive.org/web/19980210014150/interdyn.com.
Emails from various individuals to Linux IPsec re: DNS-LDAP Splicing.
Microsoft Corporation's Fifth Amended Invalidity Contentions dated Sep. 18, 2009, *VirnetX Inc. and Science Applications International Corp. v. Microsoft Corporation* and invalidity claim charts for U.S. Patent Nos. 7,188,180 and 6,839,759.
The IPSEC Protocol as described in Atkinson, et al., "Security Architecture for the Internet Protocol," Network Working Group, RFC 2401 (Nov. 1998) ("RFC 2401"); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.
S. Kent and R. Atkinson, "IP Authentication Header," RFC 2402 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.
C. Madson and R. Glenn, "The Use of HMAC-MD5-96 within ESP and AH," RFC 2403 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.
C. Madson and R. Glenn, "The Use HMAC-SHA-1-96 within ESP and AH," RFC 2404 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.
C. Madson and N. Doraswamy, "The ESP DES-CBC Cipher Algorithm With Explicit IV", RFC 2405 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.
S. Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)," RFC 2406 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.
Derrell Piper, "The Internet IP Security Domain of Interpretation for ISAKMP," RFC 2407 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.
Douglas Maughan, et al, "Internet Security Association and Key Management Protocol (ISAKMP)," RFC 2408 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.
D. Harkins and D. Carrell, "The Internet Key Exchange (IKE)," RFC 2409 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.
R. Glenn and S. Kent, "The NULL Encryption Algorithm and Its Use With IPsec," RFC 2410 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.

R. Thayer, et al., "IP Security Document Roadmap," RFC 2411 (Nov. 1998); http://web.archive.org/web/19991007070353/http://www.imib.med.tu-dresden.de/imib/Internet/Literatur/ipsec-docu_eng.html.
Hilarie K. Orman, "The OAKLEY Key Determination Protocol," RFC 2412 (Nov. 1998) in combination with J.M. Galvin, "Public Key Distribution with Secure DNS," Proceedings of the Sixth USENIX UNIX Security Symposium, San Jose California (Jul. 1996) ("Galvin").
WatchGuard Technologies, Inc., *WatchGuard Firebox System Powerpoint* (2000).
WatchGuard Technologies, Inc., *Request for Information, Security Services* (2000).
WatchGuard Technologies, Inc., *Protecting the Internet Distributed Enterprise, White Paper* (Feb. 2000).
WatchGuard Technologies, Inc., *MSS Version 2.5, Add-On for WatchGuard SOHO Release Notes* (Jul. 21, 2000).
Yuan Dong Feng, "A novel scheme combining interleaving technique with cipher in Rayleigh fading channels," Proceedings of the International Conference on Communication technology, 2:S47-02-1-S47-02-4 (1998).
D.W. Davies and W.L. Price, edited by Tadahiro Uezono, "Network Security", Japan, Nikkei McGraw-Hill, Dec. 5, 1958, First Edition, first copy, p. 102-108.
David Kosiur, "Building and Managing Virtual Private Networks" (1998).
P. Mockapetris, "Domain Names—Implementation and Specification," Network Working Group, RFC 1035 (Nov. 1987).
Request for *Inter Partes* Reexamination of Patent No. 6,502,135, dated Nov. 25, 2009.
Request for *Inter Partes* Reexamination of Patent No. 7,188,180, dated Nov. 25, 2009.

* cited by examiner

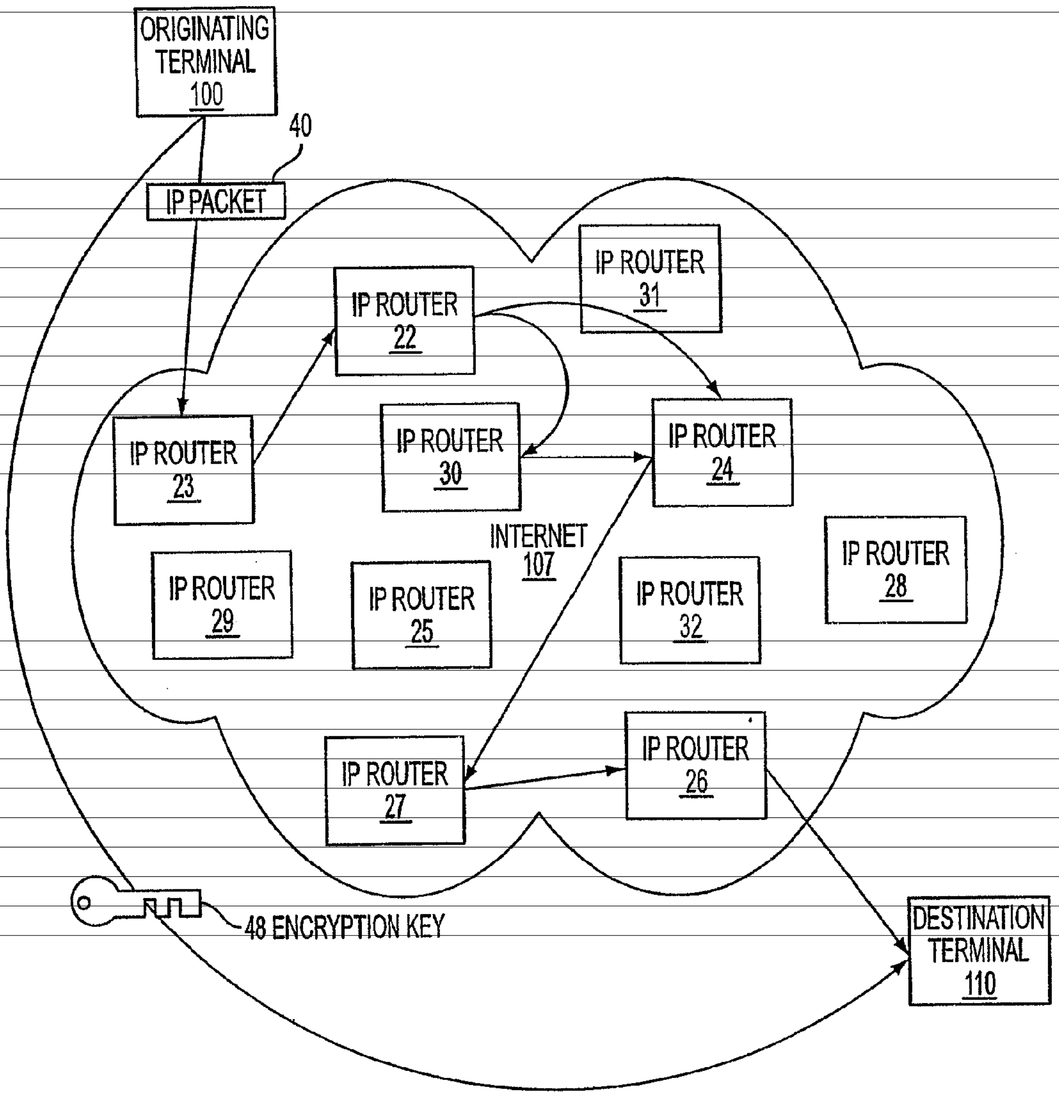

ORIGINATING TERMINAL 100
40
IP PACKET
IP ROUTER 22
IP ROUTER 31
IP ROUTER 23
IP ROUTER 30
IP ROUTER 24
INTERNET 107
IP ROUTER 29
IP ROUTER 25
IP ROUTER 32
IP ROUTER 28
IP ROUTER 27
IP ROUTER 26
48 ENCRYPTION KEY
DESTINATION TERMINAL 110

FIG. 1

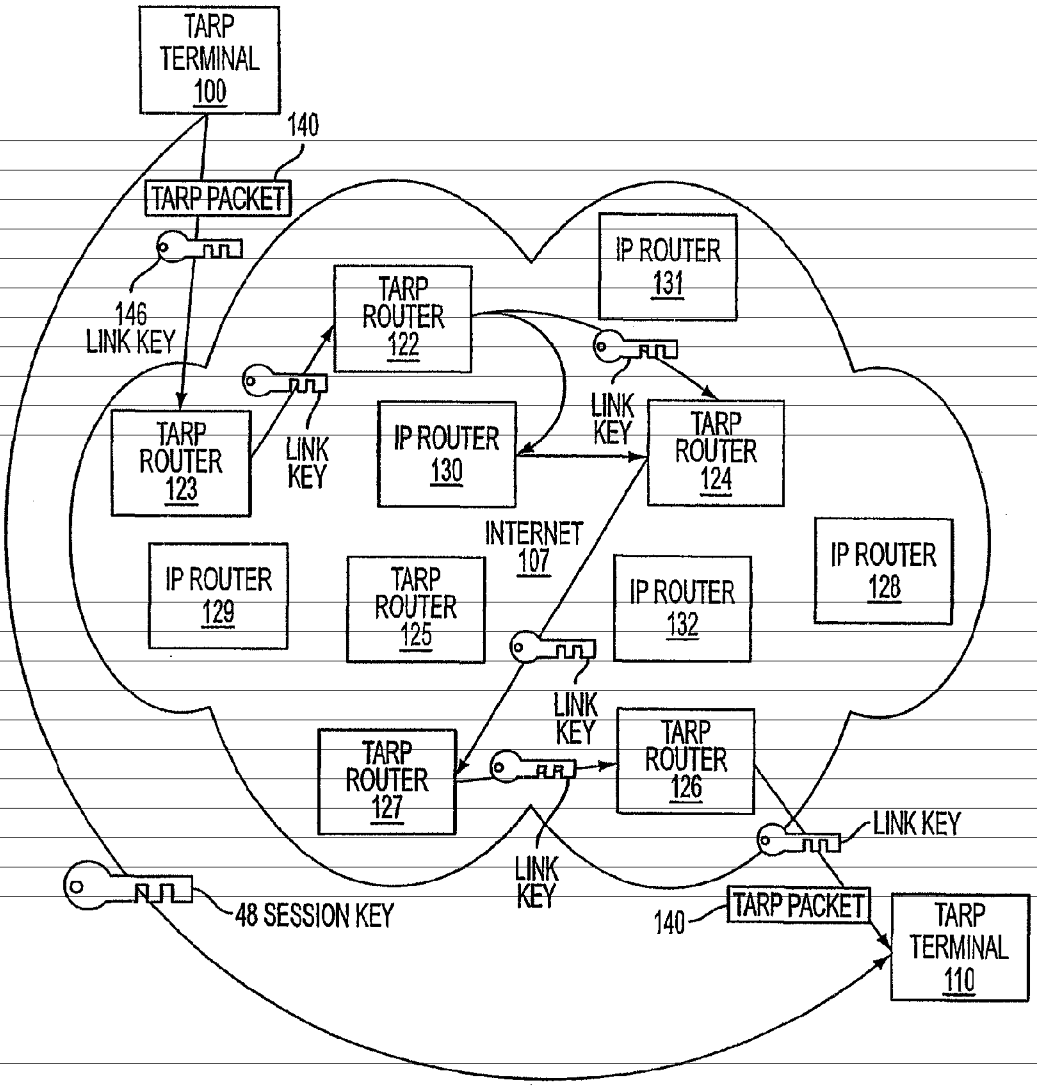

TARP TERMINAL 100
140
TARP PACKET
146 LINK KEY
TARP ROUTER 122
IP ROUTER 131
LINK KEY
TARP ROUTER 123
IP ROUTER 130
LINK KEY
TARP ROUTER 124
INTERNET 107
IP ROUTER 129
TARP ROUTER 125
IP ROUTER 132
IP ROUTER 128
LINK KEY
TARP ROUTER 127
TARP ROUTER 126
LINK KEY
LINK KEY
48 SESSION KEY
140
TARP PACKET
TARP TERMINAL 110

FIG. 2

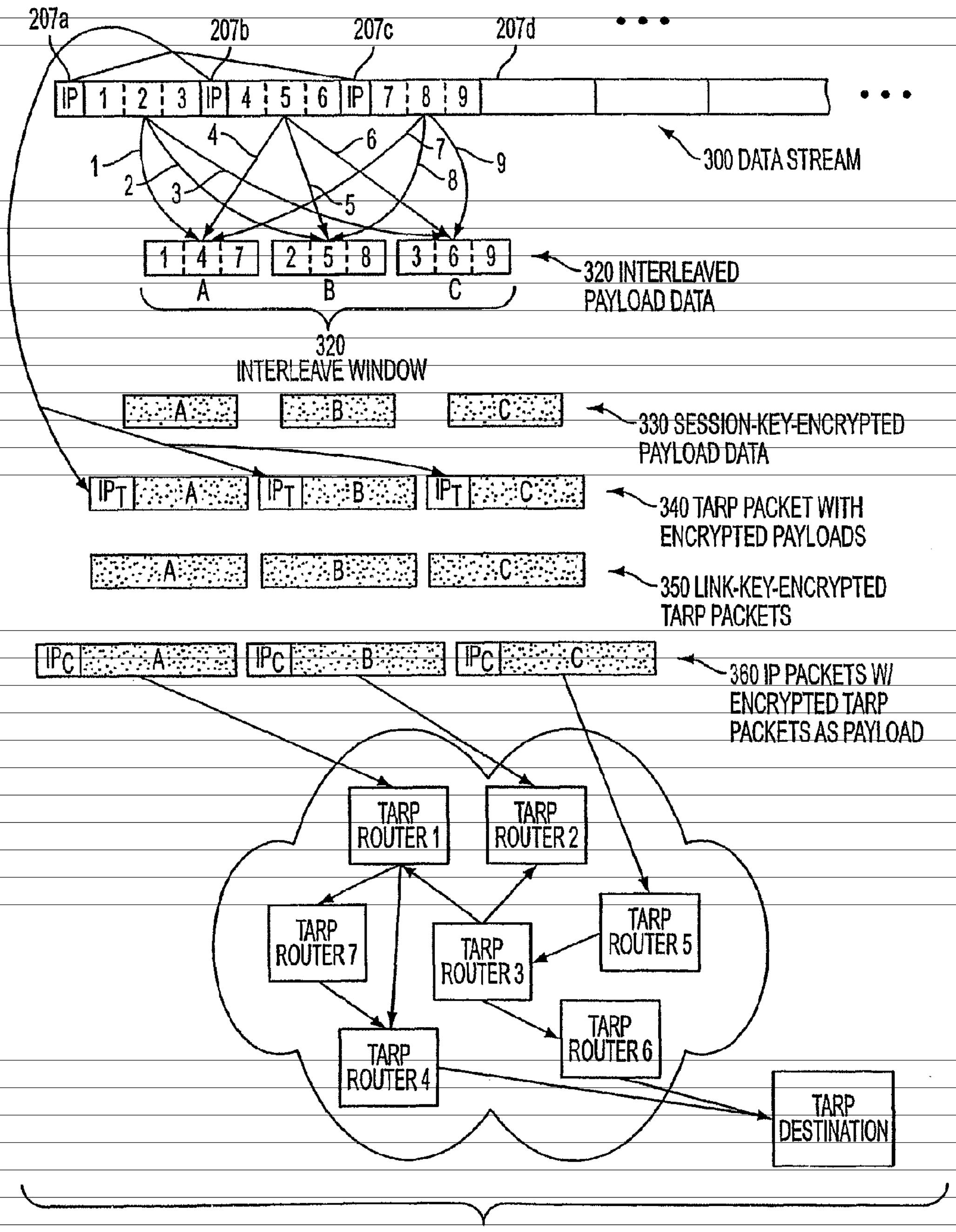

207a
207b
207c
207d
IP 1 2 3 IP 4 5 6 IP 7 8 9
1
2
3
4
5
6
7
8
9
300 DATA STREAM
1 4 7
2 5 8
3 6 9
A
B
C
320 INTERLEAVED PAYLOAD DATA
320 INTERLEAVE WINDOW
330 SESSION-KEY-ENCRYPTED PAYLOAD DATA
IPT A
IPT B
IPT C
340 TARP PACKET WITH ENCRYPTED PAYLOADS
350 LINK-KEY-ENCRYPTED TARP PACKETS
IPC A
IPC B
IPC C
360 IP PACKETS W/ ENCRYPTED TARP PACKETS AS PAYLOAD
TARP ROUTER 1
TARP ROUTER 2
TARP ROUTER 7
TARP ROUTER 3
TARP ROUTER 5
TARP ROUTER 6
TARP ROUTER 4
TARP DESTINATION

FIG. 3A

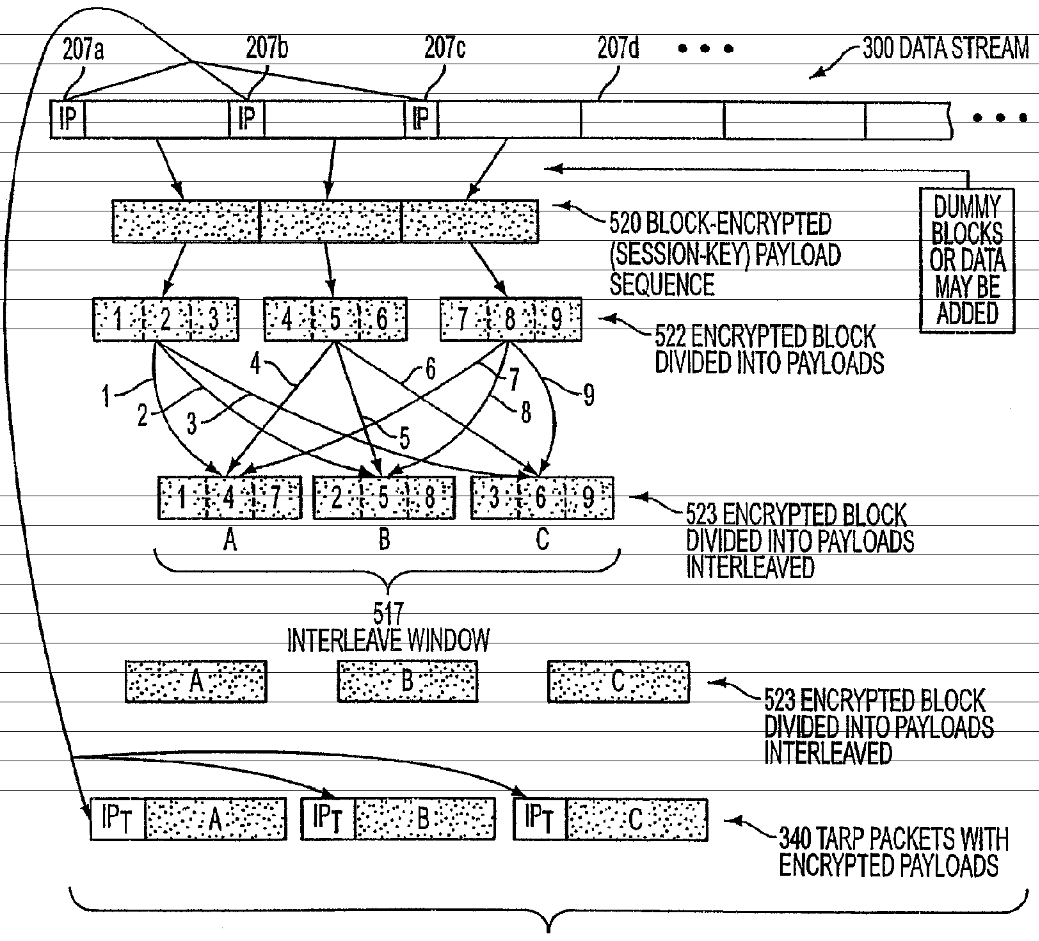

207a
207b
207c
207d
300 DATA STREAM
IP
IP
IP
DUMMY BLOCKS OR DATA MAY BE ADDED
520 BLOCK-ENCRYPTED (SESSION-KEY) PAYLOAD SEQUENCE
1 2 3
4 5 6
7 8 9
522 ENCRYPTED BLOCK DIVIDED INTO PAYLOADS
1 4 7
2 5 8
3 6 9
A
B
C
523 ENCRYPTED BLOCK DIVIDED INTO PAYLOADS INTERLEAVED
517 INTERLEAVE WINDOW
A
B
C
523 ENCRYPTED BLOCK DIVIDED INTO PAYLOADS INTERLEAVED
IPT A
IPT B
IPT C
340 TARP PACKETS WITH ENCRYPTED PAYLOADS

FIG. 3B

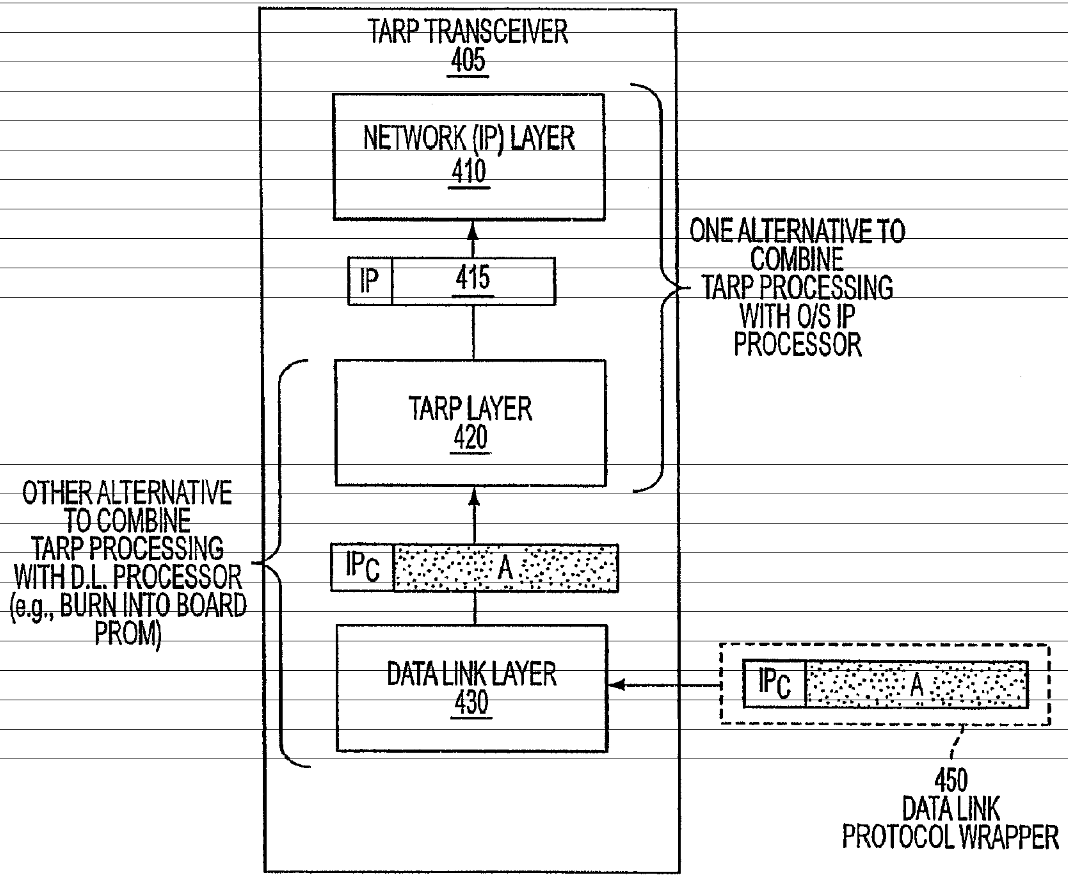

TARP TRANSCEIVER
405
NETWORK (IP) LAYER
410
IP
415
TARP LAYER
420
IPC
A
DATA LINK LAYER
430
ONE ALTERNATIVE TO COMBINE TARP PROCESSING WITH O/S IP PROCESSOR
OTHER ALTERNATIVE TO COMBINE TARP PROCESSING WITH D.L. PROCESSOR (e.g., BURN INTO BOARD PROM)
IPC
A
450
DATA LINK PROTOCOL WRAPPER

FIG. 4

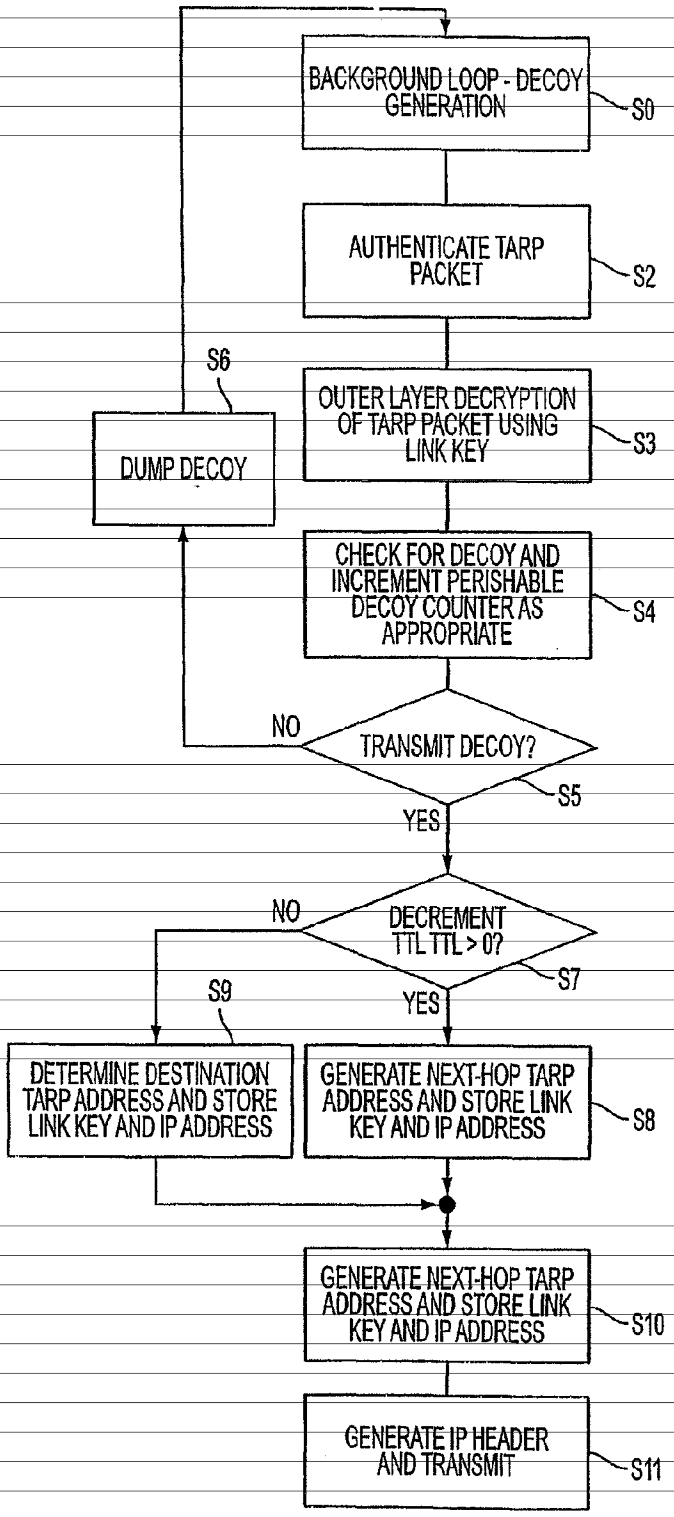

BACKGROUND LOOP - DECOY GENERATION
S0
AUTHENTICATE TARP PACKET
S2
OUTER LAYER DECRYPTION OF TARP PACKET USING LINK KEY
S3
S6
DUMP DECOY
CHECK FOR DECOY AND INCREMENT PERISHABLE DECOY COUNTER AS APPROPRIATE
S4
NO
TRANSMIT DECOY?
S5
YES
NO
DECREMENT TTL TTL > 0?
S7
YES
S9
DETERMINE DESTINATION TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
GENERATE NEXT-HOP TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
S8
GENERATE NEXT-HOP TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
S10
GENERATE IP HEADER AND TRANSMIT
S11

FIG. 5

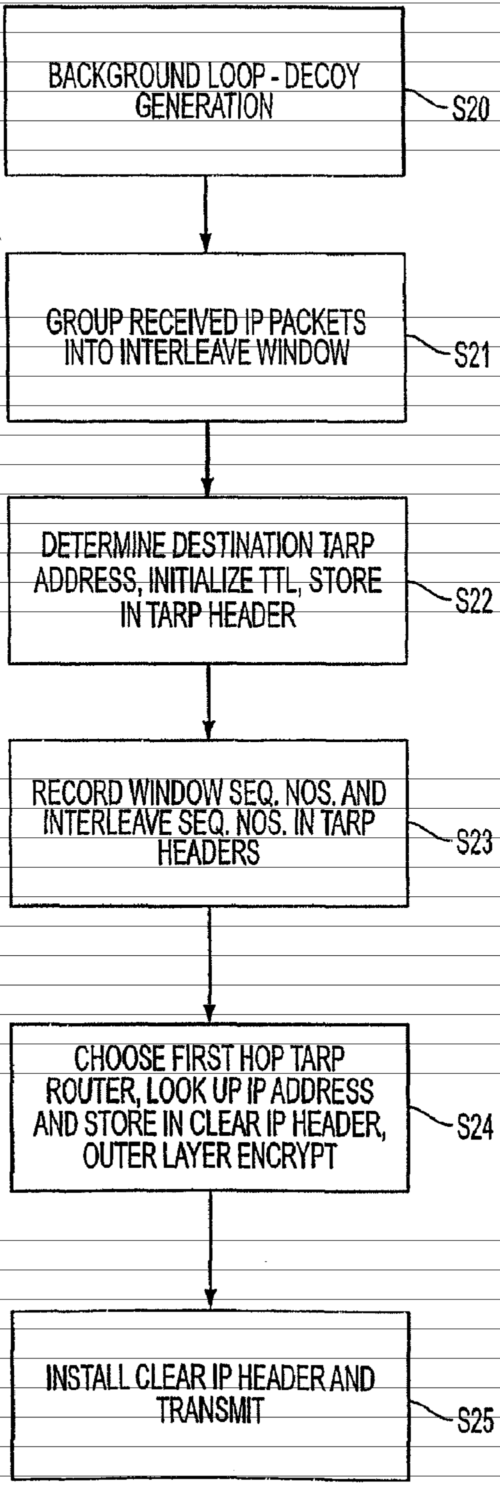

BACKGROUND LOOP - DECOY GENERATION
S20
GROUP RECEIVED IP PACKETS INTO INTERLEAVE WINDOW
S21
DETERMINE DESTINATION TARP ADDRESS, INITIALIZE TTL, STORE IN TARP HEADER
S22
RECORD WINDOW SEQ. NOS. AND INTERLEAVE SEQ. NOS. IN TARP HEADERS
S23
CHOOSE FIRST HOP TARP ROUTER, LOOK UP IP ADDRESS AND STORE IN CLEAR IP HEADER, OUTER LAYER ENCRYPT
S24
INSTALL CLEAR IP HEADER AND TRANSMIT
S25

FIG. 6

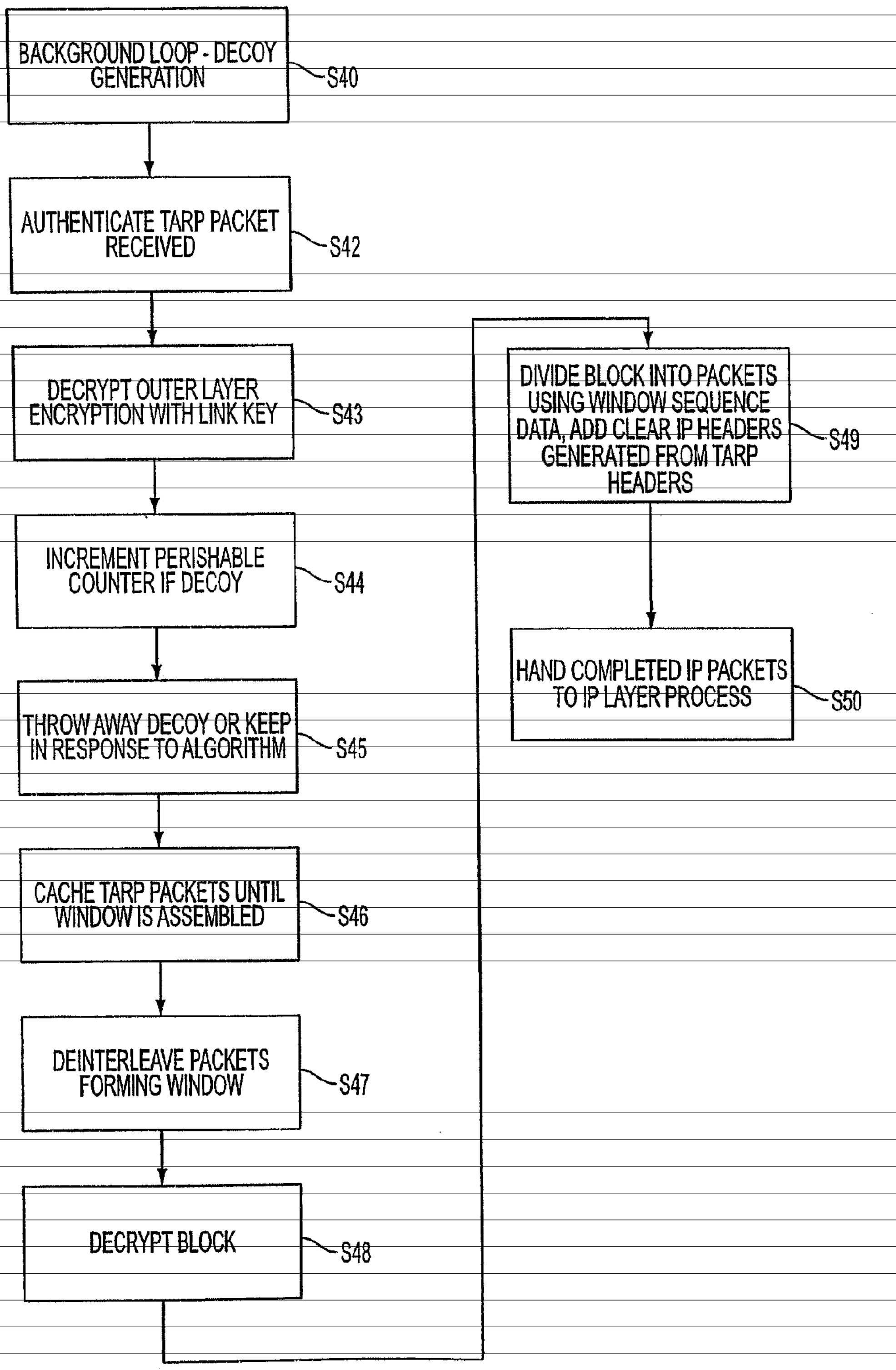

BACKGROUND LOOP - DECOY GENERATION
S40
AUTHENTICATE TARP PACKET RECEIVED
S42
DECRYPT OUTER LAYER ENCRYPTION WITH LINK KEY
S43
INCREMENT PERISHABLE COUNTER IF DECOY
S44
THROW AWAY DECOY OR KEEP IN RESPONSE TO ALGORITHM
S45
CACHE TARP PACKETS UNTIL WINDOW IS ASSEMBLED
S46
DEINTERLEAVE PACKETS FORMING WINDOW
S47
DECRYPT BLOCK
S48
DIVIDE BLOCK INTO PACKETS USING WINDOW SEQUENCE DATA, ADD CLEAR IP HEADERS GENERATED FROM TARP HEADERS
S49
HAND COMPLETED IP PACKETS TO IP LAYER PROCESS
S50

FIG. 7

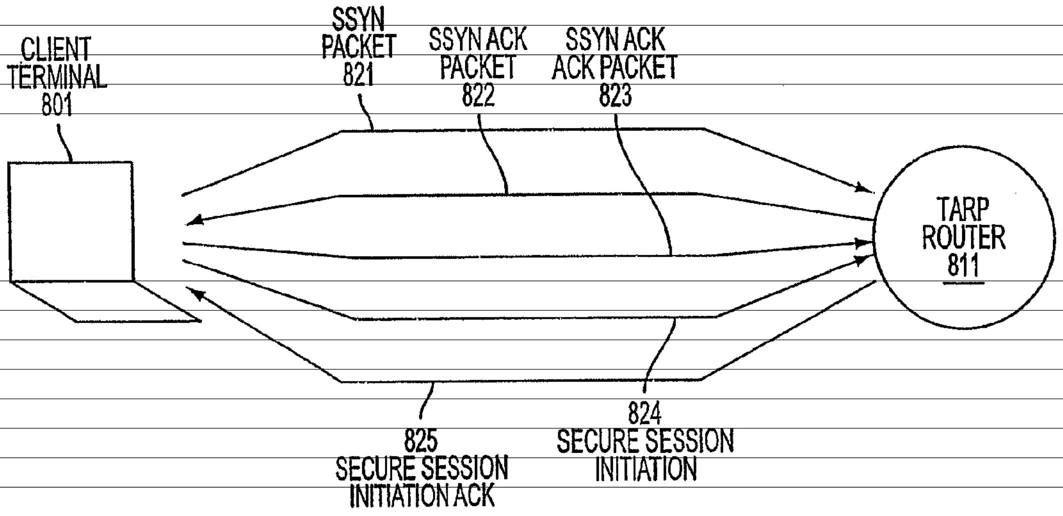
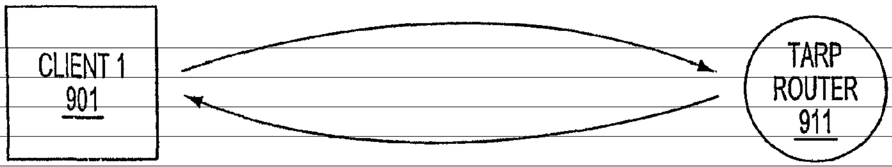

CLIENT TERMINAL 801
SSYN PACKET 821
SSYN ACK PACKET 822
SSYN ACK ACK PACKET 823
TARP ROUTER 811
825 SECURE SESSION INITIATION ACK
824 SECURE SESSION INITIATION

FIG. 8



TRANSMIT TABLE
921

131.218.204.98	•	131.218.204.65
131.218.204.221	•	131.218.204.97
131.218.204.139	•	131.218.204.186
131.218.204.12	•	131.218.204.55
⋮		⋮

RECEIVE TABLE
924

131.218.204.98	•	131.218.204.65
131.218.204.221	•	131.218.204.97
131.218.204.139	•	131.218.204.186
131.218.204.12	•	131.218.204.55
⋮		⋮

RECEIVE TABLE
922

131.218.204.161	•	131.218.204.89
131.218.204.66	•	131.218.204.212
131.218.204.201	•	131.218.204.127
131.218.204.119	•	131.218.204.49
⋮		⋮

TRANSMIT TABLE
923

131.218.204.161	•	131.218.204.89
131.218.204.66	•	131.218.204.212
131.218.204.201	•	131.218.204.127
131.218.204.119	•	131.218.204.49
⋮		⋮

FIG. 9

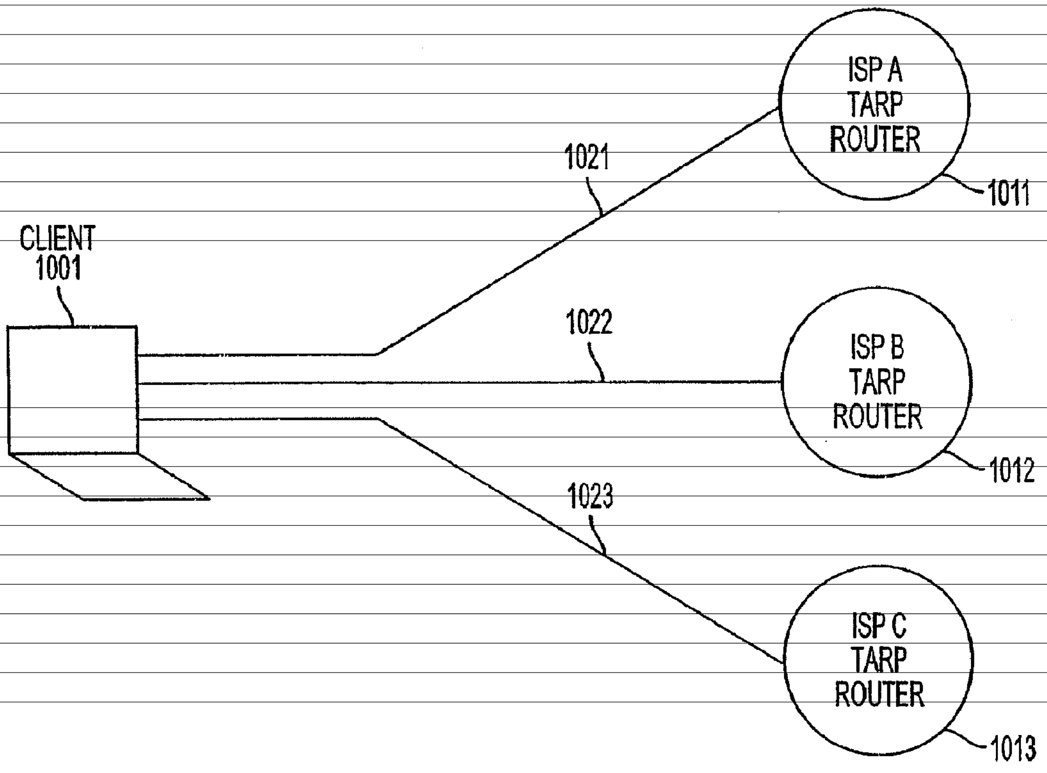

ISP A
TARP
ROUTER
1021
1011
CLIENT
1001
1022
ISP B
TARP
ROUTER
1012
1023
ISP C
TARP
ROUTER
1013

FIG. 10

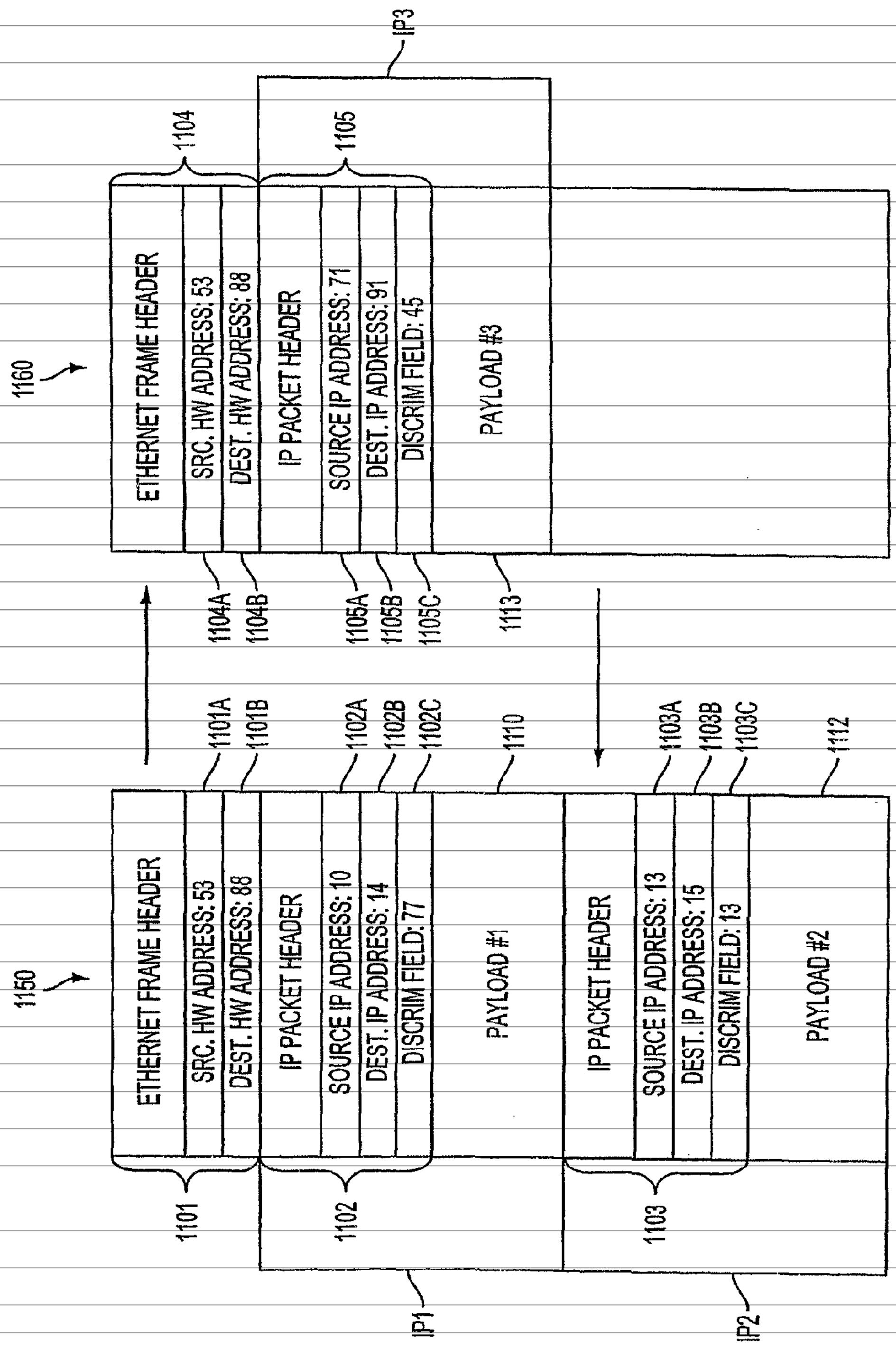

1150
1101
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
1101A
1101B
1102
IP PACKET HEADER
SOURCE IP ADDRESS: 10
DEST. IP ADDRESS: 14
DISCRIM FIELD: 77
1102A
1102B
1102C
PAYLOAD #1
1110
IP1
1103
IP PACKET HEADER
SOURCE IP ADDRESS: 13
DEST. IP ADDRESS: 15
DISCRIM FIELD: 13
1103A
1103B
1103C
PAYLOAD #2
1112
IP2
1160
1104
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
1104A
1104B
1105
IP PACKET HEADER
SOURCE IP ADDRESS: 71
DEST. IP ADDRESS: 91
DISCRIM FIELD: 45
1105A
1105B
1105C
PAYLOAD #3
1113
IP3

FIG. 11

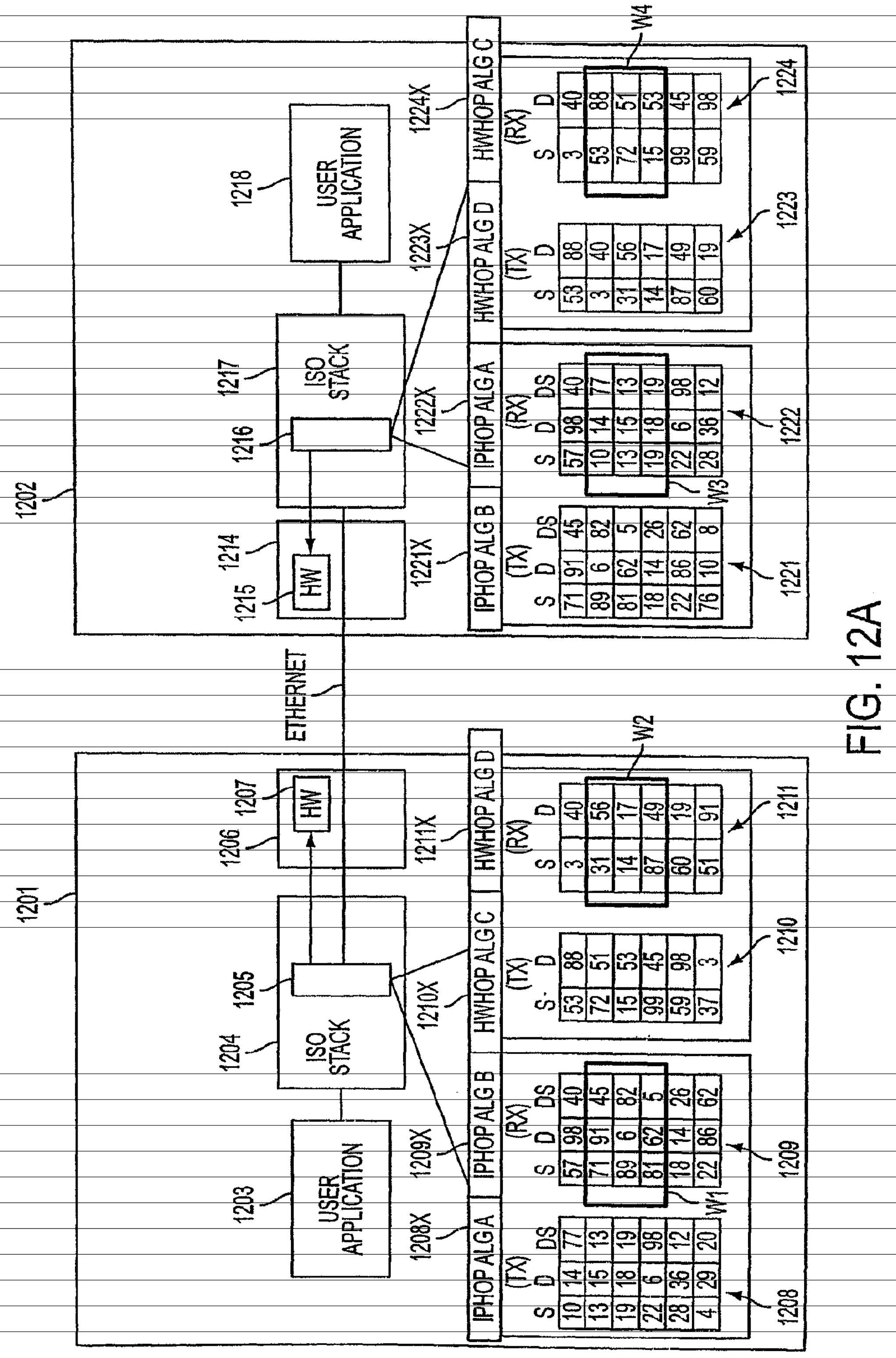

1201
1203
USER APPLICATION
1204
ISO STACK
1205
1206
1207
HW
ETHERNET
1208X
IPHOP ALG A
(TX)
S D DS
10 14 77
13 15 13
19 18 19
22 6 98
28 36 12
4 29 20
1208
1209X
IPHOP ALG B
(RX)
S D DS
57 98 40
71 91 45
89 6 82
81 62 5
18 14 26
22 86 62
W1
1209
1210X
HWHOP ALG C
(TX)
S D
53 88
72 51
15 53
99 45
59 98
37 3
1210
1211X
HWHOP ALG D
(RX)
S D
3 40
31 56
14 17
87 49
60 19
51 91
W2
1211
1202
1214
1215
HW
1216
1217
ISO STACK
1218
USER APPLICATION
1221X
IPHOP ALG B
(TX)
S D DS
71 91 45
89 6 82
81 62 5
18 14 26
22 86 62
76 10 8
1221
1222X
IPHOP ALG A
(RX)
S D DS
57 98 40
10 14 77
13 15 13
19 18 19
22 6 98
28 36 12
W3
1222
1223X
HWHOP ALG D
(TX)
S D
53 88
3 40
31 56
14 17
87 49
60 19
1223
1224X
HWHOP ALG C
(RX)
S D
3 40
53 88
72 51
15 53
99 45
59 98
W4
1224

FIG. 12A

MODE OR EMBODIMENT	HARDWARE ADDRESSES	IP ADDRESSES	DISCRIMINATOR FIELD VALUES
1. PROMISCUOUS	SAME FOR ALL NODES OR COMPLETELY RANDOM	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
2. PROMISCUOUS PER VPN	FIXED FOR EACH VPN	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
3. HARDWARE HOPPING	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC

FIG. 12B

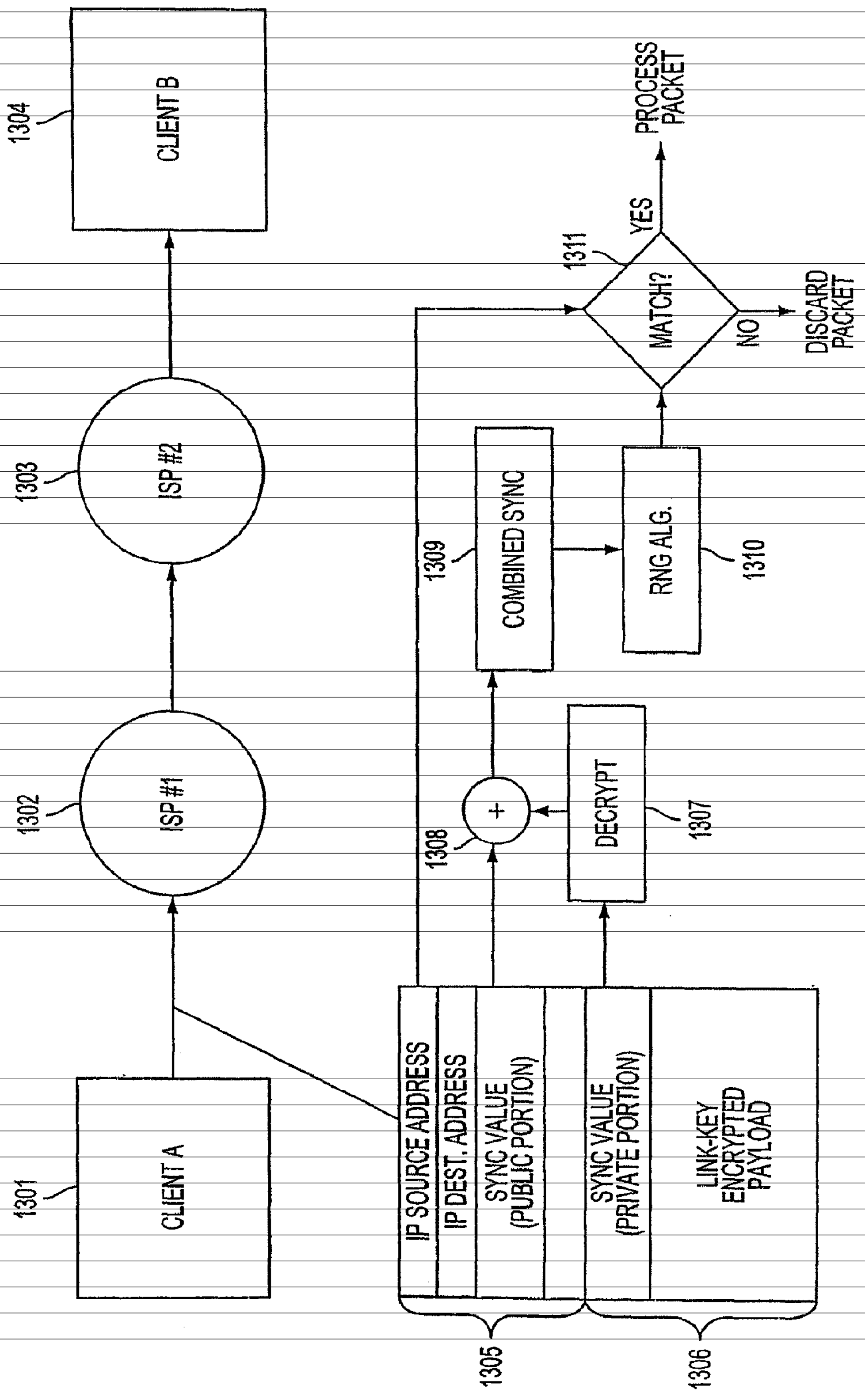

1301
CLIENT A
1302
ISP #1
1303
ISP #2
1304
CLIENT B
IP SOURCE ADDRESS
IP DEST. ADDRESS
SYNC VALUE (PUBLIC PORTION)
SYNC VALUE (PRIVATE PORTION)
LINK-KEY ENCRYPTED PAYLOAD
1305
1306
DECRYPT
1307
+
1308
COMBINED SYNC
1309
RNG ALG.
1310
MATCH?
1311
YES
PROCESS PACKET
NO
DISCARD PACKET

FIG. 13

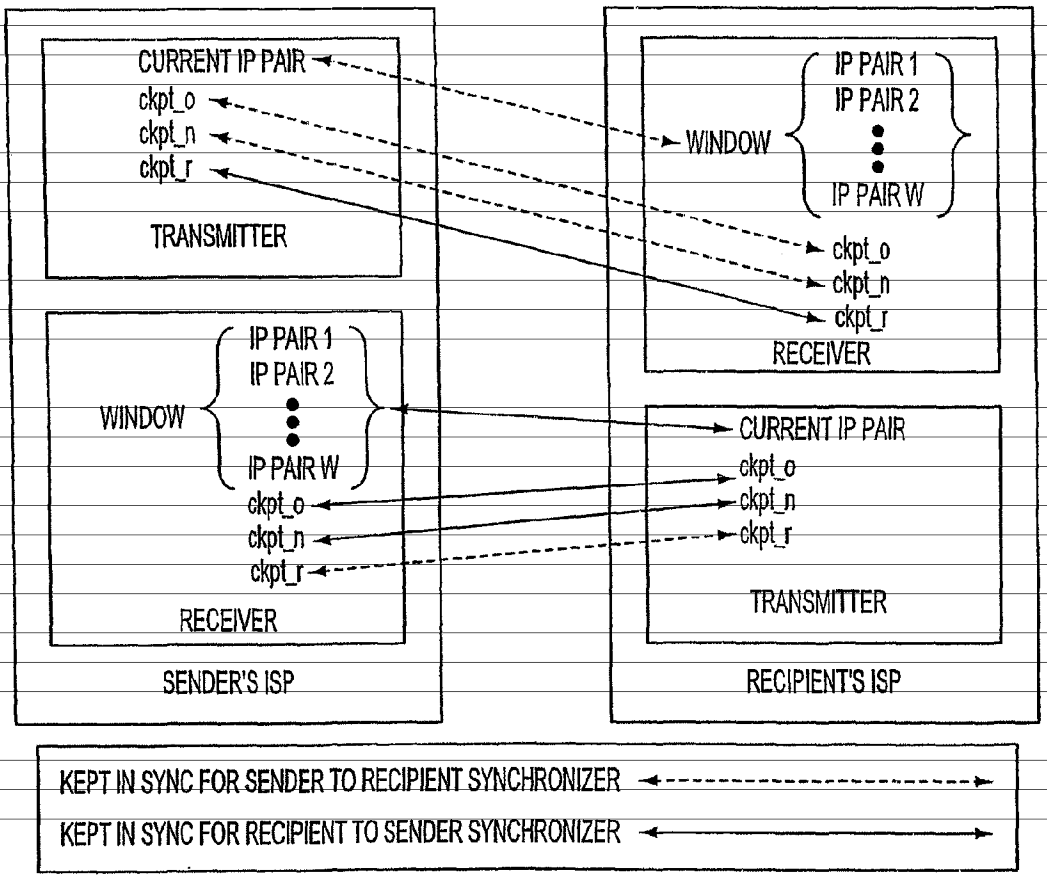

CURRENT IP PAIR
ckpt_o
ckpt_n
ckpt_r
TRANSMITTER
WINDOW
IP PAIR 1
IP PAIR 2
IP PAIR W
ckpt_o
ckpt_n
ckpt_r
RECEIVER
SENDER'S ISP
WINDOW
IP PAIR 1
IP PAIR 2
IP PAIR W
ckpt_o
ckpt_n
ckpt_r
RECEIVER
CURRENT IP PAIR
ckpt_o
ckpt_n
ckpt_r
TRANSMITTER
RECIPIENT'S ISP
KEPT IN SYNC FOR SENDER TO RECIPIENT SYNCHRONIZER
KEPT IN SYNC FOR RECIPIENT TO SENDER SYNCHRONIZER

FIG. 14

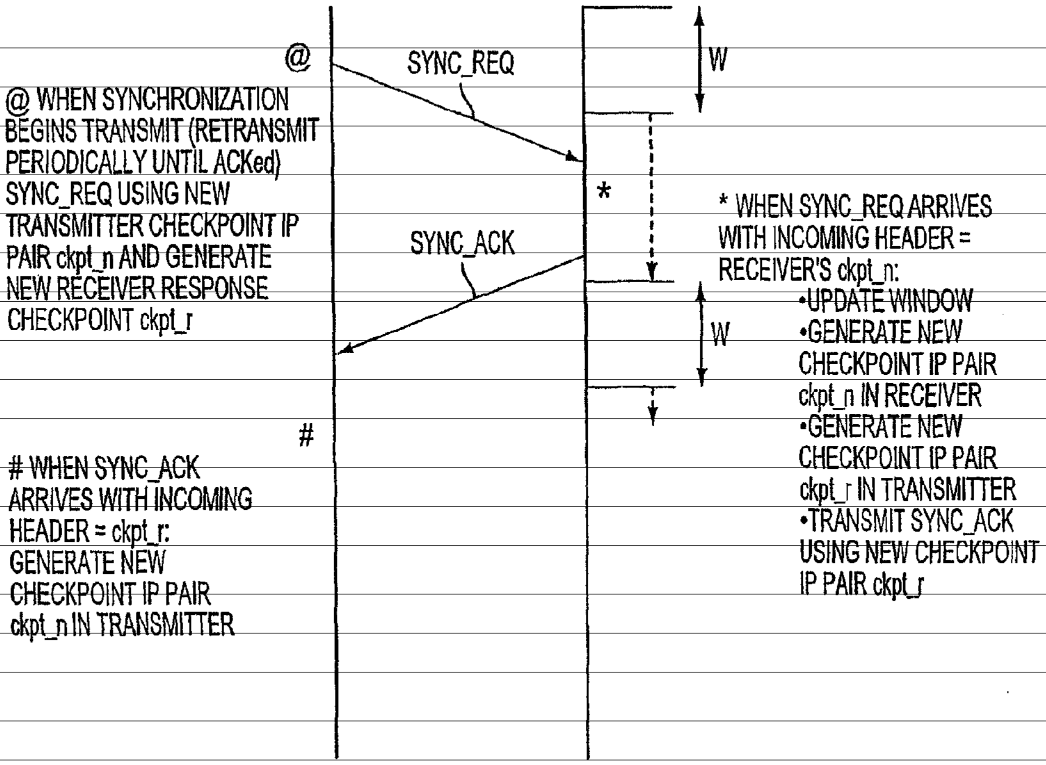

@
SYNC_REQ
W
@ WHEN SYNCHRONIZATION BEGINS TRANSMIT (RETRANSMIT PERIODICALLY UNTIL ACKed) SYNC_REQ USING NEW TRANSMITTER CHECKPOINT IP PAIR ckpt_n AND GENERATE NEW RECEIVER RESPONSE CHECKPOINT ckpt_r
*
SYNC_ACK
* WHEN SYNC_REQ ARRIVES WITH INCOMING HEADER = RECEIVER'S ckpt_n:
•UPDATE WINDOW
•GENERATE NEW CHECKPOINT IP PAIR ckpt_n IN RECEIVER
•GENERATE NEW CHECKPOINT IP PAIR ckpt_r IN TRANSMITTER
•TRANSMIT SYNC_ACK USING NEW CHECKPOINT IP PAIR ckpt_r
W
#
# WHEN SYNC_ACK ARRIVES WITH INCOMING HEADER = ckpt_r: GENERATE NEW CHECKPOINT IP PAIR ckpt_n IN TRANSMITTER

FIG. 15

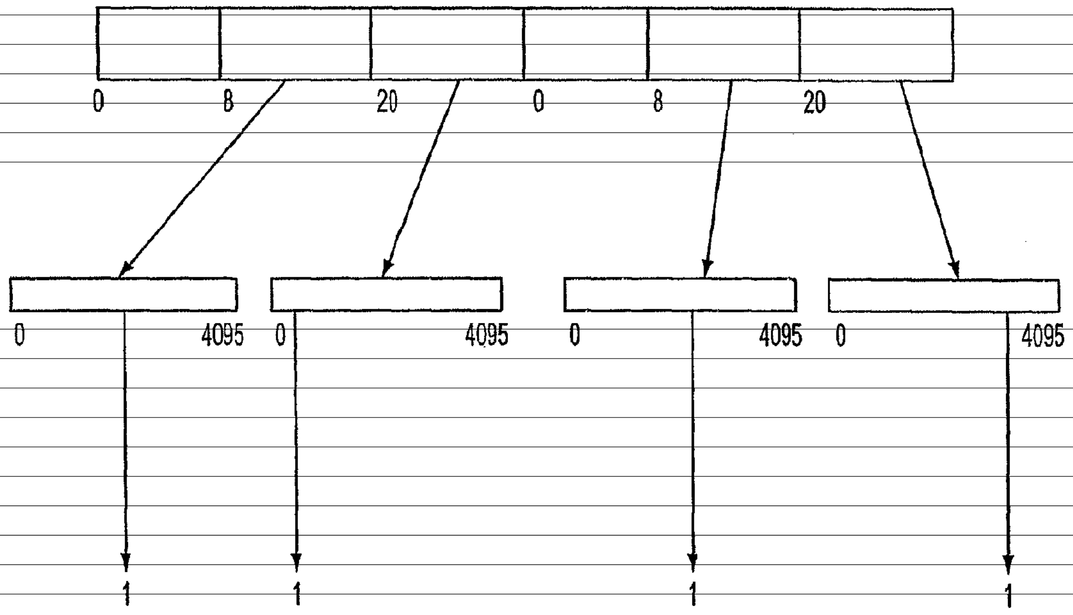


FIG. 16

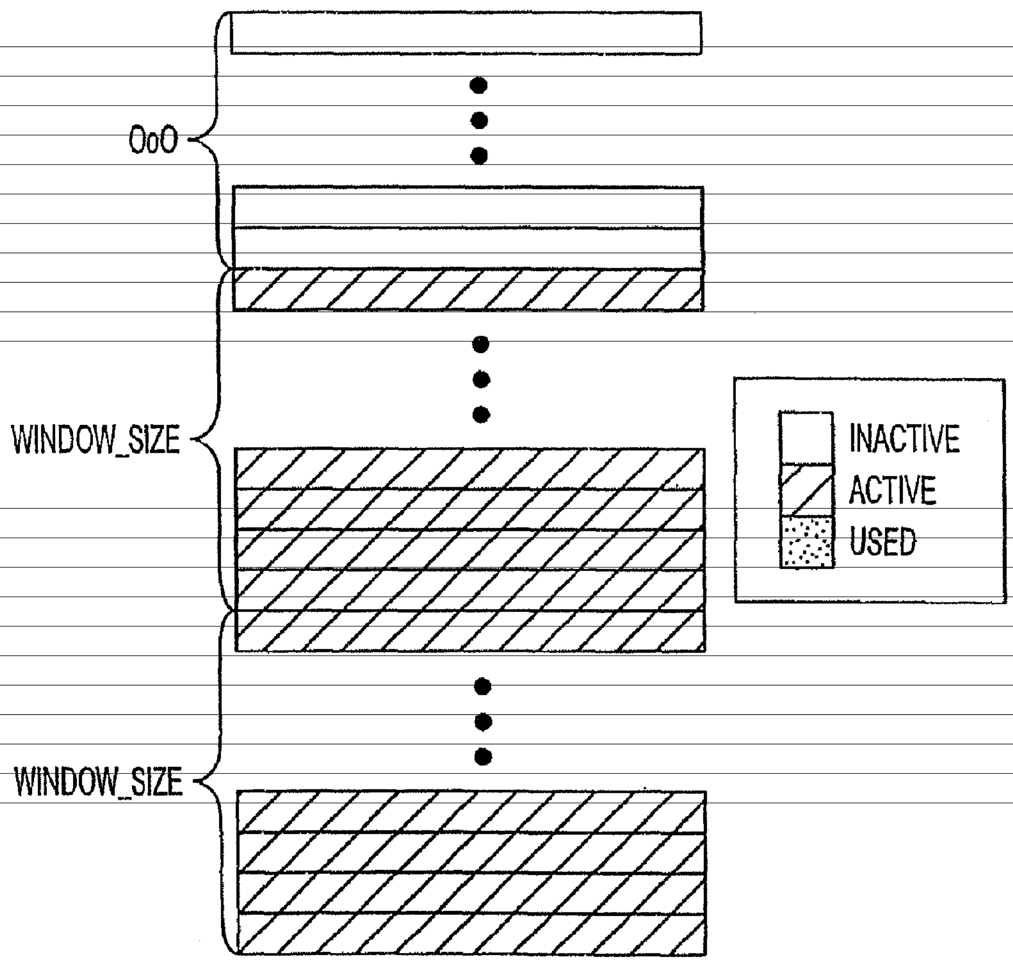


FIG. 17

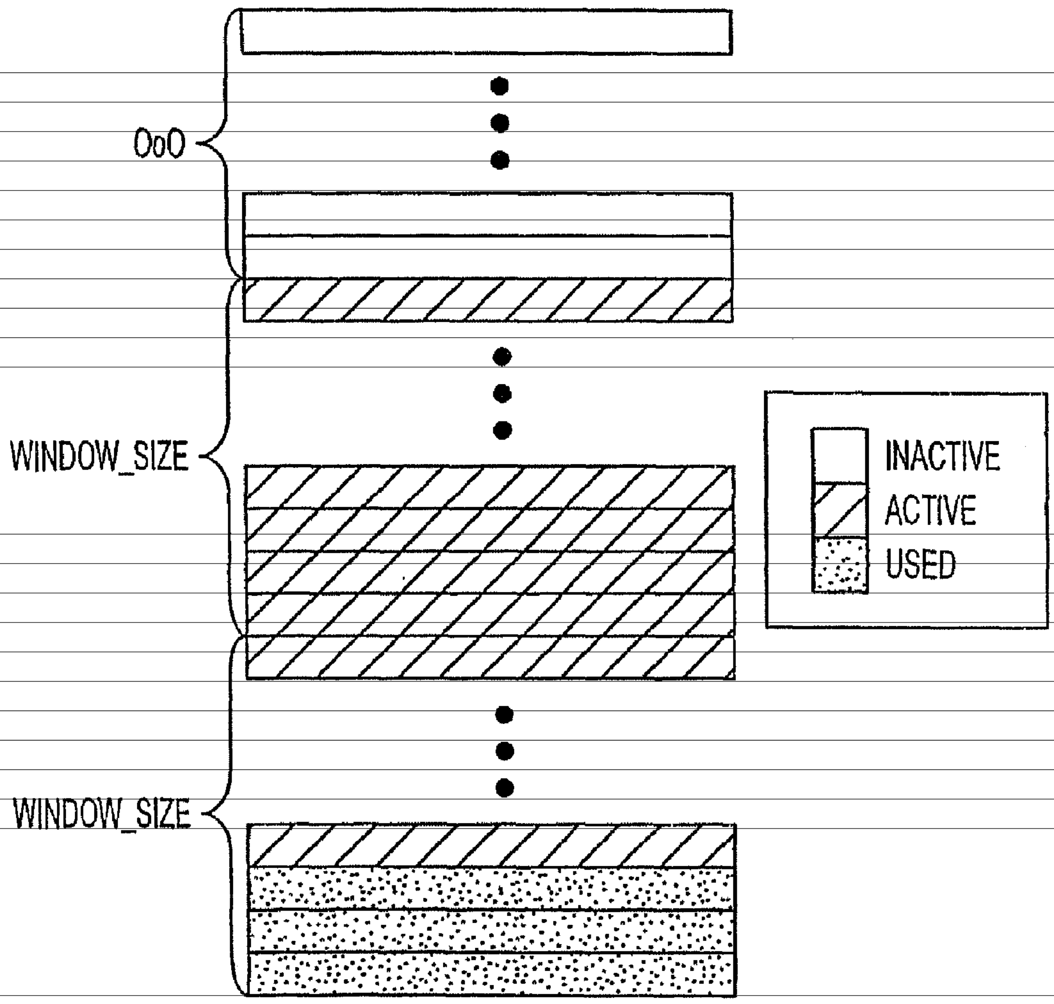


FIG. 18

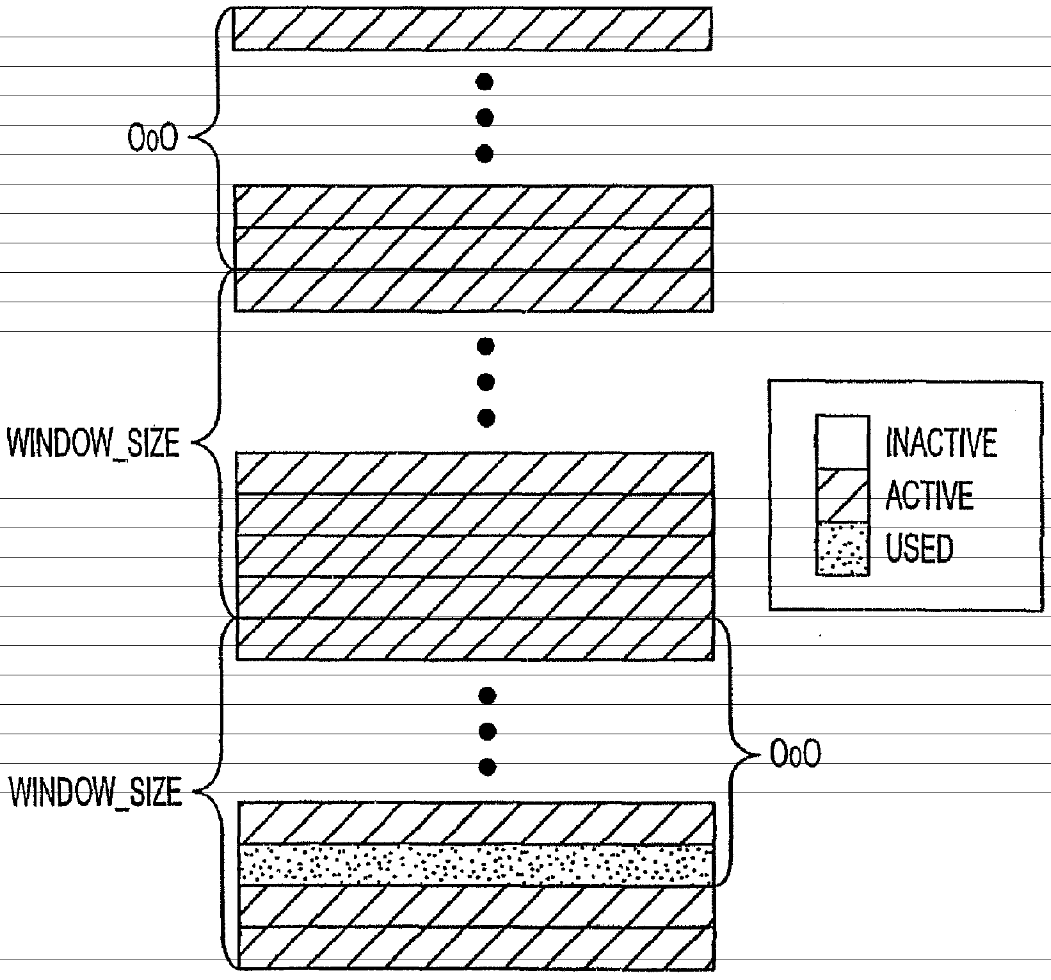


FIG. 19

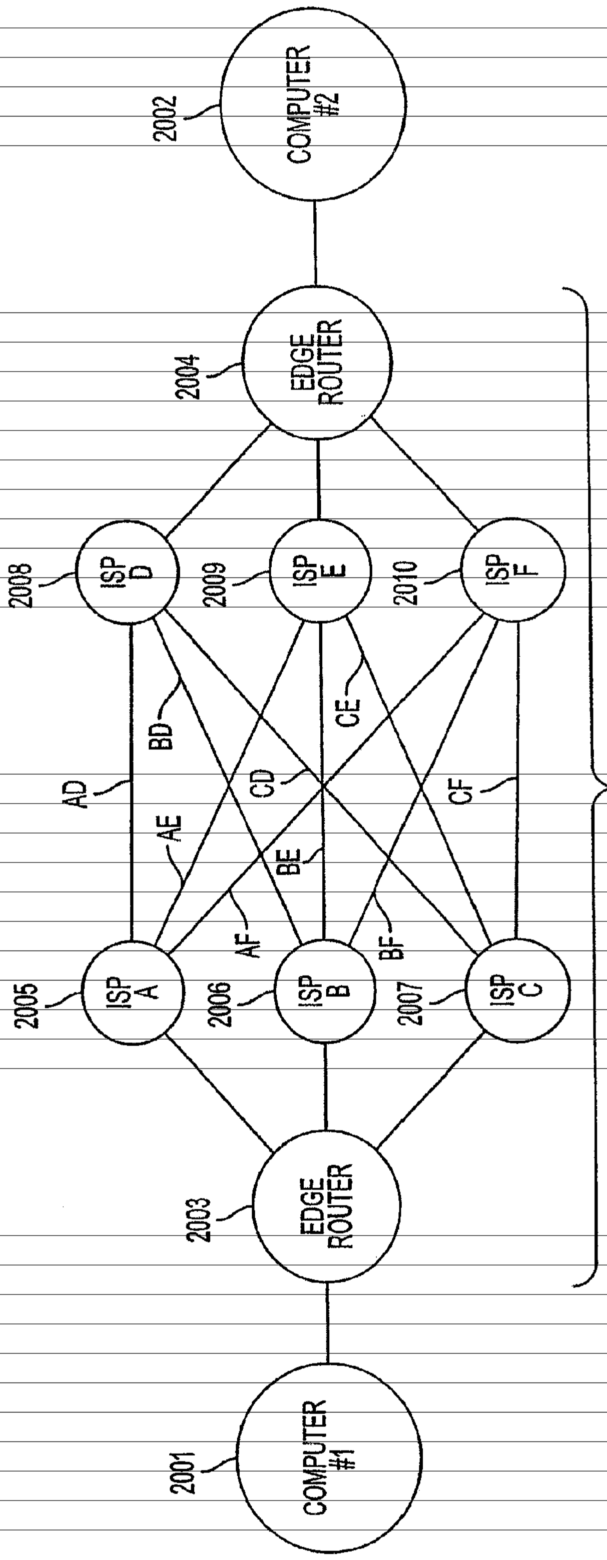
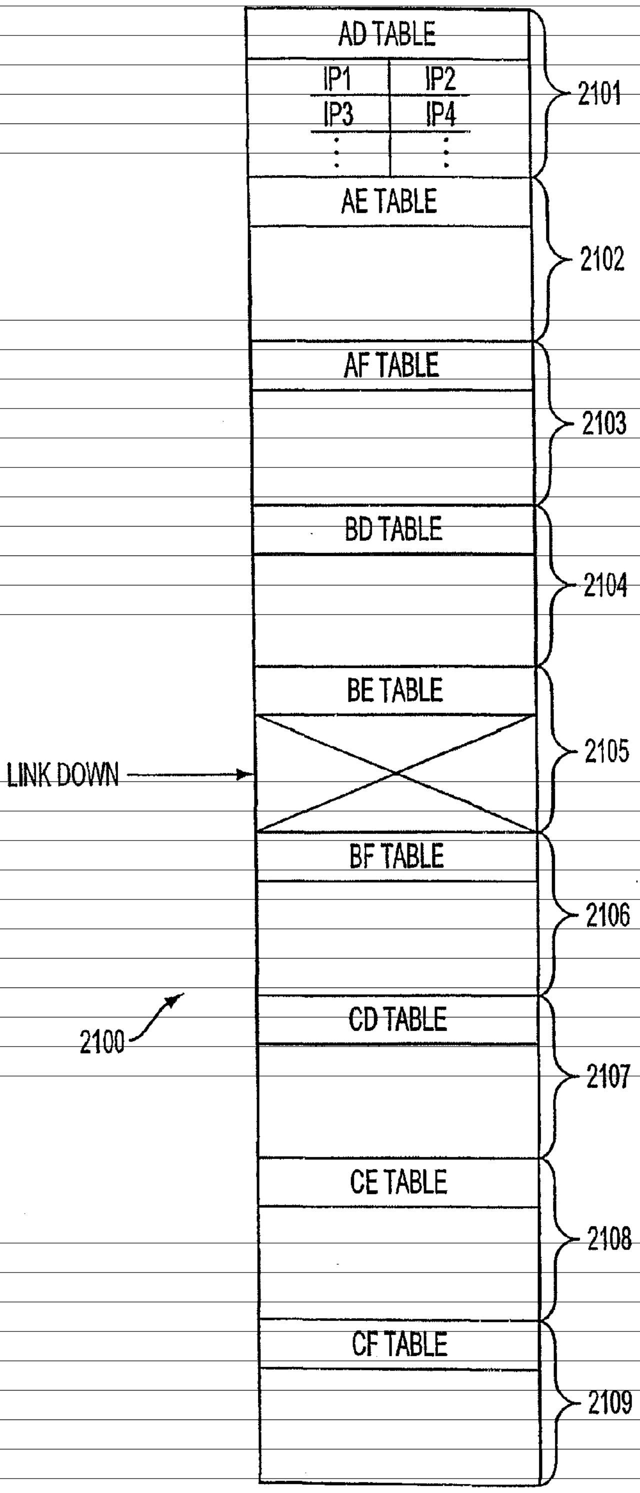

2002
COMPUTER #2
2004
EDGE ROUTER
2008
ISP D
2009
ISP E
2010
ISP F
AD
BD
AE
CD
CE
BE
CF
AF
BF
2005
ISP A
2006
ISP B
2007
ISP C
2003
EDGE ROUTER
2001
COMPUTER #1
2011

FIG. 20



AD TABLE
IP1
IP2
IP3
IP4
2101
AE TABLE
2102
AF TABLE
2103
BD TABLE
2104
BE TABLE
LINK DOWN
2105
BF TABLE
2106
CD TABLE
2100
2107
CE TABLE
2108
CF TABLE
2109

FIG. 21

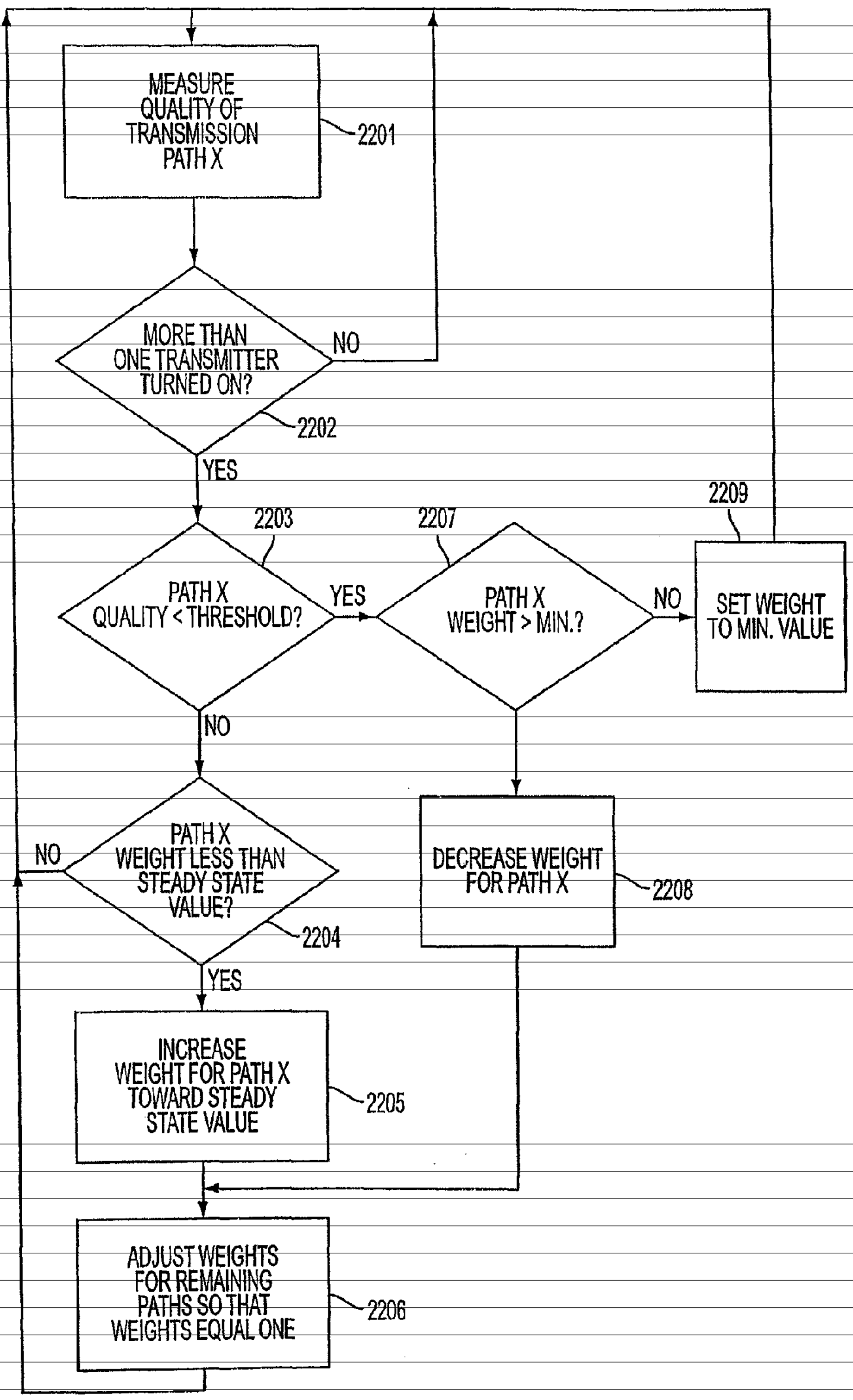

MEASURE QUALITY OF TRANSMISSION PATH X
2201
MORE THAN ONE TRANSMITTER TURNED ON?
NO
2202
YES
2203
PATH X QUALITY < THRESHOLD?
YES
2207
PATH X WEIGHT > MIN.?
NO
2209
SET WEIGHT TO MIN. VALUE
NO
PATH X WEIGHT LESS THAN STEADY STATE VALUE?
NO
2204
YES
DECREASE WEIGHT FOR PATH X
2208
INCREASE WEIGHT FOR PATH X TOWARD STEADY STATE VALUE
2205
ADJUST WEIGHTS FOR REMAINING PATHS SO THAT WEIGHTS EQUAL ONE
2206

FIG. 22A

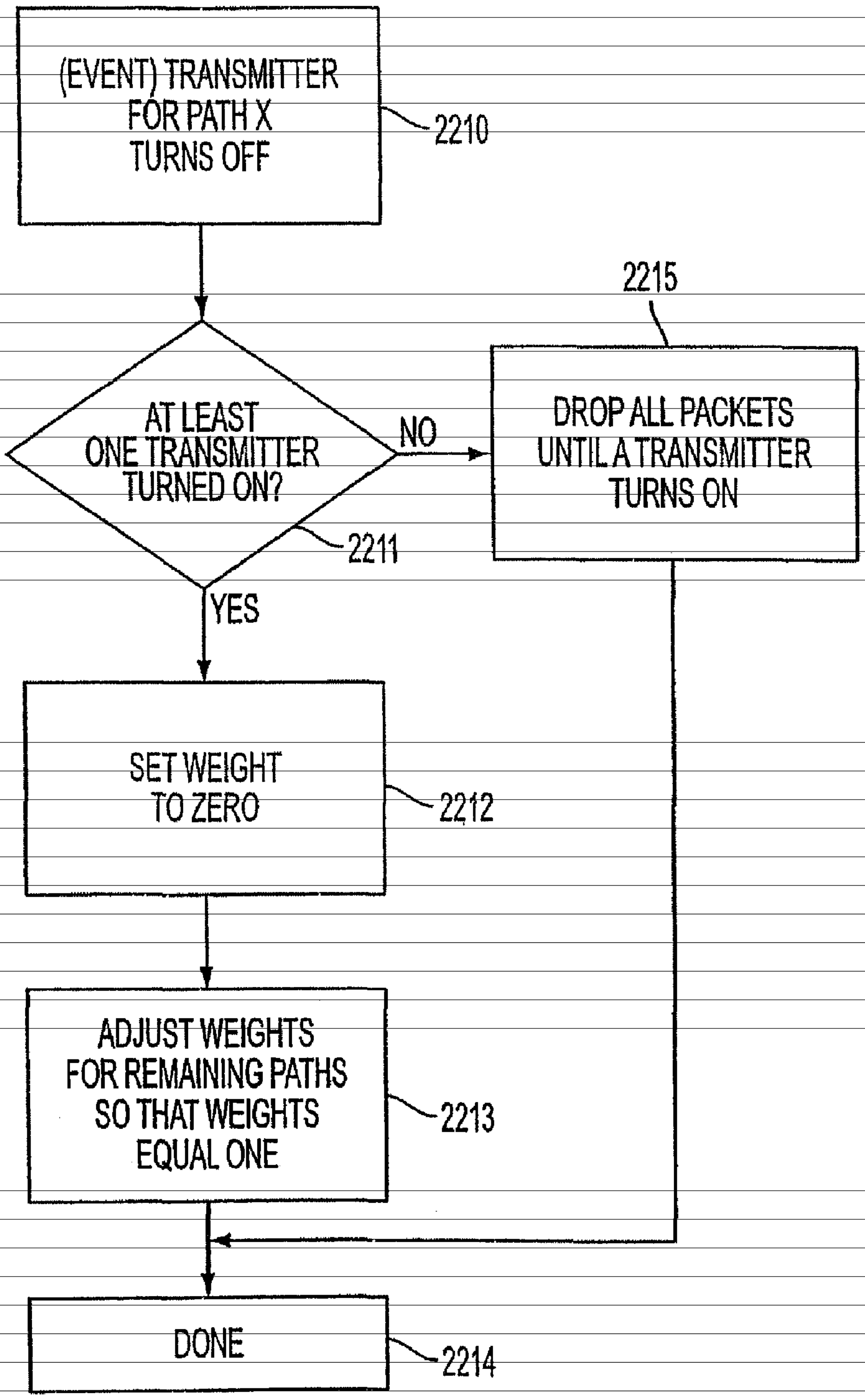

(EVENT) TRANSMITTER FOR PATH X TURNS OFF
2210
2215
AT LEAST ONE TRANSMITTER TURNED ON?
NO
DROP ALL PACKETS UNTIL A TRANSMITTER TURNS ON
2211
YES
SET WEIGHT TO ZERO
2212
ADJUST WEIGHTS FOR REMAINING PATHS SO THAT WEIGHTS EQUAL ONE
2213
DONE
2214

FIG. 22B

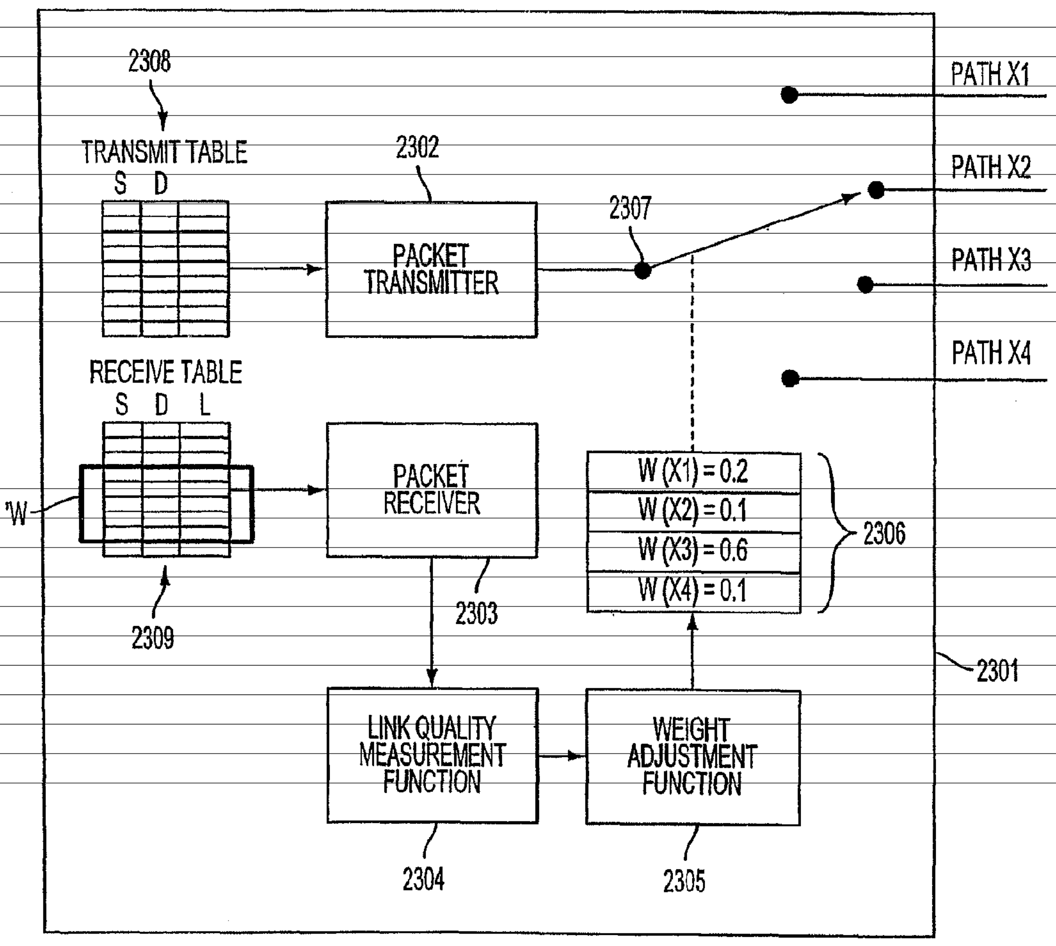

2308
TRANSMIT TABLE
S D
2302
PACKET TRANSMITTER
2307
PATH X1
PATH X2
PATH X3
PATH X4
RECEIVE TABLE
S D L
W
PACKET RECEIVER
W (X1) = 0.2
W (X2) = 0.1
W (X3) = 0.6
W (X4) = 0.1
2306
2309
2303
2301
LINK QUALITY MEASUREMENT FUNCTION
WEIGHT ADJUSTMENT FUNCTION
2304
2305

FIG. 23

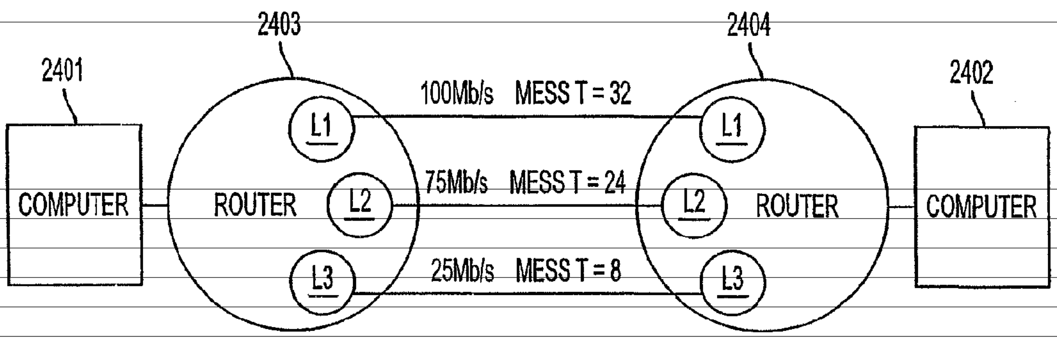

2401
COMPUTER
2403
ROUTER
L1
L2
L3
100Mb/s MESS T = 32
75Mb/s MESS T = 24
25Mb/s MESS T = 8
2404
L1
L2
L3
ROUTER
2402
COMPUTER

FIG. 24

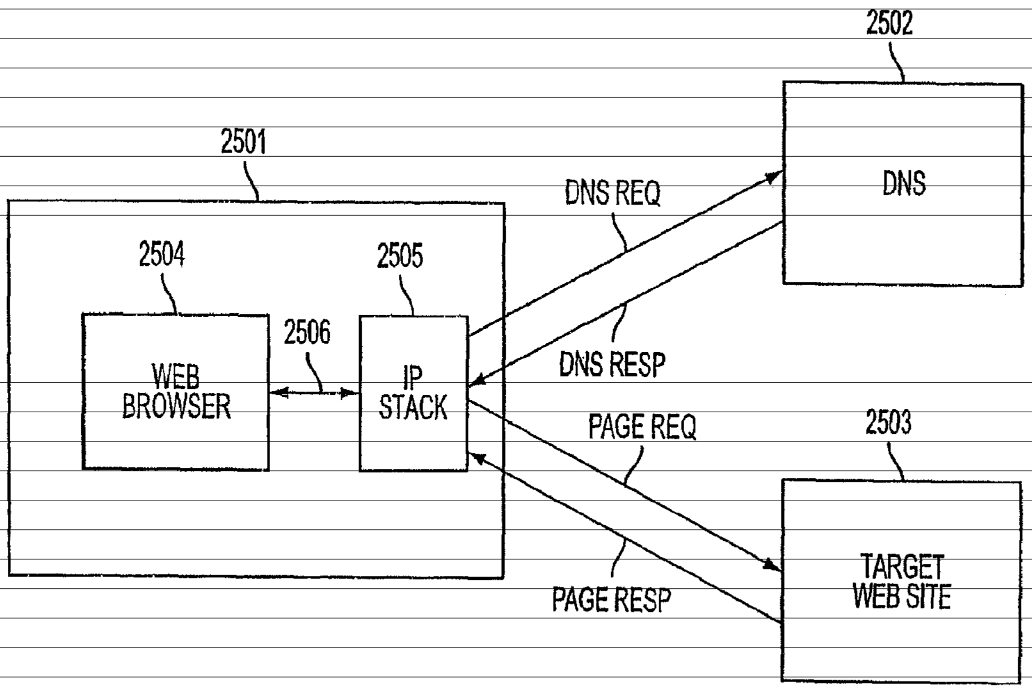


FIG. 25
(PRIOR ART)

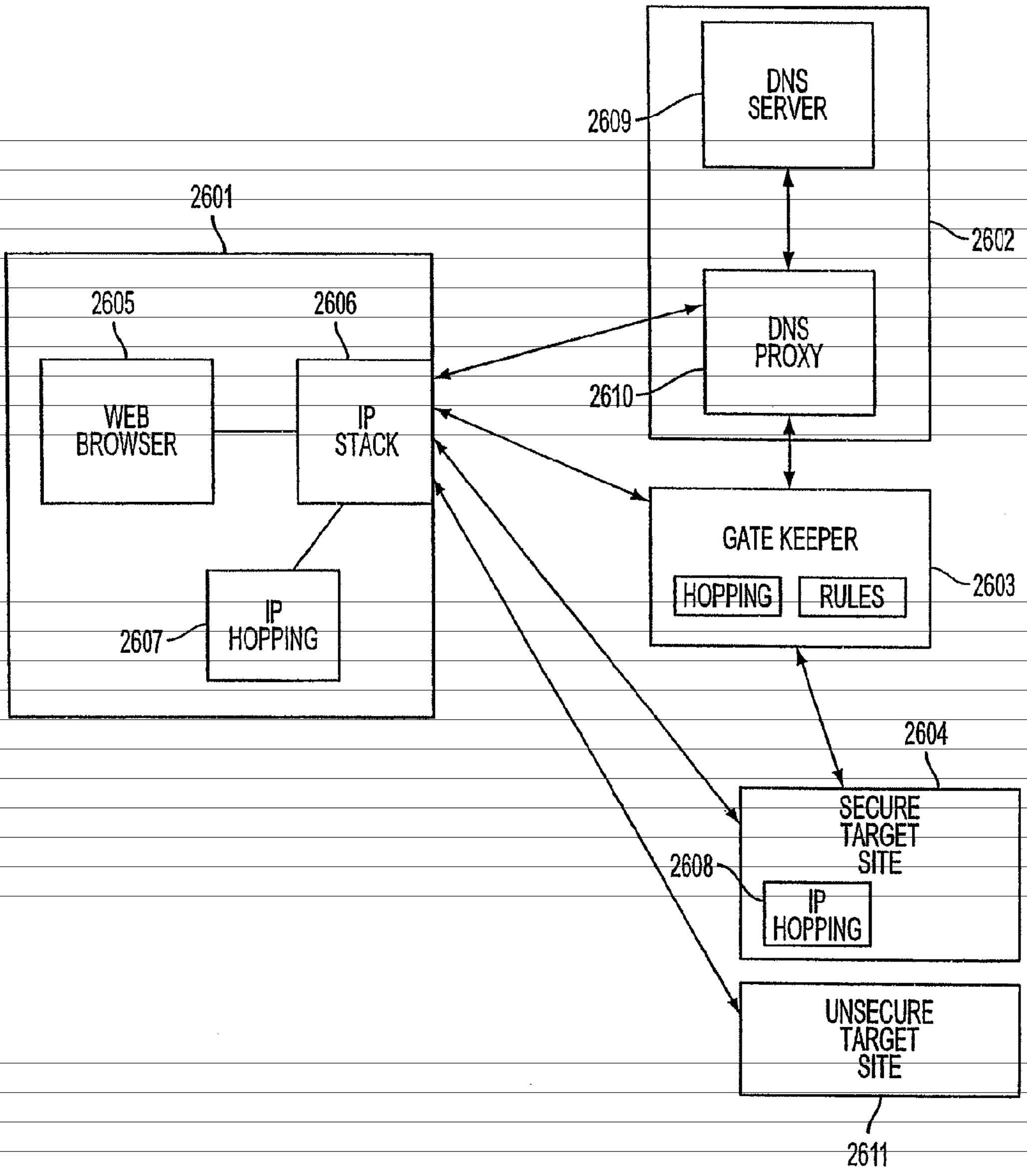

DNS SERVER
2609
2602
2601
DNS PROXY
2605
2606
2610
WEB BROWSER
IP STACK
GATE KEEPER
HOPPING
RULES
2603
2607
IP HOPPING
2604
SECURE TARGET SITE
2608
IP HOPPING
UNSECURE TARGET SITE
2611

FIG. 26

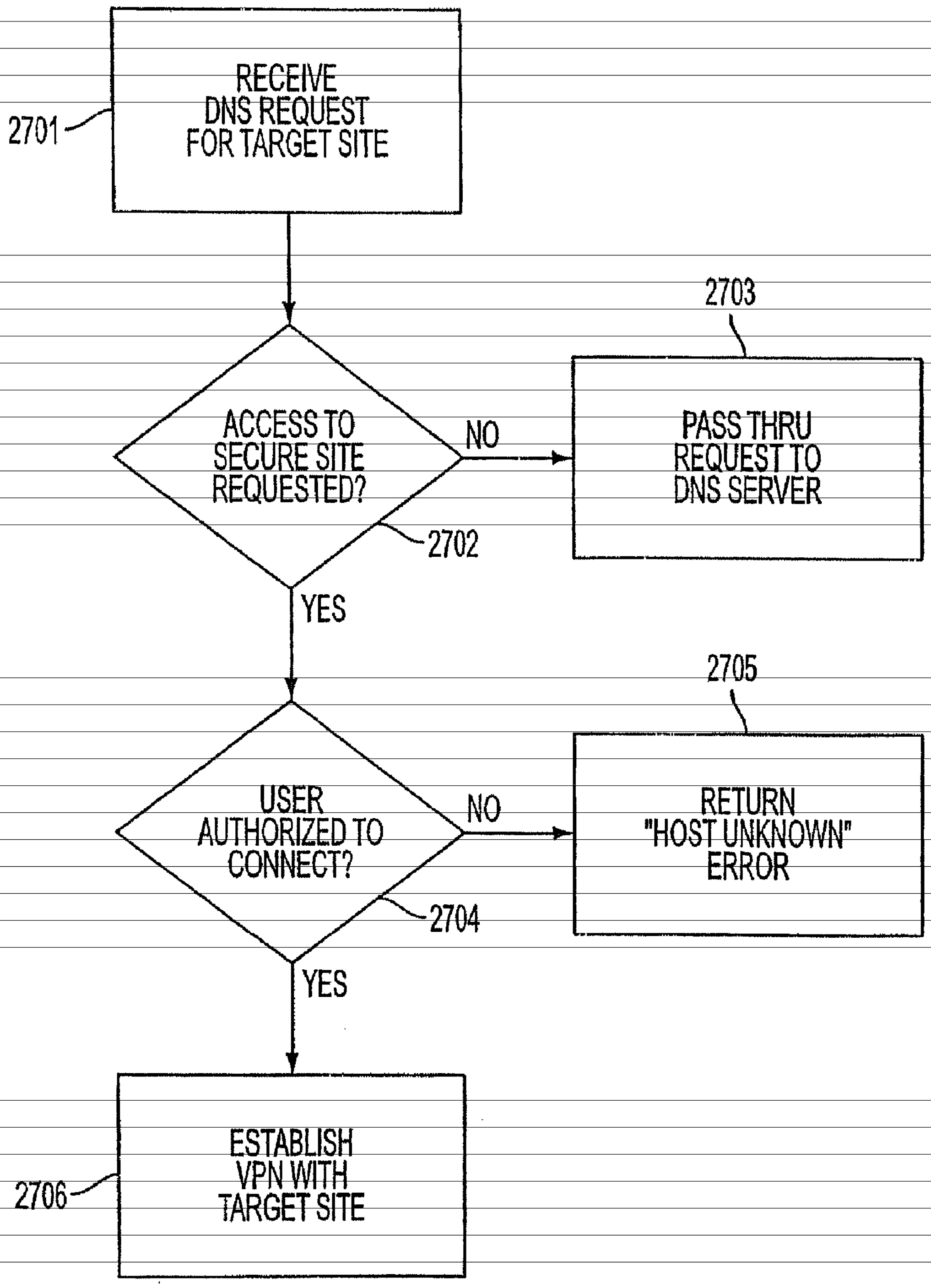

RECEIVE DNS REQUEST FOR TARGET SITE
2701
ACCESS TO SECURE SITE REQUESTED?
2702
NO
PASS THRU REQUEST TO DNS SERVER
2703
YES
USER AUTHORIZED TO CONNECT?
2704
NO
RETURN "HOST UNKNOWN" ERROR
2705
YES
ESTABLISH VPN WITH TARGET SITE
2706

FIG. 27

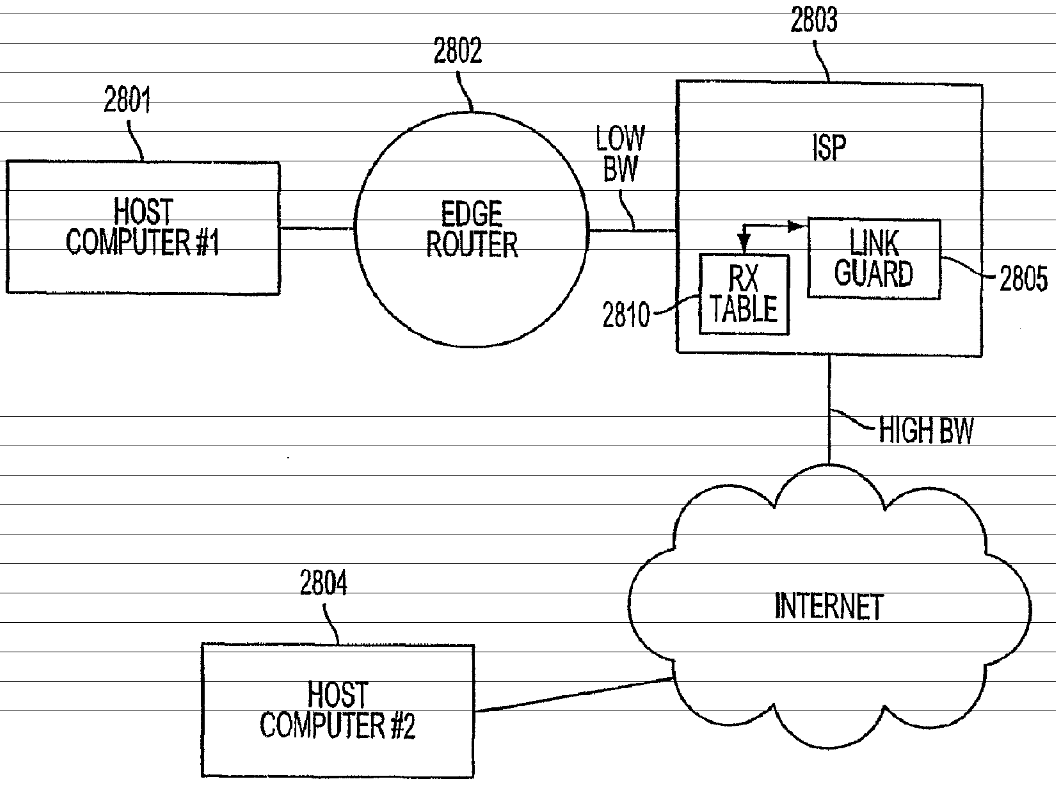

2801
HOST COMPUTER #1
2802
EDGE ROUTER
LOW BW
2803
ISP
RX TABLE
2810
LINK GUARD
2805
HIGH BW
INTERNET
2804
HOST COMPUTER #2

FIG. 28

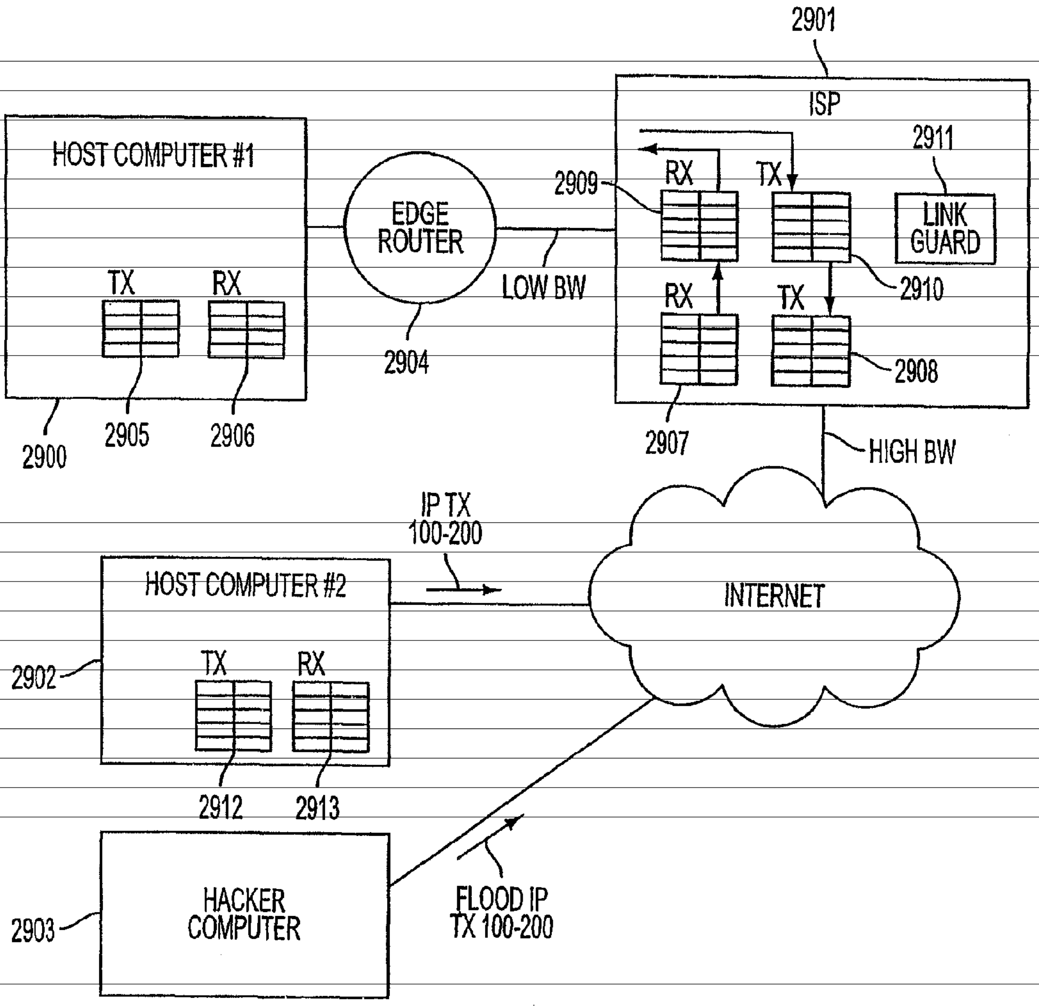

2901
ISP
2911
LINK GUARD
RX
TX
2909
2910
RX
TX
2908
2907
HOST COMPUTER #1
TX
RX
2905
2906
2900
EDGE ROUTER
2904
LOW BW
HIGH BW
INTERNET
IP TX 100-200
HOST COMPUTER #2
TX
RX
2902
2912
2913
HACKER COMPUTER
2903
FLOOD IP TX 100-200

FIG. 29

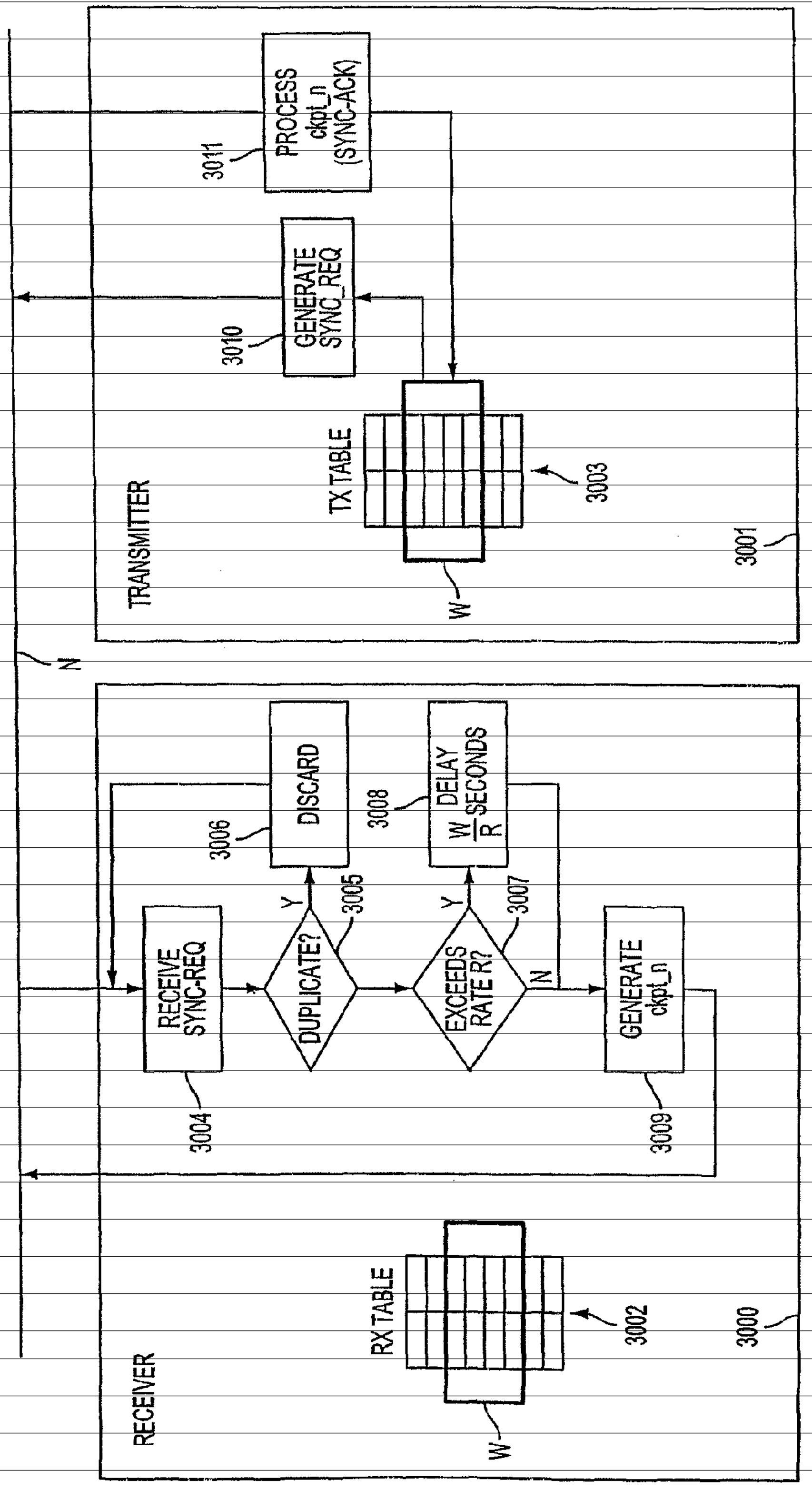

TRANSMITTER
3011
PROCESS ckpt_n (SYNC-ACK)
3010
GENERATE SYNC_REQ
TX TABLE
3003
W
3001
N
RECEIVER
3004
RECEIVE SYNC-REQ
DUPLICATE?
3005
Y
3006
DISCARD
EXCEEDS RATE R?
3007
Y
N
3008
DELAY $\frac{W}{R}$ SECONDS
GENERATE ckpt_n
3009
RX TABLE
3002
3000
W

FIG. 30

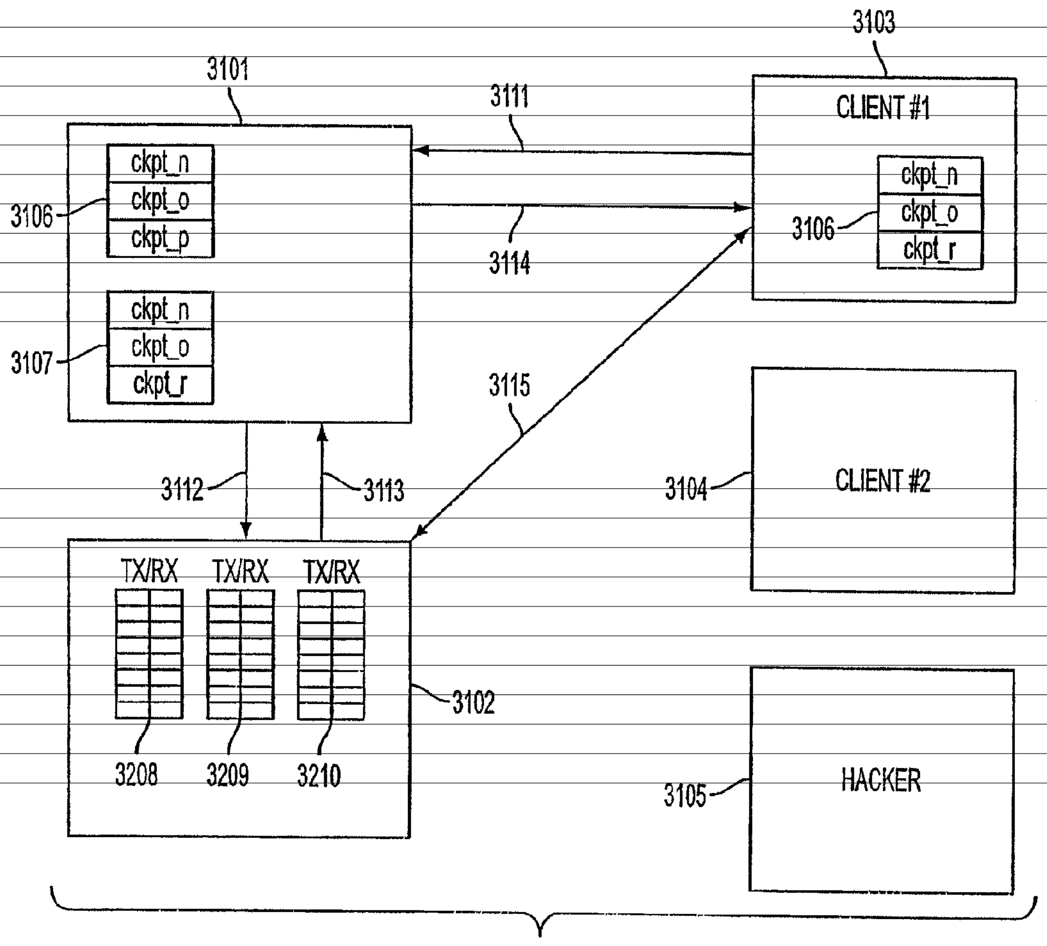

3101
3103
CLIENT #1
3111
3114
3115
ckpt_n
ckpt_o
ckpt_p
ckpt_n
ckpt_o
ckpt_r
ckpt_n
ckpt_o
ckpt_r
3106
3107
3106
3112
3113
3104
CLIENT #2
TX/RX
TX/RX
TX/RX
3102
3208
3209
3210
3105
HACKER

FIG. 31

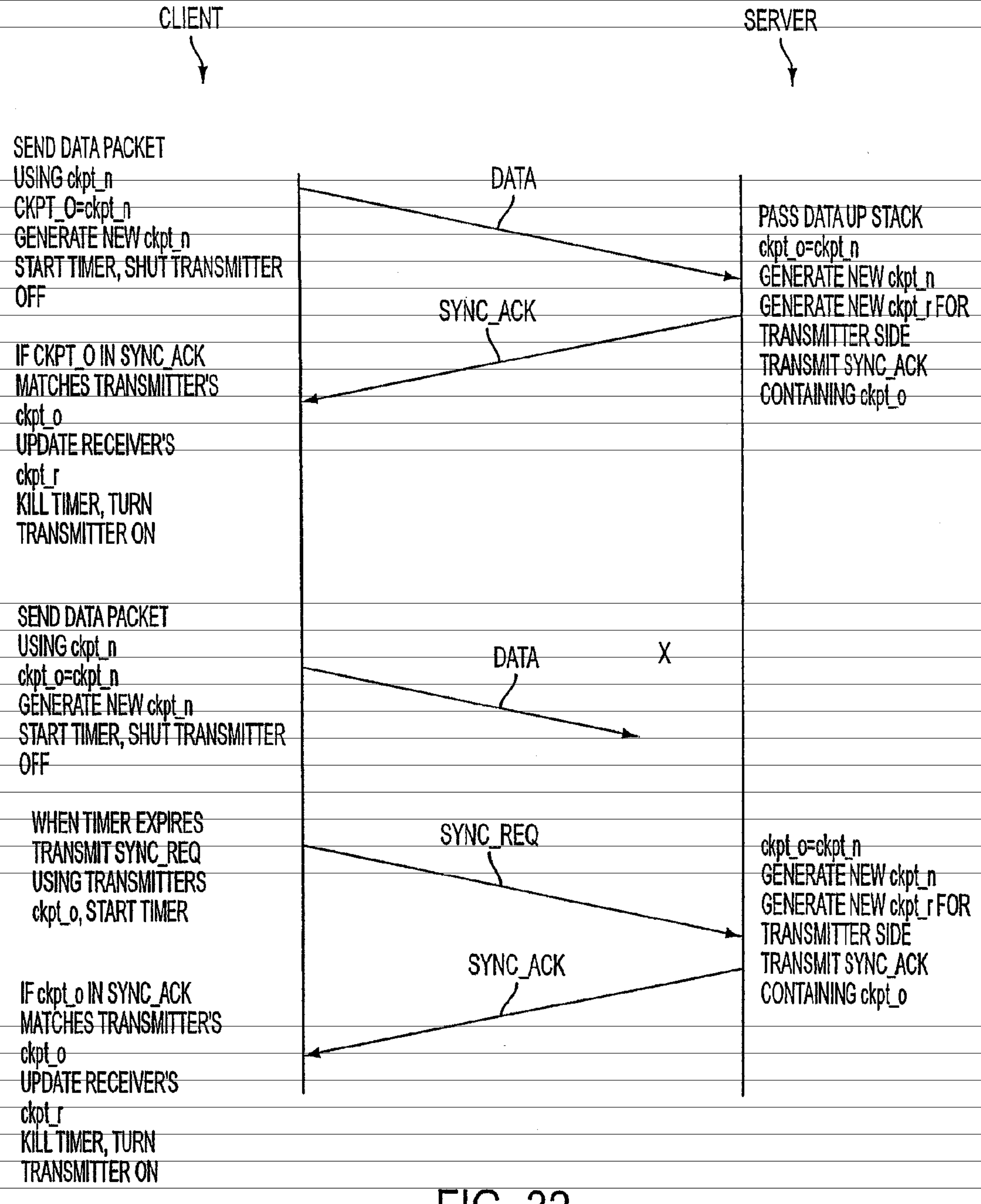

CLIENT
SERVER
SEND DATA PACKET
USING ckpt_n
CKPT_O=ckpt_n
GENERATE NEW ckpt_n
START TIMER, SHUT TRANSMITTER
OFF
DATA
PASS DATA UP STACK
ckpt_o=ckpt_n
GENERATE NEW ckpt_n
GENERATE NEW ckpt_r FOR
TRANSMITTER SIDE
TRANSMIT SYNC_ACK
CONTAINING ckpt_o
SYNC_ACK
IF CKPT_O IN SYNC_ACK
MATCHES TRANSMITTER'S
ckpt_o
UPDATE RECEIVER'S
ckpt_r
KILL TIMER, TURN
TRANSMITTER ON
SEND DATA PACKET
USING ckpt_n
ckpt_o=ckpt_n
GENERATE NEW ckpt_n
START TIMER, SHUT TRANSMITTER
OFF
DATA
X
WHEN TIMER EXPIRES
TRANSMIT SYNC_REQ
USING TRANSMITTERS
ckpt_o, START TIMER
SYNC_REQ
ckpt_o=ckpt_n
GENERATE NEW ckpt_n
GENERATE NEW ckpt_r FOR
TRANSMITTER SIDE
TRANSMIT SYNC_ACK
CONTAINING ckpt_o
SYNC_ACK
IF ckpt_o IN SYNC_ACK
MATCHES TRANSMITTER'S
ckpt_o
UPDATE RECEIVER'S
ckpt_r
KILL TIMER, TURN
TRANSMITTER ON

FIG. 32

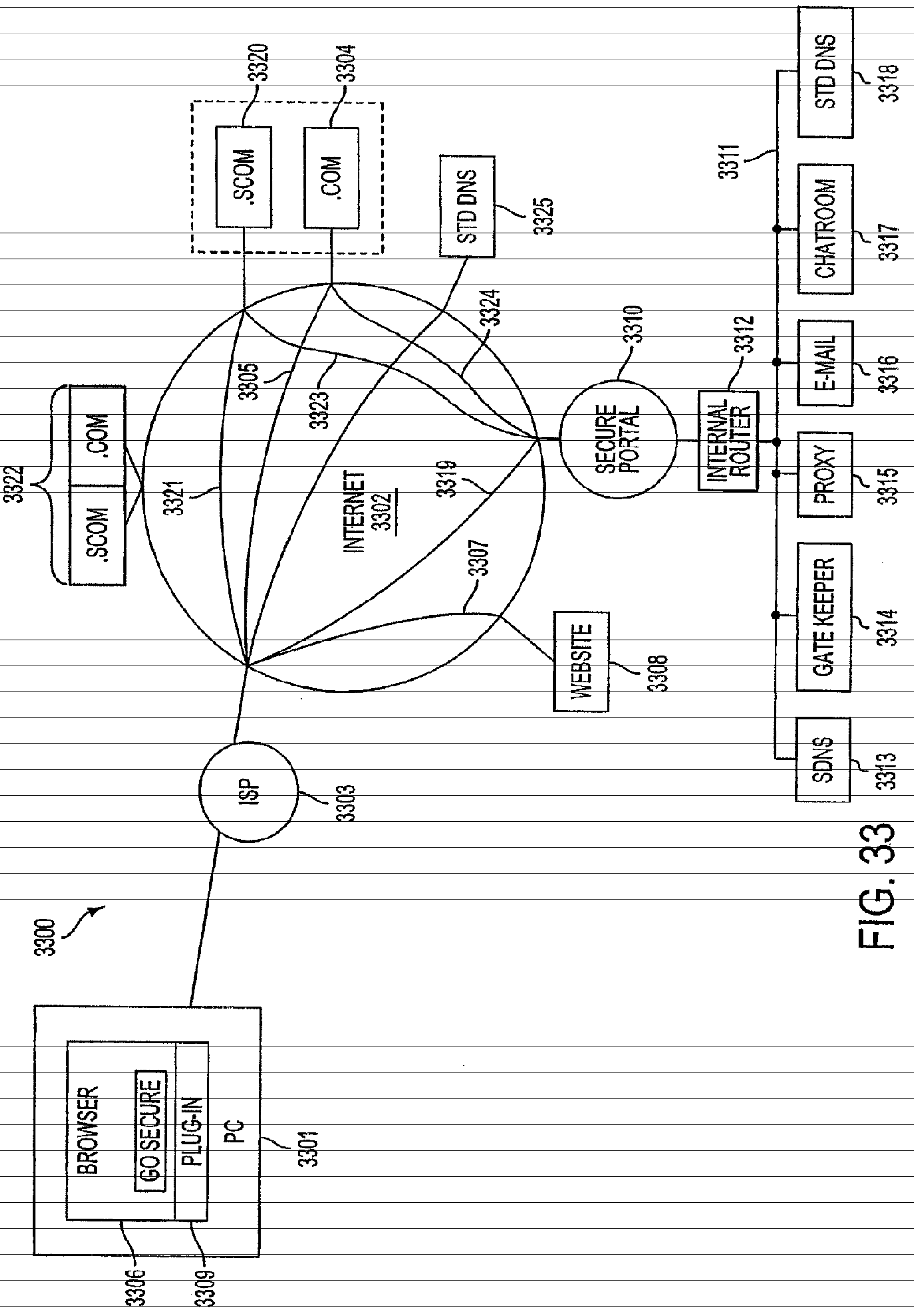

3300
BROWSER
GO SECURE
PLUG-IN
PC
3301
3306
3309
ISP
3303
INTERNET
3302
.SCOM
.COM
3322
3321
3305
3323
3319
3307
3320
3304
STD DNS
3325
3324
WEBSITE
3308
SECURE PORTAL
3310
INTERNAL ROUTER
3312
3311
SDNS
3313
GATE KEEPER
3314
PROXY
3315
E-MAIL
3316
CHATROOM
3317
STD DNS
3318

FIG. 33

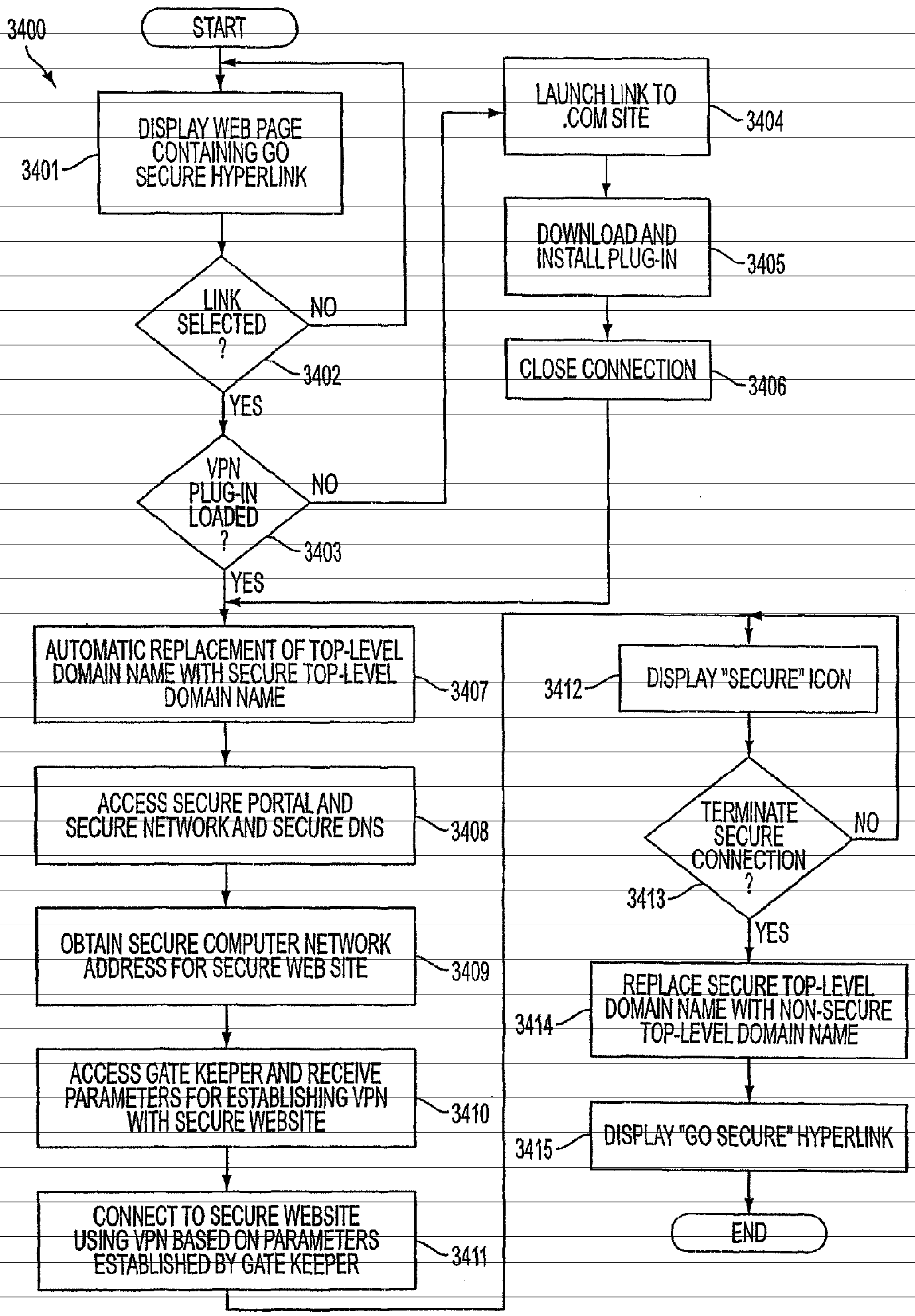

3400
START
DISPLAY WEB PAGE CONTAINING GO SECURE HYPERLINK
3401
LINK SELECTED ?
NO
3402
YES
VPN PLUG-IN LOADED ?
NO
3403
YES
LAUNCH LINK TO .COM SITE
3404
DOWNLOAD AND INSTALL PLUG-IN
3405
CLOSE CONNECTION
3406
AUTOMATIC REPLACEMENT OF TOP-LEVEL DOMAIN NAME WITH SECURE TOP-LEVEL DOMAIN NAME
3407
ACCESS SECURE PORTAL AND SECURE NETWORK AND SECURE DNS
3408
OBTAIN SECURE COMPUTER NETWORK ADDRESS FOR SECURE WEB SITE
3409
ACCESS GATE KEEPER AND RECEIVE PARAMETERS FOR ESTABLISHING VPN WITH SECURE WEBSITE
3410
CONNECT TO SECURE WEBSITE USING VPN BASED ON PARAMETERS ESTABLISHED BY GATE KEEPER
3411
3412
DISPLAY "SECURE" ICON
TERMINATE SECURE CONNECTION ?
NO
3413
YES
REPLACE SECURE TOP-LEVEL DOMAIN NAME WITH NON-SECURE TOP-LEVEL DOMAIN NAME
3414
DISPLAY "GO SECURE" HYPERLINK
3415
END

FIG. 34

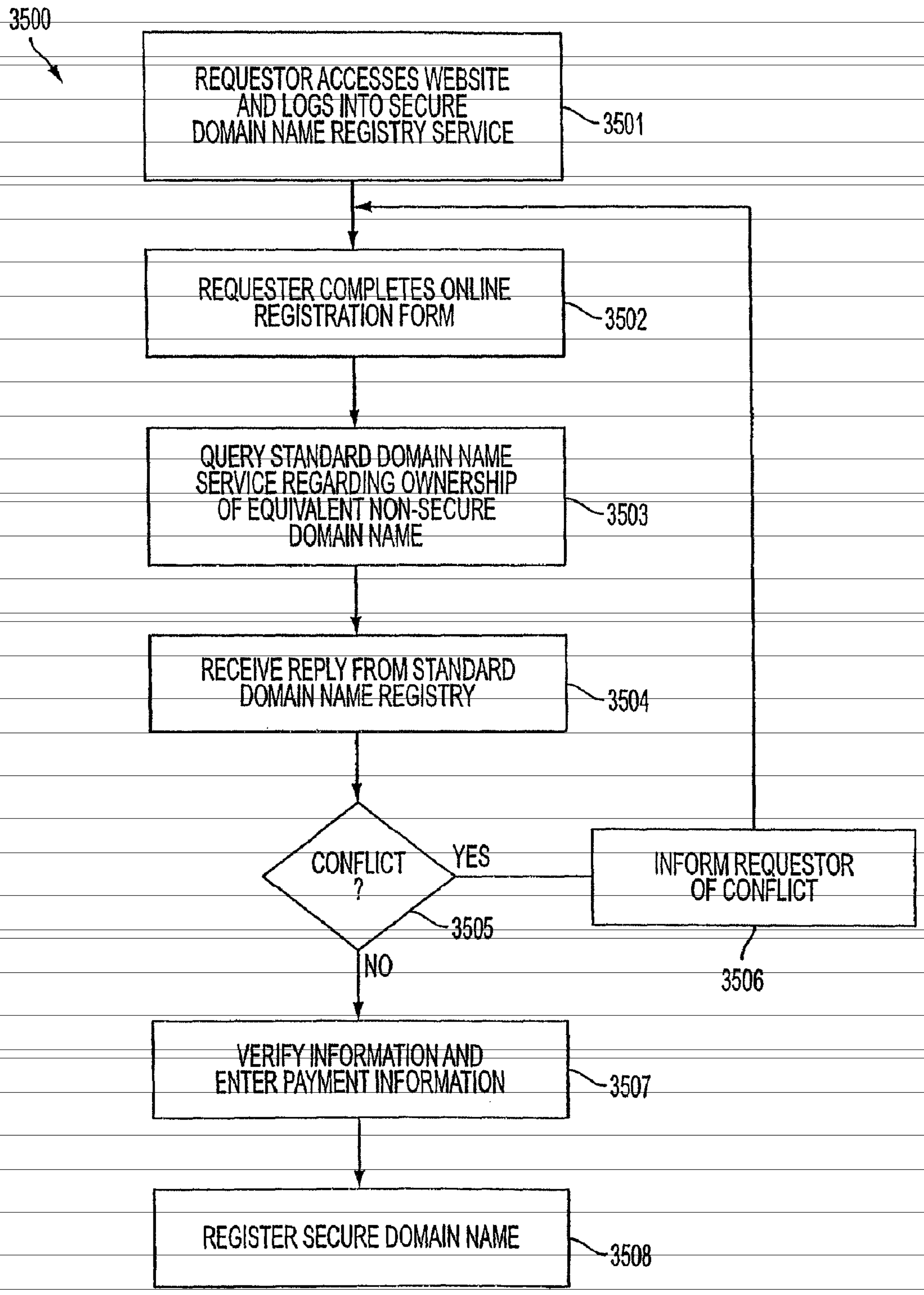

3500
REQUESTOR ACCESSES WEBSITE AND LOGS INTO SECURE DOMAIN NAME REGISTRY SERVICE
3501
REQUESTER COMPLETES ONLINE REGISTRATION FORM
3502
QUERY STANDARD DOMAIN NAME SERVICE REGARDING OWNERSHIP OF EQUIVALENT NON-SECURE DOMAIN NAME
3503
RECEIVE REPLY FROM STANDARD DOMAIN NAME REGISTRY
3504
CONFLICT ?
YES
3505
INFORM REQUESTOR OF CONFLICT
3506
NO
VERIFY INFORMATION AND ENTER PAYMENT INFORMATION
3507
REGISTER SECURE DOMAIN NAME
3508

FIG. 35

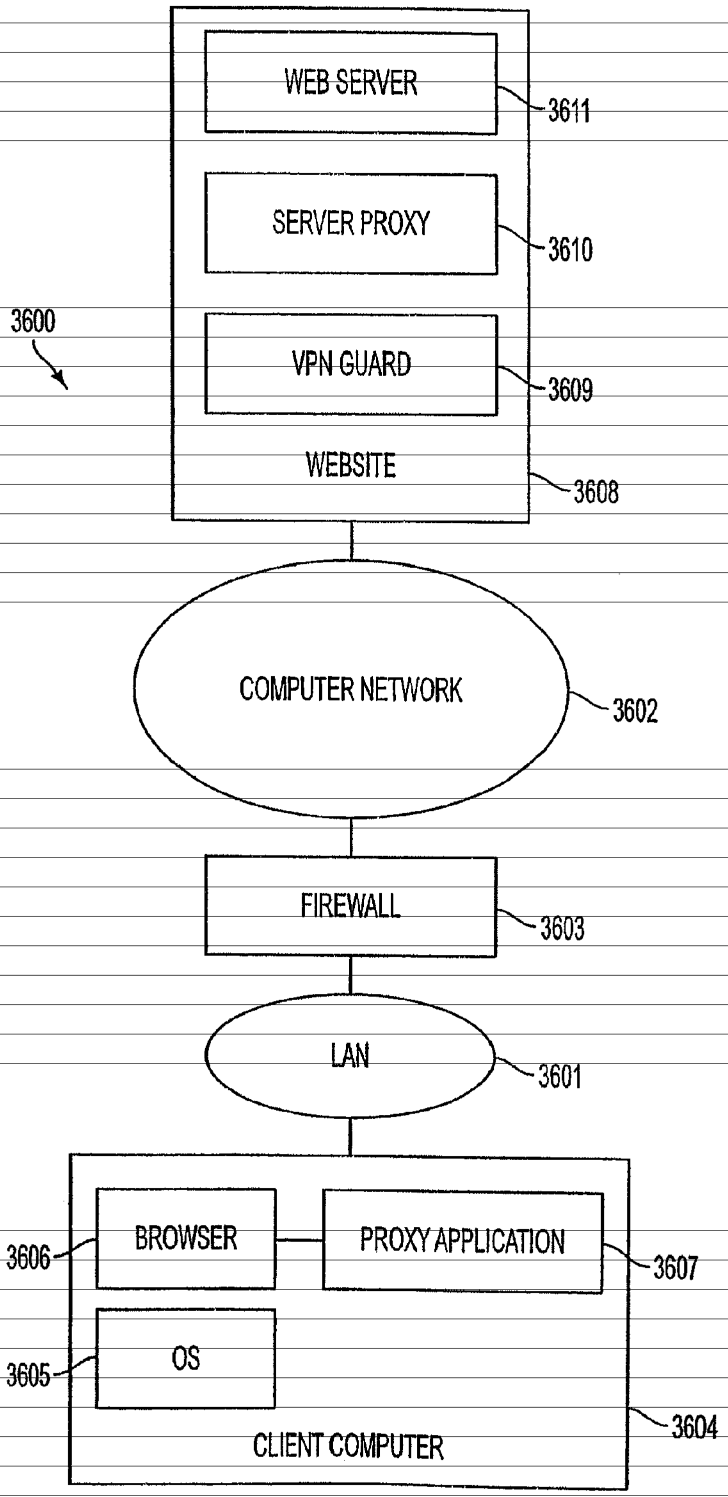

WEB SERVER
3611
SERVER PROXY
3610
3600
VPN GUARD
3609
WEBSITE
3608
COMPUTER NETWORK
3602
FIREWALL
3603
LAN
3601
3606
BROWSER
PROXY APPLICATION
3607
3605
OS
CLIENT COMPUTER
3604

FIG. 36

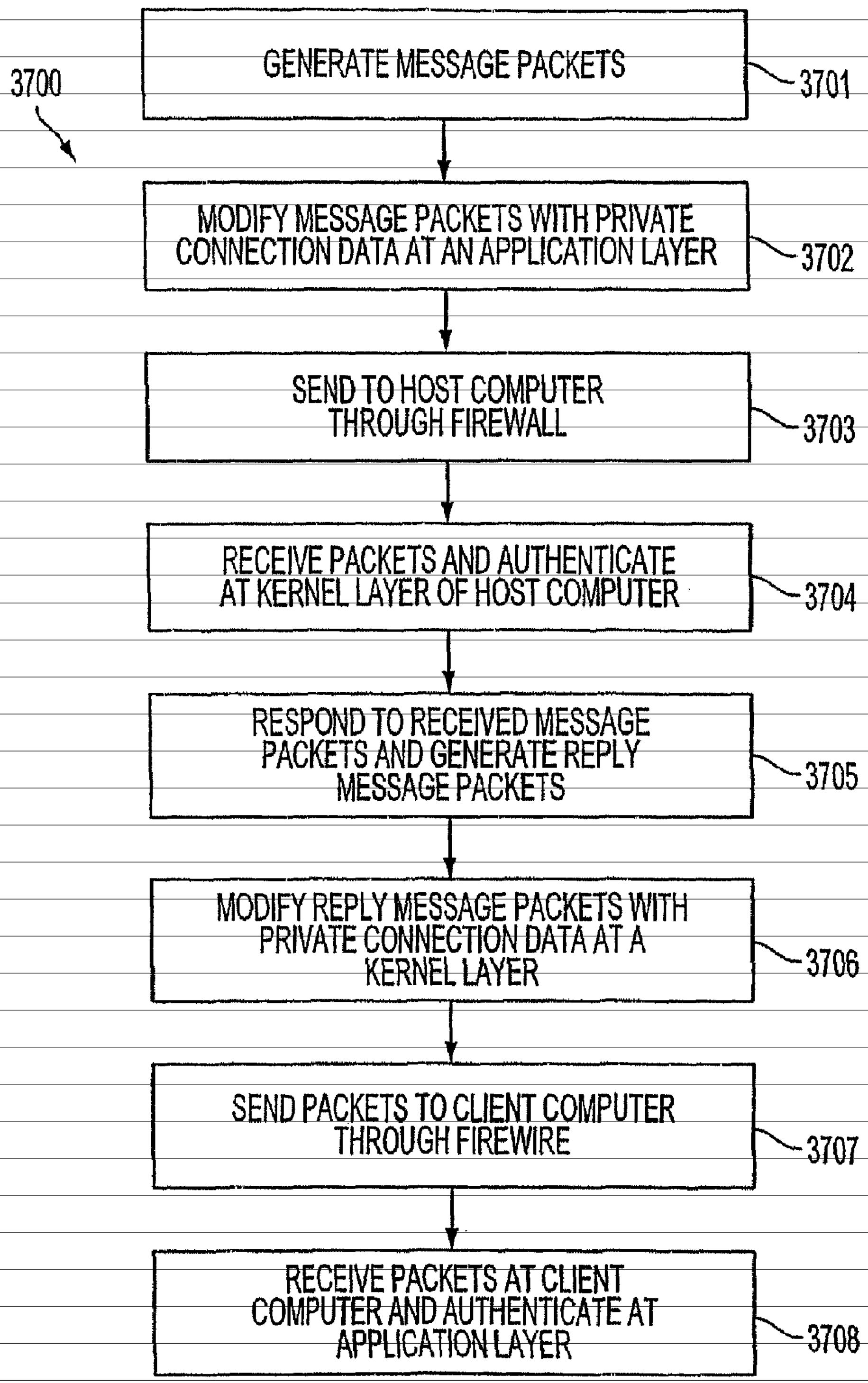

3700
GENERATE MESSAGE PACKETS
3701
MODIFY MESSAGE PACKETS WITH PRIVATE CONNECTION DATA AT AN APPLICATION LAYER
3702
SEND TO HOST COMPUTER THROUGH FIREWALL
3703
RECEIVE PACKETS AND AUTHENTICATE AT KERNEL LAYER OF HOST COMPUTER
3704
RESPOND TO RECEIVED MESSAGE PACKETS AND GENERATE REPLY MESSAGE PACKETS
3705
MODIFY REPLY MESSAGE PACKETS WITH PRIVATE CONNECTION DATA AT A KERNEL LAYER
3706
SEND PACKETS TO CLIENT COMPUTER THROUGH FIREWIRE
3707
RECEIVE PACKETS AT CLIENT COMPUTER AND AUTHENTICATE AT APPLICATION LAYER
3708

FIG. 37

AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority from and is a continuation of a U.S. application Ser. No. 10/714,849, filed Nov. 18, 2003, now U.S. Pat. No. 7,418,504, which is a continuation of U.S. application Ser. No. 09/558,210, filed Apr. 26, 2000, now abandoned, which in turn is a continuation-in-part of previously-filed U.S. application Ser. No. 09/504,783, filed on Feb. 15, 2000, now U.S. Pat. No. 6,502,135, issued Dec. 31, 2002, which in turn claims priority from and is a continuation-in-part patent application of previously-filed U.S. application Ser. No. 09/429,643, filed on Oct. 29, 1999, now U.S. Pat. No. 7,010,604, issued Mar. 07, 2006. The subject matter of U.S. application Ser. No. 09/429,643, now U.S. Pat. No. 7,010,604, which is bodily incorporated herein, derives from provisional U.S. application Nos. 60/106,261 (filed Oct. 30, 1998) and 60/137,704 (filed Jun. 7, 1999). The present application is also related to U.S. application Ser. No. 09/558,209, filed Apr. 26, 2000, now abandoned, and which is incorporated by reference herein.

BACKGROUND OF THE INVENTION

A tremendous variety of methods have been proposed and implemented to provide security and anonymity for communications over the Internet. The variety stems, in part, from the different needs of different Internet users. A basic heuristic framework to aid in discussing these different security techniques is illustrated in FIG. 1. Two terminals, an originating terminal **100** and a destination terminal **110** are in communication over the Internet. It is desired for the communications to be secure, that is, immune to eavesdropping. For example, terminal **100** may transmit secret information to terminal **110** over the Internet **107**. Also, it may be desired to prevent an eavesdropper from discovering that terminal **100** is in communication with terminal **110**. For example, if terminal **100** is a user and terminal **110** hosts a web site, terminal **100**'s user may not want anyone in the intervening networks to know what web sites he is "visiting." Anonymity would thus be an issue, for example, for companies that want to keep their market research interests private and thus would prefer to prevent outsiders from knowing which web-sites or other Internet resources they are "visiting." These two security issues may be called data security and anonymity, respectively.

Data security is usually tackled using some form of data encryption. An encryption key **48** is known at both the originating and terminating terminals **100** and **110**. The keys may be private and public at the originating and destination terminals **100** and **110**, respectively or they may be symmetrical keys (the same key is used by both parties to encrypt and decrypt). Many encryption methods are known and usable in this context.

To hide traffic from a local administrator or ISP, a user can employ a local proxy server in communicating over an encrypted channel with an outside proxy such that the local administrator or ISP only sees the encrypted traffic. Proxy servers prevent destination servers from determining the identities of the originating clients. This system employs an intermediate server interposed between client and destination server. The destination server sees only the Internet Protocol (IP) address of the proxy server and not the originating client.

The target server only sees the address of the outside proxy. This scheme relies on a trusted outside proxy server. Also, proxy schemes are vulnerable to traffic analysis methods of determining identities of transmitters and receivers. Another important limitation of proxy servers is that the server knows the identities of both calling and called parties. In many instances, an originating terminal, such as terminal A, would prefer to keep its identity concealed from the proxy, for example, if the proxy server is provided by an Internet service provider (ISP).

To defeat traffic analysis, a scheme called Chaum's mixes employs a proxy server that transmits and receives fixed length messages, including dummy messages. Multiple originating terminals are connected through a mix (a server) to multiple target servers. It is difficult to tell which of the originating terminals are communicating to which of the connected target servers, and the dummy messages confuse eavesdroppers' efforts to detect communicating pairs by analyzing traffic. A drawback is that there is a risk that the mix server could be compromised. One way to deal with this risk is to spread the trust among multiple mixes. If one mix is compromised, the identities of the originating and target terminals may remain concealed. This strategy requires a number of alternative mixes so that the intermediate servers interposed between the originating and target terminals are not determinable except by compromising more than one mix. The strategy wraps the message with multiple layers of encrypted addresses. The first mix in a sequence can decrypt only the outer layer of the message to reveal the next destination mix in sequence. The second mix can decrypt the message to reveal the next mix and so on. The target server receives the message and, optionally, a multi-layer encrypted payload containing return information to send data back in the same fashion. The only way to defeat such a mix scheme is to collude among mixes. If the packets are all fixed-length and intermixed with dummy packets, there is no way to do any kind of traffic analysis.

Still another anonymity technique, called 'crowds,' protects the identity of the originating terminal from the intermediate proxies by providing that originating terminals belong to groups of proxies called crowds. The crowd proxies are interposed between originating and target terminals. Each proxy through which the message is sent is randomly chosen by an upstream proxy. Each intermediate proxy can send the message either to another randomly chosen proxy in the "crowd" or to the destination. Thus, even crowd members cannot determine if a preceding proxy is the originator of the message or if it was simply passed from another proxy.

ZKS (Zero-Knowledge Systems) Anonymous IP Protocol allows users to select up to any of five different pseudonyms, while desktop software encrypts outgoing traffic and wraps it in User Datagram Protocol (UDP) packets. The first server in a 2+-hop system gets the UDP packets, strips off one layer of encryption to add another, then sends the traffic to the next server, which strips off yet another layer of encryption and adds a new one. The user is permitted to control the number of hops. At the final server, traffic is decrypted with an untraceable IP address. The technique is called onion-routing. This method can be defeated using traffic analysis. For a simple example, bursts of packets from a user during low-duty periods can reveal the identities of sender and receiver.

Firewalls attempt to protect LANs from unauthorized access and hostile exploitation or damage to computers connected to the LAN. Firewalls provide a server through which all access to the LAN must pass. Firewalls are centralized systems that require administrative overhead to maintain. They can be compromised by virtual-machine applications ("applets"). They instill a false sense of security that leads to security breaches for example by users sending sensitive information to servers outside the firewall or encouraging use of modems to sidestep the firewall security. Firewalls are not useful for distributed systems such as business travelers, extranets, small teams, etc.

SUMMARY OF THE INVENTION

A secure mechanism for communicating over the internet, including a protocol referred to as the Tunneled Agile Routing Protocol (TARP), uses a unique two-layer encryption format and special TARP routers. TARP routers are similar in function to regular IP routers. Each TARP router has one or more IP addresses and uses normal IP protocol to send IP packet messages ("packets" or "datagrams"). The IP packets exchanged between TARP terminals via TARP routers are actually encrypted packets whose true destination address is concealed except to TARP routers and servers. The normal or "clear" or "outside" IP header attached to TARP IP packets contains only the address of a next hop router or destination server. That is, instead of indicating a final destination in the destination field of the IP header, the TARP packet's IP header always points to a next-hop in a series of TARP router hops, or to the final destination. This means there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet since the destination could always be next-hop TARP router as well as the final destination.

Each TARP packet's true destination is concealed behind a layer of encryption generated using a link key. The link key is the encryption key used for encrypted communication between the hops intervening between an originating TARP terminal and a destination TARP terminal. Each TARP router can remove the outer layer of encryption to reveal the destination router for each TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal by the sender/receiver IP numbers in the cleartext IP header.

Once the outer layer of encryption is removed, the TARP router determines the final destination. Each TARP packet 140 undergoes a minimum number of hops to help foil traffic analysis. The hops may be chosen at random or by a fixed value. As a result, each TARP packet may make random trips among a number of geographically disparate routers before reaching its destination. Each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined. This feature is called agile routing. The fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. The associated advantages have to do with the inner layer of encryption discussed below. Agile routing is combined with another feature that furthers this purpose; a feature that ensures that any message is broken into multiple packets.

The IP address of a TARP router can be changed, a feature called IP agility. Each TARP router, independently or under direction from another TARP terminal or router, can change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs.

The message payload is hidden behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the intervening TARP routers. The session key is used to decrypt the payloads of the TARP packets permitting the data stream to be reconstructed.

Communication may be made private using link and session keys, which in turn may be shared and used according to any desired method. For example, public/private keys or symmetric keys may be used.

To transmit a data stream, a TARP originating terminal constructs a series of TARP packets from a series of IP packets generated by a network (IP) layer process. (Note that the terms "network layer," "data link layer," "application layer," etc. used in this specification correspond to the Open Systems Interconnection (OSI) network terminology.) The payloads of these packets are assembled into a block and chain-block encrypted using the session key. This assumes, of course, that all the IP packets are destined for the same TARP terminal. The block is then interleaved and the interleaved encrypted block is broken into a series of payloads, one for each TARP packet to be generated. Special TARP headers IP_T are then added to each payload using the IP headers from the data stream packets. The TARP headers can be identical to normal IP headers or customized in some way. They should contain a formula or data for deinterleaving the data at the destination TARP terminal, a time-to-live (TTL) parameter to indicate the number of hops still to be executed, a data type identifier which indicates whether the payload contains, for example, TCP or UDP data, the sender's TARP address, the destination TARP address, and an indicator as to whether the packet contains real or decoy data or a formula for filtering out decoy data if decoy data is spread in some way through the TARP payload data.

Note that although chain-block encryption is discussed here with reference to the session key, any encryption method may be used. Preferably, as in chain block encryption, a method should be used that makes unauthorized decryption difficult without an entire result of the encryption process. Thus, by separating the encrypted block among multiple packets and making it difficult for an interloper to obtain access to all of such packets, the contents of the communications are provided an extra layer of security.

Decoy or dummy data can be added to a stream to help foil traffic analysis by reducing the peak-to-average network load. It may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to a portion, or entirety, of a message, and that portion or entirety then interleaved into a number of separate packets. Considering the agile IP routing of the packets, and the attendant difficulty of reconstructing an entire sequence of packets to form a single block-encrypted message element, decoy packets can significantly increase the difficulty of reconstructing an entire data stream.

The above scheme may he implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Because the encryption system described above is insertable between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the Network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicating that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. In addition, it may create a subprocess that maintains the original IP address and continues interacting with the attacker in some manner.

Decoy packets may be generated by each TARP terminal on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis.

In various other embodiments of the invention, a scalable version of the system may be constructed in which a plurality of IP addresses are preassigned to each pair of communicating nodes in the network. Each pair of nodes agrees upon an algorithm for "hopping" between IP addresses (both sending and receiving), such that an eavesdropper sees apparently continuously random IP address pairs (source and destination) for packets transmitted between the pair. Overlapping or "reusable" IP addresses may be allocated to different users on the same subnet, since each node merely verifies that a particular packet includes a valid source/destination pair from the agreed-upon algorithm. Source/destination pairs are preferably not reused between any two nodes during any given end-to-end session, though limited IP block sizes or lengthy sessions might require it.

Further improvements described in this continuation-in-part application include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities.

The present invention provides key technologies for implementing a secure virtual Internet by using a new agile network protocol that is built on top of the existing Internet protocol (IP). The secure virtual Internet works over the existing Internet infrastructure, and interfaces with client applications the same way as the existing Internet. The key technologies provided by the present invention that support the secure virtual Internet include a "one-click" and "no-click" technique to become part of the secure virtual Internet, a secure domain name service (SDNS) for the secure virtual Internet, and a new approach for interfacing specific client applications onto the secure virtual Internet. According to the invention, the secure domain name service interfaces with existing applications, in addition to providing a way to register and serve domain names and addresses.

According to one aspect of the present invention, a user can conveniently establish a VPN using a "one-click" or a "no-click" technique without being required to enter user identification information, a password and/or an encryption key for establishing a VPN. The advantages of the present invention are provided by a method for establishing a secure communication link between a first computer and a second computer over a computer network, such as the Internet. In one embodiment, a secure communication mode is enabled at a first computer without a user entering any cryptographic information for establishing the secure communication mode of communication, preferably by merely selecting an icon displayed on the first computer. Alternatively, the secure communication mode of communication can be enabled by entering a command into the first computer. Then, a secure communication link is established between the first computer and a second computer over a computer network based on the enabled secure communication mode of communication. According to the invention, it is determined whether a secure communication software module is stored on the first computer in response to the step of enabling the secure communication mode of communication. A predetermined computer network address is then accessed for loading the secure communication software module when the software module is not stored on the first computer. Subsequently, the proxy software module is stored in the first computer. The secure communication link is a virtual private network communication link over the computer network. Preferably, the virtual private network can be based on inserting into each data packet one or more data values that vary according to a pseudo-random sequence. Alternatively, the virtual private network can be based on a computer network address hopping regime that is used to pseudorandomly change computer network addresses or other data values in packets transmitted between the first computer and the second computer, such that the second computer compares the data values in each data packet transmitted between the first computer and the second computer to a moving window of valid values. Yet another alternative provides that the virtual private network can be based on a comparison between a discriminator field in each data packet to a table of valid discriminator fields maintained for the first computer.

According to another aspect of the invention, a command is entered to define a setup parameter associated with the secure communication link mode of communication. Consequently, the secure communication mode is automatically established when a communication link is established over the computer network.

The present invention also provides a computer system having a communication link to a computer network, and a display showing a hyperlink for establishing a virtual private network through the computer network. When the hyperlink for establishing the virtual private network is selected, a virtual private network is established over the computer network. A non-standard top-level domain name is then sent over the virtual private network communication to a predetermined computer network address, such as a computer network address for a secure domain name service (SDNS).

The present invention provides a domain name service that provides secure computer network addresses for secure, non-standard top-level domain names. The advantages of the present invention are provided by a secure domain name service for a computer network that includes a portal connected to a computer network, such as the Internet, and a domain name database connected to the computer network through the portal. According to the invention, the portal authenticates a query for a secure computer network address, and the domain name database stores secure computer network addresses for the computer network. Each secure computer network address is based on a non-standard top-level domain name, such as .scom, .sorg, .snet, .snet, .sedu, .smil and .sint.

The present invention provides a way to encapsulate existing application network traffic at the application layer of a client computer so that the client application can securely communicate with a server protected by an agile network protocol. The advantages of the present invention are provided by a method for communicating using a private communication link between a client computer and a server computer over a computer network, such as the Internet. According to the invention, an information packet is sent from the client computer to the server computer over the computer network. The information packet contains data that is inserted into the payload portion of the packet at the application layer of the client computer and is used for forming a virtual private connection between the client computer and the server computer. The modified information packet can be sent through a firewall before being sent over the computer network to the server computer and by working on top of existing protocols (i.e., UDP, ICMP and TCP), the present invention more easily penetrates the firewall. The information packet is received at a kernel layer of an operating system on the server side. It is then determined at the kernel layer of the operating system on the host computer whether the information packet contains the data that is used for forming the virtual private connection. The server side replies by sending an information packet to the client computer that has been modified at the kernel layer to containing virtual private connection information in the payload portion of the reply information packet. Preferably, the information packet from the client computer and the reply information packet from the server side are each a UDP protocol information packet.

Alternative, both information packets could be a TCP/IP protocol information packet, or an ICMP protocol information packet.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is an illustration of secure communications over the Internet according to a prior art embodiment.

FIG. **2** is an illustration of secure communications over the Internet according to a an embodiment of the invention.

FIG. **3***a* is an illustration of a process of forming a tunneled IP packet according to an embodiment of the invention.

FIG. **3***b* is an illustration of a process of forming a tunneled IP packet according to another embodiment of the invention.

FIG. **4** is an illustration of an OSI layer location of processes that may be used to implement the invention.

FIG. **5** is a flow chart illustrating a process for routing a tunneled packet according to an embodiment of the invention.

FIG. **6** is a flow chart illustrating a process for forming a tunneled packet according to an embodiment of the invention.

FIG. **7** is a flow chart illustrating a process for receiving a tunneled packet according to an embodiment of the invention.

FIG. **8** shows how a secure session is established and synchronized between a client and a TARP router.

FIG. **9** shows an IP address hopping scheme between a client computer and TARP router using transmit and receive tables in each computer.

FIG. **10** shows physical link redundancy among three Internet Service Providers (ISPs) and a client computer.

FIG. **11** shows how multiple IP packets can be embedded into a single "frame" such as an Ethernet frame, and further shows the use of a discriminator field to camouflage true packet recipients.

FIG. **12**A shows a system that employs hopped hardware addresses, hopped IP addresses, and hopped discriminator fields.

FIG. **12**B shows several different approaches for hopping hardware addresses, IP addresses, and discriminator fields in combination.

FIG. **13** shows a technique for automatically re-establishing synchronization between sender and receiver through the use of a partially public sync value.

FIG. **14** shows a "checkpoint" scheme for regaining synchronization between a sender and recipient.

FIG. **15** shows further details of the checkpoint scheme of FIG. **14**.

FIG. **16** shows how two addresses can be decomposed into a plurality of segments for comparison with presence vectors.

FIG. **17** shows a storage array for a receiver's active addresses.

FIG. **18** shows the receiver's storage array after receiving a sync request.

FIG. **19** shows the receiver's storage array after new addresses have been generated.

FIG. **20** shows a system employing distributed transmission paths.

FIG. **21** shows a plurality of link transmission tables that can be used to route packets in the system of FIG. **20**.

FIG. **22**A shows a flowchart for adjusting weight value distributions associated with a plurality of transmission links.

FIG. **22**B shows a flowchart for setting a weight value to zero if a transmitter turns off.

FIG. **23** shows a system employing distributed transmission paths with adjusted weight value distributions for each path.

FIG. **24** shows an example using the system of FIG. **23**.

FIG. **25** shows a conventional domain-name look-up service.

FIG. **26** shows a system employing a DNS proxy server with transparent VPN creation.

FIG. **27** shows steps that can be carried out to implement transparent VPN creation based on a DNS look-up function.

FIG. **28** shows a system including a link guard function that prevents packet overloading on a low-bandwidth link LOW BW.

FIG. **29** shows one embodiment of a system employing the principles of FIG. **28**.

FIG. **30** shows a system that regulates packet transmission rates by throttling the rate at which synchronizations are performed.

FIG. **31** shows a signaling server **3101** and a transport server **3102** used to establish a VPN with a client computer.

FIG. **32** shows message flows relating to synchronization protocols of FIG. **31**.

FIG. **33** shows a system block diagram of a computer network in which the "one-click" secure communication link of the present invention is suitable for use.

FIG. **34** shows a flow diagram for installing and establishing a "one-click" secure communication link over a computer network according to the present invention.

FIG. **35** shows a flow diagram for registering a secure domain name according to the present invention.

FIG. **36** shows a system block diagram of a computer network in which a private connection according to the present invention can be configured to more easily traverse a firewall between two computer networks.

FIG. **37** shows a flow diagram for establishing a virtual private connection that is encapsulated using an existing network protocol.

DETAILED DESCRIPTION OF THE INVENTION

Referring to FIG. **2**, a secure mechanism for communicating over the internet employs a number of special routers or servers, called TARP routers **122-127** that are similar to regular IP routers **128-132** in that each has one or more IP addresses and uses normal IP protocol to send normal-looking IP packet messages, called TARP packets **140**. TARP packets **140** are identical to normal IP packet messages that are routed by regular IP routers **128-132** because each TARP packet **140** contains a destination address as in a normal IP packet. However, instead of indicating a final destination in the destination field of the IP header, the TARP packet's **140** IP header always points to a next-hop in a series of TARP router hops, or the final destination, TARP terminal **110**. Because the header of the TARP packet contains only the next-hop destination, there is no overt indication from an intercepted TARP packet of the true destination of the TARP packet **140** since the destination could always be the next-hop TARP router as well as the final destination, TARP terminal **110**.

Each TARP packet's true destination is concealed behind an outer layer of encryption generated using a link key **146**. The link key **146** is the encryption key used for encrypted communication between the end points (TARP terminals or TARP routers) of a single link in the chain of hops connecting the originating TARP terminal **100** and the destination TARP terminal **110**. Each TARP router **122-127**, using the link key **146** it uses to communicate with the previous hop in a chain, can use the link key to reveal the true destination of a TARP packet. To identify the link key needed to decrypt the outer layer of encryption of a TARP packet, a receiving TARP or routing terminal may identify the transmitting terminal

(which may indicate the link key used) by the sender field of the clear IP header. Alternatively, this identity may be hidden behind another layer of encryption in available bits in the clear IP header. Each TARP router, upon receiving a TARP message, determines if the message is a TARP message by using authentication data in the TARP packet. This could be recorded in available bytes in the TARP packet's IP header. Alternatively, TARP packets could be authenticated by attempting to decrypt using the link key **146** and determining if the results are as expected. The former may have computational advantages because it does not involve a decryption process.

Once the outer layer of decryption is completed by a TARP router **122-127**, the TARP router determines the final destination. The system is preferably designed to cause each TARP packet **140** to undergo a minimum number of hops to help foil traffic analysis. The time to live counter in the IP header of the TARP message may be used to indicate a number of TARP router hops yet to be completed. Each TARP router then would decrement the counter and determine from that whether it should forward the TARP packet **140** to another TARP router **122-127** or to the destination TARP terminal **110**. If the time to live counter is zero or below zero after decrementing, for an example of usage, the TARP router receiving the TARP packet **140** may forward the TARP packet **140** to the destination TARP terminal **110**. If the time to live counter is above zero after decrementing, for an example of usage, the TARP router receiving the TARP packet **140** may forward the TARP packet **140** to a TARP router **122-127** that the current TARP terminal chooses at random. As a result, each TARP packet **140** is routed through some minimum number of hops of TARP routers **122-127** which are chosen at random.

Thus, each TARP packet, irrespective of the traditional factors determining traffic in the Internet, makes random trips among a number of geographically disparate routers before reaching its destination and each trip is highly likely to be different for each packet composing a given message because each trip is independently randomly determined as described above. This feature is called agile routing. For reasons that will become clear shortly, the fact that different packets take different routes provides distinct advantages by making it difficult for an interloper to obtain all the packets forming an entire multi-packet message. Agile routing is combined with another feature that furthers this purpose, a feature that ensures that any message is broken into multiple packets.

A TARP router receives a TARP packet when an IP address used by the TARP router coincides with the IP address in the TARP packet's IP header IPc. The IP address of a TARP router, however, may not remain constant. To avoid and manage attacks, each TARP router, independently or under direction from another TARP terminal or router, may change its IP address. A separate, unchangeable identifier or address is also defined. This address, called the TARP address, is known only to TARP routers and terminals and may be correlated at any time by a TARP router or a TARP terminal using a Lookup Table (LUT). When a TARP router or terminal changes its IP address, it updates the other TARP routers and terminals which in turn update their respective LUTs. In reality, whenever a TARP router looks up the address of a destination in the encrypted header, it must convert a TARP address to a real IP address using its LUT.

While every TARP router receiving a TARP packet has the ability to determine the packet's final destination, the message payload is embedded behind an inner layer of encryption in the TARP packet that can only be unlocked using a session key. The session key is not available to any of the TARP routers **122-127** intervening between the originating **100** and destination **110** TARP terminals. The session key is used to decrypt the payloads of the TARP packets **140** permitting an entire message to be reconstructed.

In one embodiment, communication may be made private using link and session keys, which in turn may be shared and used according any desired method. For example, a public key or symmetric keys may be communicated between link or session endpoints using a public key method. Any of a variety of other mechanisms for securing data to ensure that only authorized computers can have access to the private information in the TARP packets **140** may be used as desired.

Referring to FIG. **3***a*, to construct a series of TARP packets, a data stream **300** of IP packets **207***a*, **207***b*, **207***c*, etc., such series of packets being formed by a network (IP) layer process, is broken into a series of small sized segments. In the present example, equal-sized segments **1-9** are defined and used to construct a set of interleaved data packets A, B, and C. Here it is assumed that the number of interleaved packets A, B, and C formed is three and that the number of IP packets **207***a*-**207***c* used to form the three interleaved packets A, B, and C is exactly three. Of course, the number of IP packets spread over a group of interleaved packets may be any convenient number as may be the number of interleaved packets over which the incoming data stream is spread. The latter, the number of interleaved packets over which the data stream is spread, is called the interleave window.

To create a packet, the transmitting software interleaves the normal IP packets **207***a* et. seq, to form a new set of interleaved payload data **320**. This payload data **320** is then encrypted using a session key to form a set of session-key-encrypted payload data **330**, each of which, A, B, and C, will form the payload of a TARP packet. Using the IP header data, from the original packets **207***a*-**207***c*, new TARP headers IPT are formed. The TARP headers IPT can be identical to normal IP headers or customized in some way. In a preferred embodiment, the TARP headers IPT are IP headers with added data providing the following information required for routing and reconstruction of messages, some of which data is ordinarily, or capable of being, contained in normal IP headers:

1. A window sequence number—an identifier that indicates where the packet belongs in the original message sequence.
2. An interleave sequence number—an identifier that indicates the interleaving sequence used to form the packet so that the packet can be deinterleaved along with other packets in the interleave window.
3. A time-to-live (TTL) datum—indicates the number of TARP-router-hops to be executed before the packet reaches its destination. Note that the TTL parameter may provide a datum to be used in a probabilistic formula for determining whether to route the packet to the destination or to another hop.
4. Data type identifier—indicates whether the payload contains, for example, TCP or UDP data.
5. Sender's address—indicates the sender's address in the TARP network.
6. Destination address—indicates the destination terminal's address in the TARP network.
7. Decoy/Real—an indicator of whether the packet contains real message data or dummy decoy data or a combination.

Obviously, the packets going into a single interleave window must include only packets with a common destination. Thus, it is assumed in the depicted example that the IP headers of IP packets **207***a*-**207***c* all contain the same destination address or at least will be received by the same terminal so

that they can be deinterleaved. Note that dummy or decoy data or packets can be added to form a larger interleave window than would otherwise be required by the size of a given message. Decoy or dummy data can be added to a stream to help foil traffic analysis by leveling the load on the network. Thus, it may be desirable to provide the TARP process with an ability to respond to the time of day or other criteria to generate more decoy data during low traffic periods so that communication bursts at one point in the Internet cannot be tied to communication bursts at another point to reveal the communicating endpoints.

Dummy data also helps to break the data into a larger number of inconspicuously-sized packets permitting the interleave window size to be increased while maintaining a reasonable size for each packet. (The packet size can be a single standard size or selected from a fixed range of sizes.) One primary reason for desiring for each message to be broken into multiple packets is apparent if a chain block encryption scheme is used to form the first encryption layer prior to interleaving. A single block encryption may be applied to a portion, or the entirety, of a message, and that portion or entirety then interleaved into a number of separate packets.

Referring to FIG. **3***b*, in an alternative mode of TARP packet construction, a series of IP packets are accumulated to make up a predefined interleave window. The payloads of the packets are used to construct a single block **520** for chain block encryption using the session key. The payloads used to form the block are presumed to be destined for the same terminal. The block size may coincide with the interleave window as depicted in the example embodiment of FIG. **3***b*. After encryption, the encrypted block is broken into separate payloads and segments which are interleaved as in the embodiment of FIG. **3***a*. The resulting interleaved packets A, B, and C, are then packaged as TARP packets with TARP headers as in the Example of FIG. **3***a*. The remaining process is as shown in, and discussed with reference to, FIG. **3***a*.

Once the TARP packets **340** are formed, each entire TARP packet **340**, including the TARP header IPT, is encrypted using the link key for communication with the first-hop-TARP router. The first hop TARP router is randomly chosen. A final unencrypted IP header IPc is added to each encrypted TARP packet **340** to form a normal IP packet **360** that can be transmitted to a TARP router. Note that the process of constructing the TARP packet **360** does not have to be done in stages as described. The above description is just a useful heuristic for describing the final product, namely, the TARP packet.

Note that, TARP header IP_T could be a completely custom header configuration with no similarity to a normal IP header except that it contain the information identified above. This is so since this header is interpreted by only TARP routers.

The above scheme may be implemented entirely by processes operating between the data link layer and the network layer of each server or terminal participating in the TARP system. Referring to FIG. **4**, a TARP transceiver **405** can be an originating terminal **100**, a destination terminal **110**, or a TARP router **122-127**. In each TARP Transceiver **405**, a transmitting process is generated to receive normal packets from the Network (IP) layer and generate TARP packets for communication over the network. A receiving process is generated to receive normal IP packets containing TARP packets and generate from these normal IP packets which are "passed up" to the Network (IP) layer. Note that where the TARP Transceiver **405** is a router, the received TARP packets **140** are not processed into a stream of IP packets **415** because they need only be authenticated as proper TARP packets and then passed to another TARP router or a TARP destination terminal **110**. The intervening process, a "TARP Layer" **420**, could be combined with either the data link layer **430** or the Network layer **410**. In either case, it would intervene between the data link layer **430** so that the process would receive regular IP packets containing embedded TARP packets and "hand up" a series of reassembled IP packets to the Network layer **410**. As an example of combining the TARP layer **420** with the data link layer **430**, a program may augment the normal processes running a communications card, for example, an Ethernet card. Alternatively, the TARP layer processes may form part of a dynamically loadable module that is loaded and executed to support communications between the network and data link layers.

Because the encryption system described above can be inserted between the data link and network layers, the processes involved in supporting the encrypted communication may be completely transparent to processes at the IP (network) layer and above. The TARP processes may also be completely transparent to the data link layer processes as well. Thus, no operations at or above the network layer, or at or below the data link layer, are affected by the insertion of the TARP stack. This provides additional security to all processes at or above the network layer, since the difficulty of unauthorized penetration of the network layer (by, for example, a hacker) is increased substantially. Even newly developed servers running at the session layer leave all processes below the session layer vulnerable to attack. Note that in this architecture, security is distributed. That is, notebook computers used by executives on the road, for example, can communicate over the Internet without any compromise in security.

Note that IP address changes made by TARP terminals and routers can be done at regular intervals, at random intervals, or upon detection of "attacks." The variation of IP addresses hinders traffic analysis that might reveal which computers are communicating, and also provides a degree of immunity from attack. The level of immunity from attack is roughly proportional to the rate at which the IP address of the host is changing.

As mentioned, IP addresses may be changed in response to attacks. An attack may be revealed, for example, by a regular series of messages indicates that a router is being probed in some way. Upon detection of an attack, the TARP layer process may respond to this event by changing its IP address. To accomplish this, the TARP process will construct a TARP-formatted message, in the style of Internet Control Message Protocol (ICMP) datagrams as an example; this message will contain the machine's TARP address, its previous IP address, and its new IP address. The TARP layer will transmit this packet to at least one known TARP router; then upon receipt and validation of the message, the TARP router will update its LUT with the new IP address for the stated TARP address. The TARP router will then format a similar message, and broadcast it to the other TARP routers so that they may update their LUTs. Since the total number of TARP routers on any given subnet is expected to be relatively small, this process of updating the LUTs should be relatively fast. It may not, however, work as well when there is a relatively large number of TARP routers and/or a relatively large number of clients; this has motivated a refinement of this architecture to provide scalability; this refinement has led to a second embodiment, which is discussed below.

Upon detection of an attack, the TARP process may also create a subprocess that maintains the original IP address and continues interacting with the attacker. The latter may provide an opportunity to trace the attacker or study the attacker's methods (called "fishbowling" drawing upon the analogy of a small fish in a fish bowl that "thinks" it is in the ocean but is actually under captive observation). A history of the communication between the attacker and the abandoned (fish-bowled) IP address can be recorded or transmitted for human analysis or further synthesized for purposes of responding in some way.

As mentioned above, decoy or dummy data or packets can be added to outgoing data streams by TARP terminals or routers. In addition to making it convenient to spread data over a larger number of separate packets, such decoy packets can also help to level the load on inactive portions of the Internet to help foil traffic analysis efforts.

Decoy packets may be generated by each TARP terminal **100**, **110** or each router **122-127** on some basis determined by an algorithm. For example, the algorithm may be a random one which calls for the generation of a packet on a random basis when the terminal is idle. Alternatively, the algorithm may be responsive to time of day or detection of low traffic to generate more decoy packets during low traffic times. Note that packets are preferably generated in groups, rather than one by one, the groups being sized to simulate real messages. In addition, so that decoy packets may be inserted in normal TARP message streams, the background loop may have a latch that makes it more likely to insert decoy packets when a message stream is being received. That is, when a series of messages are received, the decoy packet generation rate may be increased. Alternatively, if a large number of decoy packets is received along with regular TARP packets, the algorithm may increase the rate of dropping of decoy packets rather than forwarding them. The result of dropping and generating decoy packets in this way is to make the apparent incoming message size different from the apparent outgoing message size to help foil traffic analysis. The rate of reception of packets, decoy or otherwise, may be indicated to the decoy packet dropping and generating processes through perishable decoy and regular packet counters. (A perishable counter is one that resets or decrements its value in response to time so that it contains a high value when it is incremented in rapid succession and a small value when incremented either slowly or a small number of times in rapid succession.) Note that destination TARP terminal **110** may generate decoy packets equal in number and size to those TARP packets received to make it appear it is merely routing packets and is therefore not the destination terminal.

Referring to FIG. **5**, the following particular steps may be employed in the above-described method for routing TARP packets.

S0. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.

S2. The TARP packet may be probed in some way to authenticate the packet before attempting to decrypt it using the link key. That is, the router may determine that the packet is an authentic TARP packet by performing a selected operation on some data included with the clear IP header attached to the encrypted TARP packet contained in the payload. This makes it possible to avoid performing decryption on packets that are not authentic TARP packets.

S3. The TARP packet is decrypted to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.

S4. If the packet is a decoy packet, the perishable decoy counter is incremented.

S5. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the router may choose to throw it away. If the received packet is a decoy packet and it is determined that it should be thrown away (S6), control returns to step S0.

S7. The TTL parameter of the TARP header is decremented and it is determined if the TTL parameter is greater than zero.

S8. If the TTL parameter is greater than zero, a TARP address is randomly chosen from a list of TARP addresses maintained by the router and the link key and IP address corresponding to that TARP address memorized for use in creating a new IP packet containing the TARP packet.

S9. If the TTL parameter is zero or less, the link key and IP address corresponding to the TARP address of the destination are memorized for use in creating the new IP packet containing the TARP packet.

S10. The TARP packet is encrypted using the memorized link key.

S11. An IP header is added to the packet that contains the stored IP address, the encrypted TARP packet wrapped with an IP header, and the completed packet transmitted to the next hop or destination.

Referring to FIG. 6, the following particular steps may be employed in the above-described method for generating TARP packets.

S20. A background loop operation applies an algorithm that determines the generation of decoy IP packets. The loop is interrupted when a data stream containing IP packets is received for transmission.

S21. The received IP packets are grouped into a set consisting of messages with a constant IP destination address. The set is further broken down to coincide with a maximum size of an interleave window The set is encrypted, and interleaved into a set of payloads destined to become TARP packets.

S22. The TARP address corresponding to the IP address is determined from a lookup table and stored to generate the TARP header. An initial TTL count is generated and stored in the header. The TTL count may be random with minimum and maximum values or it may be fixed or determined by some other parameter.

S23. The window sequence numbers and interleave sequence numbers are recorded in the TARP headers of each packet.

S24. One TARP router address is randomly chosen for each TARP packet and the IP address corresponding to it stored for use in the clear IP header. The link key corresponding to this router is identified and used to encrypt TARP packets containing interleaved and encrypted data and TARP headers.

S25. A clear IP header with the first hop router's real IP address is generated and added to each of the encrypted TARP packets and the resulting packets.

Referring to FIG. 7, the following particular steps may be employed in the above-described method for receiving TARP packets.

S40. A background loop operation is performed which applies an algorithm which determines the generation of decoy IP packets. The loop is interrupted when an encrypted TARP packet is received.

S42. The TARP packet may be probed to authenticate the packet before attempting to decrypt it using the link key.

S43. The TARP packet is decrypted with the appropriate link key to expose the destination TARP address and an indication of whether the packet is a decoy packet or part of a real message.

S44. If the packet is a decoy packet, the perishable decoy counter is incremented.

S45. Based on the decoy generation/dropping algorithm and the perishable decoy counter value, if the packet is a decoy packet, the receiver may choose to throw it away.

S46. The TARP packets are cached until all packets forming an interleave window are received.

S47. Once all packets of an interleave window are received, the packets are deinterleaved.

S48. The packets block of combined packets defining the interleave window is then decrypted using the session key.

S49. The decrypted block is then divided using the window sequence data and the IP_T headers are converted into normal IP_C headers. The window sequence numbers are integrated in the IP_C headers.

S50. The packets are then handed up to the IP layer processes.

I. Scalability Enhancements

The IP agility feature described above relies on the ability to transmit IP address changes to all TARP routers. The embodiments including this feature will be referred to as "boutique" embodiments due to potential limitations in scaling these features up for a large network, such as the Internet. (The "boutique" embodiments would, however, be robust for use in smaller networks, such as small virtual private networks, for example). One problem with the boutique embodiments is that if IP address changes are to occur frequently, the message traffic required to update all routers sufficiently quickly creates a serious burden on the Internet when the TARP router and/or client population gets large. The bandwidth burden added to the networks, for example in ICMP packets, that would be used to update all the TARP routers could overwhelm the Internet for a large scale implementation that approached the scale of the Internet. In other words, the boutique system's scalability is limited.

A system can be constructed which trades some of the features of the above embodiments to provide the benefits of IP agility without the additional messaging burden. This is accomplished by IP address-hopping according to shared algorithms that govern IP addresses used between links participating in communications sessions between nodes such as TARP nodes. (Note that the IP hopping technique is also applicable to the boutique embodiment.) The IP agility feature discussed with respect to the boutique system can be modified so that it becomes decentralized under this scalable regime and governed by the above-described shared algorithm. Other features of the boutique system may be combined with this new type of IP-agility.

The new embodiment has the advantage of providing IP agility governed by a local algorithm and set of IP addresses exchanged by each communicating pair of nodes. This local governance is session-independent in that it may govern communications between a pair of nodes, irrespective of the session or end points being transferred between the directly communicating pair of nodes.

In the scalable embodiments, blocks of IP addresses are allocated to each node in the network. (This scalability will increase in the future, when Internet Protocol addresses are increased to 128-bit fields, vastly increasing the number of distinctly addressable nodes). Each node can thus use any of the IP addresses assigned to that node to communicate with other nodes in the network. Indeed, each pair of communicating nodes can use a plurality of source IP addresses and destination IP addresses for communicating with each other.

Each communicating pair of nodes in a chain participating in any session stores two blocks of IP addresses, called netblocks, and an algorithm and randomization seed for selecting, from each netblock, the next pair of source/destination IP addresses that will be used to transmit the next message. In other words, the algorithm governs the sequential selection of IP-address pairs, one sender and one receiver IP address, from each netblock. The combination of algorithm, seed, and netblock (IP address block) will be called a "hopblock." A router issues separate transmit and receive hopblocks to its clients. The send address and the receive address of the IP header of each outgoing packet sent by the client are filled with the send and receive IP addresses generated by the algorithm. The algorithm is "clocked" (indexed) by a counter so that each time a pair is used, the algorithm turns out a new transmit pair for the next packet to be sent.

The router's receive hopblock is identical to the client's transmit hopblock. The router uses the receive hopblock to predict what the send and receive IP address pair for the next expected packet from that client will be. Since packets can be received out of order, it is not possible for the router to predict with certainty what IP address pair will be on the next sequential packet. To account for this problem, the router generates a range of predictions encompassing the number of possible transmitted packet send/receive addresses, of which the next packet received could leap ahead. Thus, if there is a vanishingly small probability that a given packet will arrive at the router ahead of 5 packets transmitted by the client before the given packet, then the router can generate a series of 6 send/receive IP address pairs (or "hop window") to compare with the next received packet. When a packet is received, it is marked in the hop window as such, so that a second packet with the same IP address pair will be discarded. If an out-of-sequence packet does not arrive within a predetermined timeout period, it can be requested for retransmission or simply discarded from the receive table, depending upon the protocol in use for that communications session, or possibly by convention.

When the router receives the client's packet, it compares the send and receive IP addresses of the packet with the next N predicted send and receive IP address pairs and rejects the packet if it is not a member of this set. Received packets that do not have the predicted source/destination IP addresses falling with the window are rejected, thus thwarting possible hackers. (With the number of possible combinations, even a fairly large window would be hard to fall into at random.) If it is a member of this set, the router accepts the packet and processes it further. This link-based IP-hopping strategy, referred to as "IHOP," is a network element that stands on its own and is not necessarily accompanied by elements of the boutique system described above. If the routing agility feature described in connection with the boutique embodiment is combined with this link-based IP-hopping strategy, the router's next step would be to decrypt the TARP header to determine the destination TARP router for the packet and determine what should be the next hop for the packet. The TARP router would then forward the packet to a random TARP router or the destination TARP router with which the source TARP router has a link-based IP hopping communication established.

FIG. **8** shows how a client computer **801** and a TARP router **811** can establish a secure session. When client **801** seeks to establish an IHOP session with TARP router **811**, the client **801** sends "secure synchronization" request ("SSYN") packet **821** to the TARP router **811**. This SYN packet **821** contains the client's **801** authentication token, and may be sent to the router **811** in an encrypted format. The source and destination IP numbers on the packet **821** are the client's **801** current fixed IP address, and a "known" fixed IP address for the router **811**. (For security purposes, it may be desirable to reject any packets from outside of the local network that are destined for the router's known fixed IP address.) Upon receipt and validation of the client's **801** SSYN packet **821**, the router **811** responds by sending an encrypted "secure synchronization acknowledgment" ("SSYN ACK") **822** to the client **801**. This SSYN ACK **822** will contain the transmit and receive hopblocks that the client **801** will use when communicating with the TARP router **811**. The client **801** will acknowledge the TARP router's **811** response packet **822** by generating an encrypted SSYN ACK ACK packet **823** which will be sent from the client's **801** fixed IP address and to the TARP router's **811** known fixed IP address. The client **801** will simultaneously generate a SSYN ACK ACK packet; this SSYN ACK packet, referred to as the Secure Session Initiation (SSI) packet **824**, will be sent with the first {sender, receiver} IP pair in the client's transmit table **921** (FIG. **9**), as specified in the transmit hopblock provided by the TARP router **811** in the SSYN ACK packet **822**. The TARP router **811** will respond to the SSI packet **824** with an SSI ACK packet **825**, which will be sent with the first {sender, receiver} IP pair in the TARP router's transmit table **923**. Once these packets have been successfully exchanged, the secure communications session is established, and all further secure communications between the client **801** and the TARP router **811** will be conducted via this secure session, as long as synchronization is maintained. If synchronization is lost, then the client **801** and TARP router **802** may re-establish the secure session by the procedure outlined in FIG. **8** and described above.

While the secure session is active, both the client **901** and TARP router **911** (FIG. **9**) will maintain their respective transmit tables **921**, **923** and receive tables **922**, **924**, as provided by the TARP router during session synchronization **822**. It is important that the sequence of IP pairs in the client's transmit table **921** be identical to those in the TARP router's receive table **924**; similarly, the sequence of IP pairs in the client's receive table **922** must be identical to those in the router's transmit table **923**. This is required for the session synchronization to be maintained. The client **901** need maintain only one transmit table **921** and one receive table **922** during the course of the secure session. Each sequential packet sent by the client **901** will employ the next {send, receive} IP address pair in the transmit table, regardless of TCP or UDP session. The TARP router **911** will expect each packet arriving from the client **901** to bear the next IP address pair shown in its receive table.

Since packets can arrive out of order, however, the router **911** can maintain a "look ahead" buffer in its receive table, and will mark previously-received IP pairs as invalid for future packets; any future packet containing an IP pair that is in the look-ahead buffer but is marked as previously received will be discarded. Communications from the TARP router **911** to the client **901** are maintained in an identical manner; in particular, the router **911** will select the next IP address pair from its transmit table **923** when constructing a packet to send to the client **901**, and the client **901** will maintain a look-ahead buffer of expected IP pairs on packets that it is receiving. Each TARP router will maintain separate pairs of transmit and receive tables for each client that is currently engaged in a secure session with or through that TARP router.

While clients receive their hopblocks from the first server linking them to the Internet, routers exchange hopblocks. When a router establishes a link-based IP-hopping communication regime with another router, each router of the pair exchanges its transmit hopblock. The transmit hopblock of each router becomes the receive hopblock of the other router. The communication between routers is governed as described by the example of a client sending a packet to the first router.

While the above strategy works fine in the IP milieu, many local networks that are connected to the Internet are Ethernet systems. In Ethernet, the IP addresses of the destination devices must be translated into hardware addresses, and vice versa, using known processes ("address resolution protocol," and "reverse address resolution protocol"). However, if the link-based IP-hopping strategy is employed, the correlation process would become explosive and burdensome. An alternative to the link-based IP hopping strategy may be employed within an Ethernet network. The solution is to provide that the node linking the Internet to the Ethernet (call it the border node) use the link-based IP-hopping communication regime to communicate with nodes outside the Ethernet LAN. Within the Ethernet LAN, each TARP node would have a single IP address which would be addressed in the conventional way. Instead of comparing the {sender, receiver} IP address pairs to authenticate a packet, the intra-LAN TARP node would use one of the IP header extension fields to do so. Thus, the border node uses an algorithm shared by the intra-LAN TARP node to generate a symbol that is stored in the free field in the IP header, and the intra-LAN TARP node generates a range of symbols based on its prediction of the next expected packet to be received from that particular source IP address. The packet is rejected if it does not fall into the set of predicted symbols (for example, numerical values) or is accepted if it does. Communications from the intra-LAN TARP node to the border node are accomplished in the same manner, though the algorithm will necessarily be different for security reasons. Thus, each of the communicating nodes will generate transmit and receive tables in a similar manner to that of FIG. 9; the intra-LAN TARP nodes transmit table will be identical to the border node's receive table, and the intra-LAN TARP node's receive table will be identical to the border node's transmit table.

The algorithm used for IP address-hopping can be any desired algorithm. For example, the algorithm can be a given pseudo-random number generator that generates numbers of the range covering the allowed IP addresses with a given seed. Alternatively, the session participants can assume a certain type of algorithm and specify simply a parameter for applying the algorithm. For example the assumed algorithm could be a particular pseudo-random number generator and the session participants could simply exchange seed values.

Note that there is no permanent physical distinction between the originating and destination terminal nodes. Either device at either end point can initiate a synchronization of the pair. Note also that the authentication/synchronization-request (and acknowledgment) and hopblock-exchange may all be served by a single message so that separate message exchanges may not be required.

As another extension to the stated architecture, multiple physical paths can be used by a client, in order to provide link redundancy and further thwart attempts at denial of service and traffic monitoring. As shown in FIG. 10, for example, client **1001** can establish three simultaneous sessions with each of three TARP routers provided by different ISPs **1011**, **1012**, **1013**. As an example, the client **1001** can use three different telephone lines **1021**, **1022**, **1023** to connect to the ISPs, or two telephone lines and a cable modem, etc. In this scheme, transmitted packets will be sent in a random fashion among the different physical paths. This architecture provides a high degree of communications redundancy, with improved immunity from denial-of-service attacks and traffic monitoring.

2. Further Extensions

The following describes various extensions to the techniques, systems, and methods described above. As described above, the security of communications occurring between computers in a computer network (such as the Internet, an Ethernet, or others) can be enhanced by using seemingly random source and destination Internet Protocol (IP) addresses for data packets transmitted over the network. This feature prevents eavesdroppers from determining which computers in the network are communicating with each other while permitting the two communicating computers to easily recognize whether a given received data packet is legitimate or not. In one embodiment of the above-described systems, an IP header extension field is used to authenticate incoming packets on an Ethernet.

Various extensions to the previously described techniques described herein include: (1) use of hopped hardware or "MAC" addresses in broadcast type network; (2) a self synchronization technique that permits a computer to automatically regain synchronization with a sender; (3) synchronization algorithms that allow transmitting and receiving computers to quickly re-establish synchronization in the event of lost packets or other events; and (4) a fast-packet rejection mechanism for rejecting invalid packets. Any or all of these extensions can be combined with the features described above in any of various ways.

A. Hardware Address Hopping

Internet protocol-based communications techniques on a LAN—or across any dedicated physical medium—typically embed the IP packets within lower-level packets, often referred to as "frames." As shown in FIG. **11**, for example, a first Ethernet frame **1150** comprises a frame header **1101** and two embedded IP packets **IP1** and **IP2**, while a second Ethernet frame **1160** comprises a different frame header **1104** and a single IP packet **IP3**. Each frame header generally includes a source hardware address **1101** A and a destination hardware address **1101** B; other well-known fields in frame headers are omitted from FIG. **11** for clarity. Two hardware nodes communicating over a physical communication channel insert appropriate source and destination hardware addresses to indicate which nodes on the channel or network should receive the frame.

It may be possible for a nefarious listener to acquire information about the contents of a frame and/or its communicants by examining frames on a local network rather than (or in addition to) the IP packets themselves. This is especially true in broadcast media, such as Ethernet, where it is necessary to insert into the frame header the hardware address of the machine that generated the frame and the hardware address of the machine to which frame is being sent. All nodes on the network can potentially "see" all packets transmitted across the network. This can be a problem for secure communications, especially in cases where the communicants do not want for any third party to be able to identify who is engaging in the information exchange. One way to address this problem is to push the address-hopping scheme down to the hardware layer. In accordance with various embodiments of the invention, hardware addresses are "hopped" in a manner similar to that used to change IP addresses, such that a listener cannot determine which hardware node generated a particular message nor which node is the intended recipient.

FIG. 12A shows a system in which Media Access Control ("MAC") hardware addresses are "hopped" in order to increase security over a network such as an Ethernet. While the description refers to the exemplary case of an Ethernet environment, the inventive principles are equally applicable to other types of communications media. In the Ethernet case, the MAC address of the sender and receiver are inserted into the Ethernet frame and can be observed by anyone on the LAN who is within the broadcast range for that frame. For secure communications, it becomes desirable to generate frames with MAC addresses that are not attributable to any specific sender or receiver.

As shown in FIG. 12A, two computer nodes **1201** and **1202** communicate over a communication channel such as an Ethernet. Each node executes one or more application programs **1203** and **1218** that communicate by transmitting packets through communication software **1204** and **1217**, respectively. Examples of application programs include video conferencing, e-mail, word processing programs, telephony, and the like. Communication software **1204** and **1217** can comprise, for example, an OSI layered architecture or "stack" that standardizes various services provided at different levels of functionality.

The lowest levels of communication software **1204** and **1217** communicate with hardware components **1206** and **1214** respectively, each of which can include one or more registers **1207** and **1215** that allow the hardware to be reconfigured or controlled in accordance with various communication protocols. The hardware components (an Ethernet network interface card, for example) communicate with each other over the communication medium. Each hardware component is typically pre-assigned a fixed hardware address or MAC number that identifies the hardware component to other nodes on the network. One or more interface drivers control the operation of each card and can, for example, be configured to accept or reject packets from certain hardware addresses. As will be described in more detail below, various embodiments of the inventive principles provide for "hopping" different addresses using one or more algorithms and one or more moving windows that track a range of valid addresses to validate received packets. Packets transmitted according to one or more of the inventive principles will be generally referred to as "secure" packets or "secure communications" to differentiate them from ordinary data packets that are transmitted in the clear using ordinary, machine-correlated addresses.

One straightforward method of generating non-attributable MAC addresses is an extension of the IP hopping scheme. In this scenario, two machines on the same LAN that desire to communicate in a secure fashion exchange random-number generators and seeds, and create sequences of quasi-random MAC addresses for synchronized hopping. The implementation and synchronization issues are then similar to that of IP hopping.

This approach, however, runs the risk of using MAC addresses that are currently active on the LAN—which, in turn, could interrupt communications for those machines. Since an Ethernet MAC address is at present 48 bits in length, the chance of randomly misusing an active MAC address is actually quite small. However, if that figure is multiplied by a large number of nodes (as would be found on an extensive LAN), by a large number of frames (as might be the case with packet voice or streaming video), and by a large number of concurrent Virtual Private Networks (VPNs), then the chance that a non-secure machine's MAC address could be used in an address-hopped frame can become non-trivial. In short, any scheme that runs even a small risk of interrupting communications for other machines on the LAN is bound to receive resistance from prospective system administrators. Nevertheless, it is technically feasible, and can be implemented without risk on a LAN on which there is a small number of machines, or if all of the machines on the LAN are engaging in MAC-hopped communications.

Synchronized MAC address hopping may incur some overhead in the course of session establishment, especially if there are multiple sessions or multiple nodes involved in the communications. A simpler method of randomizing MAC addresses is to allow each node to receive and process every incident frame on the network. Typically, each network interface driver will check the destination MAC address in the header of every incident frame to see if it matches that machine's MAC address; if there is no match, then the frame is discarded. In one embodiment, however, these checks can be disabled, and every incident packet is passed to the TARP stack for processing. This will be referred to as "promiscuous" mode, since every incident frame is processed. Promiscuous mode allows the sender to use completely random, unsynchronized MAC addresses, since the destination machine is guaranteed to process the frame. The decision as to whether the packet was truly intended for that machine is handled by the TARP stack, which checks the source and destination IP addresses for a match in its IP synchronization tables. If no match is found, the packet is discarded; if there is a match, the packet is unwrapped, the inner header is evaluated, and if the inner header indicates that the packet is destined for that machine then the packet is forwarded to the IP stack—otherwise it is discarded.

One disadvantage of purely-random MAC address hopping is its impact on processing overhead; that is, since every incident frame must be processed, the machine's CPU is engaged considerably more often than if the network interface driver is discriminating and rejecting packets unilaterally. A compromise approach is to select either a single fixed MAC address or a small number of MAC addresses (e.g., one for each virtual private network on an Ethernet) to use for MAC-hopped communications, regardless of the actual recipient for which the message is intended. In this mode, the network interface driver can check each incident frame against one (or a few) pre-established MAC addresses, thereby freeing the CPU from the task of physical-layer packet discrimination. This scheme does not betray any useful information to an interloper on the LAN; in particular, every secure packet can already be identified by a unique packet type in the outer header. However, since all machines engaged in secure communications would either be using the same MAC address, or be selecting from a small pool of predetermined MAC addresses, the association between a specific machine and a specific MAC address is effectively broken.

In this scheme, the CPU will be engaged more often than it would be in non-secure communications (or in synchronized MAC address hopping), since the network interface driver cannot always unilaterally discriminate between secure packets that are destined for that machine, and secure packets from other VPNs. However, the non-secure traffic is easily eliminated at the network interface, thereby reducing the amount of processing required of the CPU. There are boundary conditions where these statements would not hold, of course—e.g., if all of the traffic on the LAN is secure traffic, then the CPU would be engaged to the same degree as it is in the purely-random address hopping case; alternatively, if each VPN on the LAN uses a different MAC address, then the network interface can perfectly discriminate secure frames destined for the local machine from those constituting other VPNs. These are engineering tradeoffs that might be best handled by providing administrative options for the users when installing the software and/or establishing VPNs.

Even in this scenario, however, there still remains a slight risk of selecting MAC addresses that are being used by one or more nodes on the LAN. One solution to this problem is to formally assign one address or a range of addresses for use in MAC-hopped communications. This is typically done via an assigned numbers registration authority; e.g., in the case of Ethernet, MAC address ranges are assigned to vendors by the Institute of Electrical and Electronics Engineers (IEEE). A formally-assigned range of addresses would ensure that secure frames do not conflict with any properly-configured and properly-functioning machines on the LAN.

Reference will now be made to FIGS. 12A and 12B in order to describe the many combinations and features that follow the inventive principles. As explained above, two computer nodes **1201** and **1202** are assumed to be communicating over a network or communication medium such as an Ethernet. A communication protocol in each node (**1204** and **1217**, respectively) contains a modified element **1205** and **1216** that performs certain functions that deviate from the standard communication protocols. In particular, computer node **1201** implements a first "hop" algorithm **1208X** that selects seemingly random source and destination IP addresses (and, in one embodiment, seemingly random IP header discriminator fields) in order to transmit each packet to the other computer node. For example, node **1201** maintains a transmit table **1208** containing triplets of source (S), destination (D), and discriminator fields (DS) that are inserted into outgoing IP packet headers. The table is generated through the use of an appropriate algorithm (e.g., a random number generator that is seeded with an appropriate seed) that is known to the recipient node **1202**. As each new IP packet is formed, the next sequential entry out of the sender's transmit table **1208** is used to populate the IP source, IP destination, and IP header extension field (e.g., discriminator field). It will be appreciated that the transmit table need not be created in advance but could instead be created on-the-fly by executing the algorithm when each packet is formed.

At the receiving node **1202**, the same IP hop algorithm **1222X** is maintained and used to generate a receive table **1222** that lists valid triplets of source IP address, destination IP address, and discriminator field. This is shown by virtue of the first five entries of transmit table **1208** matching the second five entries of receive table **1222**. (The tables may be slightly offset at any particular time due to lost packets, misordered packets, or transmission delays). Additionally, node **1202** maintains a receive window **W3** that represents a list of valid IP source, IP destination, and discriminator fields that will be accepted when received as part of an incoming IP packet. As packets are received, window **W3** slides down the list of valid entries, such that the possible valid entries change over time. Two packets that arrive out of order but are nevertheless matched to entries within window **W3** will be accepted; those falling outside of window **W3** will be rejected as invalid. The length of window **W3** can be adjusted as necessary to reflect network delays or other factors.

Node **1202** maintains a similar transmit table **1221** for creating IP packets and frames destined for node **1201** using a potentially different hopping algorithm **1221 X**, and node **1201** maintains a matching receive table **1209** using the same algorithm **1209X**. As node **1202** transmits packets to node **1201** using seemingly random IP source, IP destination, and/or discriminator fields, node **1201** matches the incoming packet values to those falling within window **W1** maintained in its receive table. In effect, transmit table **1208** of node **1201** is synchronized (i.e., entries are selected in the same order) to receive table **1222** of receiving node **1202**. Similarly, transmit table **1221** of node **1202** is synchronized to receive table **1209** of node **1201**. It will be appreciated that although a common algorithm is shown for the source, destination and discriminator fields in FIG. 12A (using, e.g., a different seed for each of the three fields), an entirely different algorithm could in fact be used to establish values for each of these fields. It will also be appreciated that one or two of the fields can be "hopped" rather than all three as illustrated.

In accordance with another aspect of the invention, hardware or "MAC" addresses are hopped instead of or in addition to IP addresses and/or the discriminator field in order to improve security in a local area or broadcast-type network. To that end, node **1201** further maintains a transmit table **1210** using a transmit algorithm **1210X** to generate source and destination hardware addresses that are inserted into frame headers (e.g., fields **1101A** and **1101 B** in FIG. **11**) that are synchronized to a corresponding receive table **1224** at node **1202**. Similarly, node **1202** maintains a different transmit table **1223** containing source and destination hardware addresses that is synchronized with a corresponding receive table **1211** at node **1201**. In this manner, outgoing hardware frames appear to be originating from and going to completely random nodes on the network, even though each recipient can determine whether a given packet is intended for it or not. It will be appreciated that the hardware hopping feature can be implemented at a different level in the communications protocol than the IP hopping feature (e.g., in a card driver or in a hardware card itself to improve performance).

FIG. 12B shows three different embodiments or modes that can be employed using the aforementioned principles. In a first mode referred to as "promiscuous" mode, a common hardware address (e.g., a fixed address for source and another for destination) or else a completely random hardware address is used by all nodes on the network, such that a particular packet cannot be attributed to any one node. Each node must initially accept all packets containing the common (or random) hardware address and inspect the IP addresses or discriminator field to determine whether the packet is intended for that node. In this regard, either the IP addresses or the discriminator field or both can be varied in accordance with an algorithm as described above. As explained previously, this may increase each node's overhead since additional processing is involved to determine whether a given packet has valid source and destination hardware addresses.

In a second mode referred to as "promiscuous per VPN" mode, a small set of fixed hardware addresses are used, with a fixed source/destination hardware address used for all nodes communicating over a virtual private network. For example, if there are six nodes on an Ethernet, and the network is to be split up into two private virtual networks such that nodes on one VPN can communicate with only the other two nodes on its own VPN, then two sets of hardware addresses could be used: one set for the first VPN and a second set for the second VPN. This would reduce the amount of overhead involved in checking for valid frames since only packets arriving from the designated VPN would need to be checked. IP addresses and one or more discriminator fields could still be hopped as before for secure communication within the VPN. Of course, this solution compromises the anonymity of the VPNs (i.e., an outsider can easily tell what traffic belongs in which VPN, though he cannot correlate it to a specific machine/person). It also requires the use of a discriminator field to mitigate the vulnerability to certain types of DoS attacks, (For example, without the discriminator field, an attacker on the LAN could stream frames containing the MAC addresses being used by the VPN; rejecting those frames could lead to excessive processing overhead. The discriminator field would provide a low-overhead means of rejecting the false packets.)

In a third mode referred to as "hardware hopping" mode, hardware addresses are varied as illustrated in FIG. 12A, such that hardware source and destination addresses are changed constantly in order to provide non-attributable addressing. Variations on these embodiments are of course possible, and the invention is not intended to be limited in any respect by these illustrative examples.

B. Extending the Address Space Address

Address hopping provides security and privacy. However, the level of protection is limited by the number of addresses in the blocks being hopped. A hopblock denotes a field or fields modulated on a packet-wise basis for the purpose of providing a VPN. For instance, if two nodes communicate with IP address hopping using hopblocks of 4 addresses (2 bits) each, there would be 16 possible address-pair combinations. A window of size 16 would result in most address pairs being accepted as valid most of the time. This limitation can be overcome by using a discriminator field in addition to or instead of the hopped address fields. The discriminator field would be hopped in exactly the same fashion as the address fields and it would be used to determine whether a packet should be processed by a receiver.

Suppose that two clients, each using four-bit hopblocks, would like the same level of protection afforded to clients communicating via IP hopping between two A blocks (24 address bits eligible for hopping). A discriminator field of 20 bits, used in conjunction with the 4 address bits eligible for hopping in the IP address field, provides this level of protection. A 24-bit discriminator field would provide a similar level of protection if the address fields were not hopped or ignored. Using a discriminator field offers the following advantages: (1) an arbitrarily high level of protection can be provided, and (2) address hopping is unnecessary to provide protection. This may be important in environments where address hopping would cause routing problems.

C. Synchronization Techniques

It is generally assumed that once a sending node and receiving node have exchanged algorithms and seeds (or similar information sufficient to generate quasi-random source and destination tables), subsequent communication between the two nodes will proceed smoothly. Realistically, however, two nodes may lose synchronization due to network delays or outages, or other problems. Consequently, it is desirable to provide means for re-establishing synchronization between nodes in a network that have lost synchronization.

One possible technique is to require that each node provide an acknowledgment upon successful receipt of each packet and, if no acknowledgment is received within a certain period of time, to re-send the unacknowledged packet. This approach, however, drives up overhead costs and may be prohibitive in high-throughput environments such as streaming video or audio, for example.

A different approach is to employ an automatic synchronizing technique that will be referred to herein as "self-synchronization." In this approach, synchronization information is embedded into each packet, thereby enabling the receiver to re-synchronize itself upon receipt of a single packet if it

determines that is has lost synchronization with the sender. (If communications are already in progress, and the receiver determines that it is still in sync with the sender, then there is no need to re-synchronize.) A receiver could detect that it was out of synchronization by, for example, employing a "dead-man" timer that expires after a certain period of time, wherein the timer is reset with each valid packet. A time stamp could be hashed into the public sync field (see below) to preclude packet-retry attacks.

In one embodiment, a "sync field" is added to the header of each packet sent out by the sender. This sync field could appear in the clear or as part of an encrypted portion of the packet. Assuming that a sender and receiver have selected a random-number generator (RNG) and seed value, this combination of RNG and seed can be used to generate a random-number sequence (RNS). The RNS is then used to generate a sequence of source/destination IP pairs (and, if desired, discriminator fields and hardware source and destination addresses), as described above. It is not necessary, however, to generate the entire sequence (or the first N−1 values) in order to generate the Nth random number in the sequence; if the sequence index N is known, the random value corresponding to that index can be directly generated (see below). Different RNGs (and seeds) with different fundamental periods could be used to generate the source and destination IP sequences, but the basic concepts would still apply. For the sake of simplicity, the following discussion will assume that IP source and destination address pairs (only) are hopped using a single RNG sequencing mechanism.

In accordance with a "self-synchronization" feature, a sync field in each packet header provides an index (i.e., a sequence number) into the RNS that is being used to generate IP pairs. Plugging this index into the RNG that is being used to generate the RNS yields a specific random number value, which in turn yields a specific IP pair. That is, an IP pair can be generated directly from knowledge of the RNG, seed, and index number; it is not necessary, in this scheme, to generate the entire sequence of random numbers that precede the sequence value associated with the index number provided.

Since the communicants have presumably previously exchanged RNGs and seeds, the only new information that must be provided in order to generate an IP pair is the sequence number. If this number is provided by the sender in the packet header, then the receiver need only plug this number into the RNG in order to generate an IP pair—and thus verify that the IP pair appearing in the header of the packet is valid. In this scheme, if the sender and receiver lose synchronization, the receiver can immediately re-synchronize upon receipt of a single packet by simply comparing the IP pair in the packet header to the IP pair generated from the index number. Thus, synchronized communications can be resumed upon receipt of a single packet, making this scheme ideal for multicast communications. Taken to the extreme, it could obviate the need for synchronization tables entirely; that is, the sender and receiver could simply rely on the index number in the sync field to validate the IP pair on each packet, and thereby eliminate the tables entirely.

The aforementioned scheme may have some inherent security issues associated with it—namely, the placement of the sync field. If the field is placed in the outer header, then an interloper could observe the values of the field and their relationship to the IP stream. This could potentially compromise the algorithm that is being used to generate the IP-address sequence, which would compromise the security of the communications. If, however, the value is placed in the inner header, then the sender must decrypt the inner header before it can extract the sync value and validate the IP pair;

this opens up the receiver to certain types of denial-of-service (DoS) attacks, such as packet replay. That is, if the receiver must decrypt a packet before it can validate the IP pair, then it could potentially be forced to expend a significant amount of processing on decryption if an attacker simply retransmits previously valid packets. Other attack methodologies are possible in this scenario.

A possible compromise between algorithm security and processing speed is to split up the sync value between an inner (encrypted) and outer (unencrypted) header. That is, if the sync value is sufficiently long, it could potentially be split into a rapidly-changing part that can be viewed in the clear, and a fixed (or very slowly changing) part that must be protected. The part that can be viewed in the clear will be called the "public sync" portion and the part that must be protected will be called the "private sync" portion.

Both the public sync and private sync portions are needed to generate the complete sync value. The private portion, however, can be selected such that it is fixed or will change only occasionally. Thus, the private sync value can be stored by the recipient, thereby obviating the need to decrypt the header in order to retrieve it. If the sender and receiver have previously agreed upon the frequency with which the private part of the sync will change, then the receiver can selectively decrypt a single header in order to extract the new private sync if the communications gap that has led to lost synchronization has exceeded the lifetime of the previous private sync. This should not represent a burdensome amount of decryption, and thus should not open up the receiver to denial-of-service attack simply based on the need to occasionally decrypt a single header.

One implementation of this is to use a hashing function with a one-to-one mapping to generate the private and public sync portions from the sync value. This implementation is shown in FIG. **13**, where (for example) a first ISP **1302** is the sender and a second ISP **1303** is the receiver. (Other alternatives are possible from FIG. **13**.) A transmitted packet comprises a public or "outer" header **1305** that is not encrypted, and a private or "inner" header **1306** that is encrypted using for example a link key. Outer header **1305** includes a public sync portion while inner header **1306** contains the private sync portion. A receiving node decrypts the inner header using a decryption function **1307** in order to extract the private sync portion. This step is necessary only if the lifetime of the currently buffered private sync has expired. (If the currently-buffered private sync is still valid, then it is simply extracted from memory and "added" (which could be an inverse hash) to the public sync, as shown in step **1308**.) The public and decrypted private sync portions are combined in function **1308** in order to generate the combined sync **1309**. The combined sync (**1309**) is then fed into the RNG (**1310**) and compared to the IP address pair (**1311**) to validate or reject the packet.

An important consideration in this architecture is the concept of "future" and "past" where the public sync values are concerned. Though the sync values, themselves, should be random to prevent spoofing attacks, it may be important that the receiver be able to quickly identify a sync value that has already been sent—even if the packet containing that sync value was never actually received by the receiver. One solution is to hash a time stamp or sequence number into the public sync portion, which could be quickly extracted, checked, and discarded, thereby validating the public sync portion itself.

In one embodiment, packets can be checked by comparing the source/destination IP pair generated by the sync field with the pair appearing in the packet header. If (1) they match, (2) the time stamp is valid, and (3) the dead-man timer has expired, then re-synchronization occurs; otherwise, the packet is rejected. If enough processing power is available, the dead-man timer and synchronization tables can be avoided altogether, and the receiver would simply resynchronize (e.g., validate) on every packet.

The foregoing scheme may require large-integer (e.g., 160-bit) math, which may affect its implementation. Without such large-integer registers, processing throughput would be affected, thus potentially affecting security from a denial-of-service standpoint. Nevertheless, as large integer math processing features become more prevalent, the costs of implementing such a feature will be reduced.

D. Other Synchronization Schemes

As explained above, if W or more consecutive packets are lost between a transmitter and receiver in a VPN (where W is the window size), the receiver's window will not have been updated and the transmitter will be transmitting packets not in the receiver's window. The sender and receiver will not recover synchronization until perhaps the random pairs in the window are repeated by chance. Therefore, there is a need to keep a transmitter and receiver in synchronization whenever possible and to re-establish synchronization whenever it is lost.

A "checkpoint" scheme can be used to regain synchronization between a sender and a receiver that have fallen out of synchronization. In this scheme, a checkpoint message comprising a random IP address pair is used for communicating synchronization information. In one embodiment, two messages are used to communicate synchronization information between a sender and a recipient:

1. SYNC_REQ is a message used by the sender to indicate that it wants to synchronize; and
2. SYNC_ACK is a message used by the receiver to inform the transmitter that it has been synchronized.

According to one variation of this approach, both the transmitter and receiver maintain three checkpoints (see FIG. **14**):

1. In the transmitter, ckpt_o ("checkpoint old") is the IP pair that was used to re-send the last SYNC_REQ packet to the receiver. In the receiver, ckpt_o ("checkpoint old") is the IP pair that receives repeated SYNC_REQ packets from the transmitter.
2. In the transmitter, ckpt_n ("checkpoint new") is the IP pair that will be used to send the next SYNC_REQ packet to the receiver. In the receiver, ckpt_n ("checkpoint new") is the IP pair that receives a new SYNC_REQ packet from the transmitter and which causes the receiver's window to be re-aligned, ckpt_o set to ckpt_n, a new ckpt_n to be generated and a new ckpt_r to be generated.
3. In the transmitter, ckpt_r is the IP pair that will be used to send the next SYNC_ACK packet to the receiver. In the receiver, ckpt_r is the IP pair that receives a new SYNC_ACK packet from the transmitter and which causes a new ckpt_n to be generated. Since SYNC_ACK is transmitted from the receiver ISP to the sender ISP, the transmitter ckpt_r refers to the ckpt_r of the receiver and the receiver ckpt_r refers to the ckpt_r of the transmitter (see FIG. **14**).

When a transmitter initiates synchronization, the IP pair it will use to transmit the next data packet is set to a predetermined value and when a receiver first receives a SYNC_REQ, the receiver window is updated to be centered on the transmitter's next IP pair. This is the primary mechanism for checkpoint synchronization.

Synchronization can be initiated by a packet counter (e.g., after every N packets transmitted, initiate a synchronization) or by a timer (every S seconds, initiate a synchronization) or a combination of both. See FIG. 15. From the transmitter's perspective, this technique operates as follows: (1) Each transmitter periodically transmits a "sync request" message to the receiver to make sure that it is in sync. (2) If the receiver is still in sync, it sends back a "sync ack" message. (If this works, no further action is necessary). (3) If no "sync ack" has been received within a period of time, the transmitter retransmits the sync request again. If the transmitter reaches the next checkpoint without receiving a "sync ack" response, then synchronization is broken, and the transmitter should stop transmitting. The transmitter will continue to send syne_reqs until it receives a sync_ack, at which point transmission is reestablished.

From the receiver's perspective, the scheme operates as follows: (1) when it receives a "sync request" request from the transmitter, it advances its window to the next checkpoint position (even skipping pairs if necessary), and sends a "sync ack" message to the transmitter. If sync was never lost, then the "jump ahead" really just advances to the next available pair of addresses in the table (i.e., normal advancement).

If an interloper intercepts the "sync request" messages and tries to interfere with communication by sending new ones, it will he ignored if the synchronization has been established or it will actually help to re-establish synchronization.

A window is realigned whenever a re-synchronization occurs. This realignment entails updating the receiver's window to straddle the address pairs used by the packet transmitted immediately after the transmission of the SYNC_REQ packet. Normally, the transmitter and receiver are in synchronization with one another. However, when network events occur, the receiver's window may have to he advanced by many steps during resynchronization. In this case, it is desirable to move the window ahead without having to step through the intervening random numbers sequentially. (This feature is also desirable for the auto-sync approach discussed above).

E. Random Number Generator with a Jump-Ahead Capability

An attractive method for generating randomly hopped addresses is to use identical random number generators in the transmitter and receiver and advance them as packets are transmitted and received. There are many random number generation algorithms that could be used. Each one has strengths and weaknesses for address hopping applications.

Linear congruential random number generators (LCRs) are fast, simple and well characterized random number generators that can be made to jump ahead n steps efficiently. An LCR generates random numbers X_1, X_2, X_3 . . . Xk starting with seed X_0 using a recurrence

$$X_i=(aX_{i-1}+b)\bmod c, \tag{1}$$

where a, b and c define a particular LCR. Another expression for X_i,

$$X_i=((a^i(X_0+b)-b)/(a-1))\bmod c \tag{2}$$

enables the jump-ahead capability. The factor a^i can grow very large even for modest i if left unfettered. Therefore some special properties of the modulo operation can be used to control the size and processing time required to compute (2). (2) can be rewritten as:

$$X_i=(a^i(X_0(a-1)+b)-b)/(a-1)\bmod c. \tag{3}$$

It can be shown that:

$$(a^i(X_0(a-1)+b)-b)/(a-1)\bmod c=((a^i \bmod((a-1)c)(X_0(a-1)+b)-b)/(a-1))\bmod c \tag{4}.$$

$(X_0(a-1)+b)$ can be stored as $(X_0(a-1)+b) \bmod c$, b as b mod c and compute $a^i \bmod((a-1)c)$ (this requires O(log(i)) steps).

A practical implementation of this algorithm would jump a fixed distance, n, between synchronizations; this is tantamount to synchronizing every n packets. The window would commence n IP pairs from the start of the previous window. Using X_j^w, the random number at the jth checkpoint, as X_0 and n as i, a node can store $a^n \bmod((a-1)c)$ once per LCR and set

$$X_{j+1}^w=X_{n(j+1)}=((a^n \bmod((a-1)c)(X_j^w(a-1)+b)-b)/(a-1))\bmod c, \tag{5}$$

to generate the random number for the j+1th synchronization. Using this construction, a node could jump ahead an arbitrary (but fixed) distance between synchronizations in a constant amount of time (independent of n).

Pseudo-random number generators, in general, and LCRs, in particular, will eventually repeat their cycles. This repetition may present vulnerability in the IP hopping scheme. An adversary would simply have to wait for a repeat to predict future sequences. One way of coping with this vulnerability is to create a random number generator with a known long cycle. A random sequence can be replaced by a new random number generator before it repeats. LCRs can be constructed with known long cycles. This is not currently true of many random number generators.

Random number generators can be cryptographically insecure. An adversary can derive the RNG parameters by examining the output or part of the output. This is true of LCGs. This vulnerability can be mitigated by incorporating an encryptor, designed to scramble the output as part of the random number generator. The random number generator prevents an adversary from mounting an attack—e.g., a known plaintext attack—against the encryptor.

F. Random Number Generator Example

Consider a RNG where a=31, b=4 and c=15. For this case equation (1) becomes:

$$X_i=(31X_{i-1}+4)\bmod 15. \tag{6}$$

If one sets X_0=1, equation (6) will produce the sequence 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 0, 4, 8, 12. This sequence will repeat indefinitely. For a jump ahead of 3 numbers in this sequence $a^n=31^3=29791$, $c*(a-1)=15*30=450$ and $a^n \bmod((a-1)c)=31^3 \bmod(15*30)=29791 \bmod(450)=91$. Equation (5) becomes:

$$((91(X_i30+4)-4)/30)\bmod 15 \tag{7}.$$

Table 1 shows the jump ahead calculations from (7). The calculations start at 5 and jump ahead 3.

TABLE 1

I	X_i	$(X_i30 + 4)$	$91\ (X_i30 + 4) - 4$	$((91\ (X_i30 + 4) - 4)/30$	X_{i+3}
1	5	154	14010	467	2
4	2	64	5820	194	14
7	14	424	38580	1286	11
10	11	334	30390	1013	8
13	8	244	22200	740	5

G. Fast Packet Filter

Address hopping VPNs must rapidly determine whether a packet has a valid header and thus requires further processing, or has an invalid header (a hostile packet) and should be immediately rejected. Such rapid determinations will be referred to as "fast packet filtering." This capability protects the VPN from attacks by an adversary who streams hostile packets at the receiver at a high rate of speed in the hope of saturating the receiver's processor (a so-called "denial of service" attack). Fast packet filtering is an important feature for implementing VPNs on shared media such as Ethernet.

Assuming that all participants in a VPN share an unassigned "A" block of addresses, one possibility is to use an experimental "A" block that will never be assigned to any machine that is not address hopping on the shared medium. "A" blocks have a 24 bits of address that can be hopped as opposed to the 8 bits in "C" blocks. In this case a hopblock will be the "A" block. The use of the experimental "A" block is a likely option on an Ethernet because:

1. The addresses have no validity outside of the Ethernet and will not be routed out to a valid outside destination by a gateway.
2. There are 2^{24} (~16 million) addresses that can be hopped within each "A" block. This yields >280 trillion possible address pairs making it very unlikely that an adversary would guess a valid address. It also provides acceptably low probability of collision between separate VPNs (all VPNs on a shared medium independently generate random address pairs from the same "A" block).
3. The packets will not be received by someone on the Ethernet who is not on a VPN (unless the machine is in promiscuous mode) minimizing impact on non-VPN computers.

The Ethernet example will be used to describe one implementation of fast packet filtering. The ideal algorithm would quickly examine a packet header, determine whether the packet is hostile, and reject any hostile packets or determine which active IP pair the packet header matches. The problem is a classical associative memory problem. A variety of techniques have been developed to solve this problem (hashing, B—trees etc). Each of these approaches has its strengths and weaknesses. For instance, hash tables can be made to operate quite fast in a statistical sense, but can occasionally degenerate into a much slower algorithm. This slowness can persist for a period of time. Since there is a need to discard hostile packets quickly at all times, hashing would be unacceptable.

H. Presence Vector Algorithm

A presence vector is a bit vector of length 2^n that can be indexed by n-bit numbers (each ranging from 0 to 2^n-1). One can indicate the presence of k n-bit numbers (not necessarily unique), by setting the bits in the presence vector indexed by each number to 1. Otherwise, the bits in the presence vector are 0. An n-bit number, x, is one of the k numbers if and only if the x^{th} bit of the presence vector is 1. A fast packet filter can be implemented by indexing the presence vector and looking for a 1, which will be referred to as the "test."

For example, suppose one wanted to represent the number 135 using a presence vector. The 135^{th} bit of the vector would be set. Consequently, one could very quickly determine whether an address of 135 was valid by checking only one bit: the 135^{th} bit. The presence vectors could be created in advance corresponding to the table entries for the IP addresses. In effect, the incoming addresses can be used as indices into a long vector, making comparisons very fast. As

each RNG generates a new address, the presence vector is updated to reflect the information. As the window moves, the presence vector is updated to zero out addresses that are no longer valid.

There is a trade-off between efficiency of the test and the amount of memory required for storing the presence vector (s). For instance, if one were to use the 48 bits of hopping addresses as an index, the presence vector would have to be 35 terabytes. Clearly, this is too large for practical purposes. Instead, the 48 bits can be divided into several smaller fields. For instance, one could subdivide the 48 bits into four 12-bit fields (see FIG. 16). This reduces the storage requirement to 2048 bytes at the expense of occasionally having to process a hostile packet. In effect, instead of one long presence vector, the decomposed address portions must match all four shorter presence vectors before further processing is allowed. (If the first part of the address portion doesn't match the first presence vector, there is no need to check the remaining three presence vectors).

A presence vector will have a 1 in the y^{th} bit if and only if one or more addresses with a corresponding field of y are active. An address is active only if each presence vector indexed by the appropriate sub-field of the address is 1.

Consider a window of 32 active addresses and 3 checkpoints. A hostile packet will be rejected by the indexing of one presence vector more than 99% of the time. A hostile packet will be rejected by the indexing of all 4 presence vectors more than 99.9999995% of the time. On average, hostile packets will be rejected in less than 1.02 presence vector index operations.

The small percentage of hostile packets that pass the fast packet filter will be rejected when matching pairs are not found in the active window or are active checkpoints. Hostile packets that serendipitously match a header will be rejected when the VPN software attempts to decrypt the header. However, these cases will be extremely rare. There are many other ways this method can be configured to arbitrate the space/speed tradeoffs.

I. Further Synchronization Enhancements

A slightly modified form of the synchronization techniques described above can be employed. The basic principles of the previously described checkpoint synchronization scheme remain unchanged. The actions resulting from the reception of the checkpoints are, however, slightly different. In this variation, the receiver will maintain between OoO ("Out of Order") and 2×WINDOW_SIZE+OoO active addresses ($1 \leqq OoO \leqq WINDOW\_SIZE$ and $WINDOW\_SIZE \geqq 1$). OoO and WINDOW_SIZE are engineerable parameters, where OoO is the minimum number of addresses needed to accommodate lost packets due to events in the network or out of order arrivals and WINDOW_SIZE is the number of packets transmitted before a SYNC_REQ is issued. FIG. 17 depicts a storage array for a receiver's active addresses.

The receiver starts with the first 2×WINDOW_SIZE addresses loaded and active (ready to receive data). As packets are received, the corresponding entries are marked as "used" and are no longer eligible to receive packets. The transmitter maintains a packet counter, initially set to 0, containing the number of data packets transmitted since the last initial transmission of a SYNC_REQ for which SYNC_ACK has been received. When the transmitter packet counter equals WINDOW_SIZE, the transmitter generates a SYNC_REQ and does its initial transmission. When the receiver receives a SYNC_REQ corresponding to its current CKPT_N, it generates the next WINDOW_SIZE addresses and starts loading them in order starting at the first location after the last active address wrapping around to the beginning of the array after the end of the array has been reached. The receiver's array might look like FIG. **18** when a SYNC_REQ has been received. In this case a couple of packets have been either lost or will be received out of order when the SYNC_REQ is received.

FIG. **19** shows the receiver's array after the new addresses have been generated. If the transmitter does not receive a SYNC_ACK, it will re-issue the SYNC_REQ at regular intervals. When the transmitter receives a SYNC_ACK, the packet counter is decremented by WINDOW_SIZE. If the packet counter reaches 2×WINDOW_SIZE—OoO then the transmitter ceases sending data packets until the appropriate SYNC_ACK is finally received. The transmitter then resumes sending data packets. Future behavior is essentially a repetition of this initial cycle. The advantages of this approach are:

1. There is no need for an efficient jump ahead in the random number generator,
2. No packet is ever transmitted that does not have a corresponding entry in the receiver side
3. No timer based re-synchronization is necessary. This is a consequence of 2.
4. The receiver will always have the ability to accept data messages transmitted within OoO messages of the most recently transmitted message.

J. Distributed Transmission Path Variant

Another embodiment incorporating various inventive principles is shown in FIG. **20**. In this embodiment, a message transmission system includes a first computer **2001** in communication with a second computer **2002** through a network **2011** of intermediary computers. In one variant of this embodiment, the network includes two edge routers **2003** and **2004** each of which is linked to a plurality of Internet Service Providers (ISPs) **2005** through **2010**. Each ISP is coupled to a plurality of other ISPs in an arrangement as shown in FIG. **20**, which is a representative configuration only and is not intended to be limiting. Each connection between ISPs is labeled in FIG. **20** to indicate a specific physical transmission path (e.g., AD is a physical path that links ISP A (element **2005**) to ISP D (element **2008**)). Packets arriving at each edge router are selectively transmitted to one of the ISPs to which the router is attached on the basis of a randomly or quasi-randomly selected basis.

As shown in FIG. **21**, computer **2001** or edge router **2003** incorporates a plurality of link transmission tables **2100** that identify, for each potential transmission path through the network, valid sets of IP addresses that can be used to transmit the packet. For example, AD table **2101** contains a plurality of IP source/destination pairs that are randomly or quasi-randomly generated. When a packet is to be transmitted from first computer **2001** to second computer **2002**, one of the link tables is randomly (or quasi-randomly) selected, and the next valid source/destination address pair from that table is used to transmit the packet through the network. If path AD is randomly selected, for example, the next source/destination IP address pair (which is predetermined to transmit between ISP A (element **2005**) and ISP B (element **2008**)) is used to transmit the packet. If one of the transmission paths becomes degraded or inoperative, that link table can be set to a "down" condition as shown in table **2105**, thus preventing addresses from being selected from that table. Other transmission paths would be unaffected by this broken link.

3. Continuation-in-Part Improvements

The following describes various improvements and features that can be applied to the embodiments described above.

The improvements include: (1) a load balancer that distributes packets across different transmission paths according to transmission path quality; (2) a DNS proxy server that transparently creates a virtual private network in response to a domain name inquiry; (3) a large-to-small link bandwidth management feature that prevents denial-of-service attacks at system chokepoints; (4) a traffic limiter that regulates incoming packets by limiting the rate at which a transmitter can be synchronized with a receiver; and (5) a signaling synchronizer that allows a large number of nodes to communicate with a central node by partitioning the communication function between two separate entities. Each is discussed separately below.

A. Load Balancer

Various embodiments described above include a system in which a transmitting node and a receiving node are coupled through a plurality of transmission paths, and wherein successive packets are distributed quasi-randomly over the plurality of paths. See, for example, FIGS. **20** and **21** and accompanying description. The improvement extends this basic concept to encompass distributing packets across different paths in such a manner that the loads on the paths are generally balanced according to transmission link quality.

In one embodiment, a system includes a transmitting node and a receiving node that are linked via a plurality of transmission paths having potentially varying transmission quality. Successive packets are transmitted over the paths based on a weight value distribution function for each path. The rate that packets will be transmitted over a given path can be different for each path. The relative "health" of each transmission path is monitored in order to identify paths that have become degraded. In one embodiment, the health of each path is monitored in the transmitter by comparing the number of packets transmitted to the number of packet acknowledgements received. Each transmission path may comprise a physically separate path (e.g., via dial-up phone line, computer network, router, bridge, or the like), or may comprise logically separate paths contained within a broadband communication medium (e.g., separate channels in an FDM, TDM, CDMA, or other type of modulated or unmodulated transmission link).

When the transmission quality of a path falls below a predetermined threshold and there are other paths that can transmit packets, the transmitter changes the weight value used for that path, making it less likely that a given packet will be transmitted over that path. The weight will preferably be set no lower than a minimum value that keeps nominal traffic on the path. The weights of the other available paths are altered to compensate for the change in the affected path. When the quality of a path degrades to where the transmitter is turned off by the synchronization function (i.e., no packets are arriving at the destination), the weight is set to zero. If all transmitters are turned off, no packets are sent.

Conventional TCP/IP protocols include a "throttling" feature that reduces the transmission rate of packets when it is determined that delays or errors are occurring in transmission. In this respect, timers are sometimes used to determine whether packets have been received. These conventional techniques for limiting transmission of packets, however, do not involve multiple transmission paths between two nodes wherein transmission across a particular path relative to the others is changed based on link quality.

According to certain embodiments, in order to damp oscillations that might otherwise occur if weight distributions are changed drastically (e.g., according to a step function), a linear or an exponential decay formula can be applied to gradually decrease the weight value over time that a degrading path will be used. Similarly, if the health of a degraded path improves, the weight value for that path is gradually increased.

Transmission link health can be evaluated by comparing the number of packets that are acknowledged within the transmission window (see embodiments discussed above) to the number of packets transmitted within that window and by the state of the transmitter (i.e., on or off). In other words, rather than accumulating general transmission statistics over time for a path, one specific implementation uses the "windowing" concepts described above to evaluate transmission path health.

The same scheme can be used to shift virtual circuit paths from an "unhealthy" path to a "healthy" one, and to select a path for a new virtual circuit.

FIG. **22**A shows a flowchart for adjusting weight values associated with a plurality of transmission links. It is assumed that software executing in one or more computer nodes executes the steps shown in FIG. 22A. It is also assumed that the software can be stored on a computer-readable medium such as a magnetic or optical disk for execution by a computer.

Beginning in step **2201**, the transmission quality of a given transmission path is measured. As described above, this measurement can be based on a comparison between the number of packets transmitted over a particular link to the number of packet acknowledgements received over the link (e.g., per unit time, or in absolute terms). Alternatively, the quality can be evaluated by comparing the number of packets that are acknowledged within the transmission window to the number of packets that were transmitted within that window. In yet another variation, the number of missed synchronization messages can be used to indicate link quality. Many other variations are of course possible.

In step **2202**, a check is made to determine whether more than one transmitter (e.g., transmission path) is turned on. If not, the process is terminated and resumes at step **2201**.

In step **2203**, the link quality is compared to a given threshold (e.g., 50%, or any arbitrary number). If the quality falls below the threshold, then in step **2207** a check is made to determine whether the weight is above a minimum level (e.g., 1%). If not, then in step **2209** the weight is set to the minimum level and processing resumes at step **2201**. If the weight is above the minimum level, then in step **2208** the weight is gradually decreased for the path, then in step **2206** the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are increased).

If in step **2203** the quality of the path was greater than or equal to the threshold, then in step **2204** a check is made to determine whether the weight is less than a steady-state value for that path. If so, then in step **2205** the weight is increased toward the steady-state value, and in step **2206** the weights for the remaining paths are adjusted accordingly to compensate (e.g., they are decreased). If in step **2204** the weight is not less than the steady-state value, then processing resumes at step **2201** without adjusting the weights.

The weights can be adjusted incrementally according to various functions, preferably by changing the value gradually. In one embodiment, a linearly decreasing function is used to adjust the weights; according to another embodiment, an exponential decay function is used. Gradually changing the weights helps to damp oscillators that might otherwise occur if the probabilities were abruptly.

Although not explicitly shown in FIG. **22**A the process can be performed only periodically (e.g., according to a time

schedule), or it can be continuously run, such as in a background mode of operation. In one embodiment, the combined weights of all potential paths should add up to unity (e.g., when the weighting for one path is decreased, the corresponding weights that the other paths will be selected will increase).

Adjustments to weight values for other paths can be prorated. For example, a decrease of 10% in weight value for one path could result in an evenly distributed increase in the weights for the remaining paths. Alternatively, weightings could be adjusted according to a weighted formula as desired (e.g., favoring healthy paths over less healthy paths). In yet another variation, the difference in weight value can be amortized over the remaining links in a manner that is proportional to their traffic weighting.

FIG. **22**B shows steps that can be executed to shut down transmission links where a transmitter turns off. In step **2210**, a transmitter shut-down event occurs. In step **2211**, a test is made to determine whether at least one transmitter is still turned on. If not, then in step **2215** all packets are dropped until a transmitter turns on. If in step **2211** at least one transmitter is turned on, then in step **2212** the weight for the path is set to zero, and the weights for the remaining paths are adjusted accordingly.

FIG. **23** shows a computer node **2301** employing various principles of the above-described embodiments. It is assumed that two computer nodes of the type shown in FIG. **23** communicate over a plurality of separate physical transmission paths. As shown in FIG. **23**, four transmission paths X1 through X4 are defined for communicating between the two nodes. Each node includes a packet transmitter **2302** that operates in accordance with a transmit table **2308** as described above. (The packet transmitter could also operate without using the IP-hopping features described above, but the following description assumes that some form of hopping is employed in conjunction with the path selection mechanism.). The computer node also includes a packet receiver **2303** that operates in accordance with a receive table **2309**, including a moving window W that moves as valid packets are received. Invalid packets having source and destination addresses that do not fall within window W are rejected.

As each packet is readied for transmission, source and destination IP addresses (or other discriminator values) are selected from transmit table **2308** according to any of the various algorithms described above, and packets containing these source/destination address pairs, which correspond to the node to which the four transmission paths are linked, are generated to a transmission path switch **2307**. Switch **2307**, which can comprise a software function, selects from one of the available transmission paths according to a weight distribution table **2306**. For example, if the weight for path X1 is 0.2, then every fifth packet will be transmitted on path X1. A similar regime holds true for the other paths as shown. Initially, each link's weight value can be set such that it is proportional to its bandwidth, which will be referred to as its "steady-state" value.

Packet receiver **2303** generates an output to a link quality measurement function **2304** that operates as described above to determine the quality of each transmission path. (The input to packet receiver **2303** for receiving incoming packets is omitted for clarity). Link quality measurement function **2304** compares the link quality to a threshold for each transmission link and, if necessary, generates an output to weight adjustment function **2305**. If a weight adjustment is required, then the weights in table **2306** are adjusted accordingly, preferably according to a gradual (e.g., linearly or exponentially declining) function. In one embodiment, the weight values for all available paths are initially set to the same value, and only when paths degrade in quality are the weights changed to reflect differences.

Link quality measurement function **2304** can be made to operate as part of a synchronizer function as described above. That is, if resynchronization occurs and the receiver detects that synchronization has been lost (e.g., resulting in the synchronization window W being advanced out of sequence), that fact can be used to drive link quality measurement function **2304**. According to one embodiment, load balancing is performed using information garnered during the normal synchronization, augmented slightly to communicate link health from the receiver to the transmitter. The receiver maintains a count, MESS_R(W), of the messages received in synchronization window W. When it receives a synchronization request (SYNC_REQ) corresponding to the end of window W, the receiver includes counter MESS_R in the resulting synchronization acknowledgement (SYNC_ACK) sent back to the transmitter. This allows the transmitter to compare messages sent to messages received in order to assess the health of the link.

If synchronization is completely lost, weight adjustment function **2305** decreases the weight value on the affected path to zero. When synchronization is regained, the weight value for the affected path is gradually increased to its original value. Alternatively, link quality can be measured by evaluating the length of time required for the receiver to acknowledge a synchronization request. In one embodiment, separate transmit and receive tables are used for each transmission path.

When the transmitter receives a SYNC_ACK, the MESS_R is compared with the number of messages transmitted in a window (MESS_T). When the transmitter receives a SYNC_ACK, the traffic probabilities will be examined and adjusted if necessary. MESS_R is compared with the number of messages transmitted in a window (MESS_T). There are two possibilities:

1. If MESS_R is less than a threshold value, THRESH, then the link will be deemed to be unhealthy. If the transmitter was turned off, the transmitter is turned on and the weight P for that link will be set to a minimum value MIN. This will keep a trickle of traffic on the link for monitoring purposes until it recovers. If the transmitter was turned on, the weight P for that link will be set to:

$$P' = \alpha \times MIN + (1-\alpha) \times P \quad (1)$$

Equation 1 will exponentially damp the traffic weight value to MIN during sustained periods of degraded service.

2. If MESS_R for a link is greater than or equal to THRESH, the link will be deemed healthy. If the weight P for that link is greater than or equal to the steady state value S for that link, then P is left unaltered. If the weight P for that link is less than THRESH then P will be set to:

$$P' = \beta \times S + (1-\beta) \times P \quad (2)$$

where β is a parameter such that 0<=β<=1 that determines the damping rate of P.

Equation 2 will increase the traffic weight to S during sustained periods of acceptable service in a damped exponential fashion.

A detailed example will now be provided with reference to FIG. **24**. As shown in FIG. **24**, a first computer **2401** communicates with a second computer **2402** through two routers **2403** and **2404**. Each router is coupled to the other router through three transmission links. As described above, these may be physically diverse links or logical links (including virtual private networks).

Suppose that a first link L1 can sustain a transmission bandwidth of 100 Mb/s and has a window size of 32; link L2 can sustain 75 Mb/s and has a window size of 24; and link L3 can sustain 25 Mb/s and has a window size of 8. The combined links can thus sustain 200 Mb/s. The steady state traffic weights are 0.5 for link L1; 0.375 for link L2, and 0.125 for link L3. MIN=1 Mb/s, THRESH=0.8 MESS_T for each link, α=0.75 and β=0.5. These traffic weights will remain stable until a link stops for synchronization or reports a number of packets received less than its THRESH. Consider the following sequence of events:

1. Link L1 receives a SYNC_ACK containing a MESS_R of 24, indicating that only 75% of the MESS_T (32) messages transmitted in the last window were successfully received. Link 1 would be below THRESH (0.8). Consequently, link L1's traffic weight value would be reduced to 0.12825, while link L2's traffic weight value would be increased to 0.65812 and link L3's traffic weight value would be increased to 0.217938.
2. Link L2 and L3 remained healthy and link L1 stopped to synchronize. Then link L1's traffic weight value would be set to 0, link L2's traffic weight value would be set to 0.75, and link L33's traffic weight value would be set to 0.25.
3. Link L1 finally received a SYNC_ACK containing a MESS_R of 0 indicating that none of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be below THRESH. Link L1's traffic weight value would be increased to 0.005, link L2's traffic weight value would be decreased to 0.74625, and link L3's traffic weight value would be decreased to 0.24875.
4. Link L1 received a SYNC_ACK containing a MESS_R of 32 indicating that 100% of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.2525, while link L2's traffic weight value would be decreased to 0.560625 and link L3's traffic weight value would be decreased to 0.186875.
5. Link L1 received a SYNC_ACK containing a MESS_R of 32 indicating that 100% of the MESS_T (32) messages transmitted in the last window were successfully received. Link L1 would be above THRESH. Link L1's traffic weight value would be increased to 0.37625; link L2's traffic weight value would be decreased to 0.4678125, and link L3's traffic weight value would be decreased to 0.1559375.
6. Link L1 remains healthy and the traffic probabilities approach their steady state traffic probabilities.

B. Use of a DNS Proxy to Transparently Create Virtual Private Networks

A second improvement concerns the automatic creation of a virtual private network (VPN) in response to a domain-name server look-up function.

Conventional Domain Name Servers (DNSs) provide a look-up function that returns the IP address of a requested computer or host. For example, when a computer user types in the web name "Yahoo.com," the user's web browser transmits a request to a DNS, which converts the name into a four-part IP address that is returned to the user's browser and then used by the browser to contact the destination web site.

This conventional scheme is shown in FIG. **25**. A user's computer **2501** includes a client application **2504** (for example, a web browser) and an IP protocol stack **2505**.

When the user enters the name of a destination host, a request DNS REQ is made (through IP protocol stack **2505**) to a DNS **2502** to look up the IP address associated with the name. The DNS returns the IP address DNS RESP to client application **2504**, which is then able to use the IP address to communicate with the host **2503** through separate transactions such as PAGE REQ and PAGE RESP.

In the conventional architecture shown in FIG. **25**, nefarious listeners on the Internet could intercept the DNS REQ and DNS RESP packets and thus learn what IP addresses the user was contacting. For example, if a user wanted to set up a secure communication path with a web site having the name "Target.com," when the user's browser contacted a DNS to find the IP address for that web site, the true IP address of that web site would be revealed over the Internet as part of the DNS inquiry. This would hamper anonymous communications on the Internet.

One conventional scheme that provides secure virtual private networks over the Internet provides the DNS server with the public keys of the machines that the DNS server has the addresses for. This allows hosts to retrieve automatically the public keys of a host that the host is to communicate with so that the host can set up a VPN without having the user enter the public key of the destination host. One implementation of this standard is presently being developed as part of the FreeS/WAN project (RFC 2535).

The conventional scheme suffers from certain drawbacks. For example, any user can perform a DNS request. Moreover, DNS requests resolve to the same value for all users.

According to certain aspects of the invention, a specialized DNS server traps DNS requests and, if the request is from a special type of user (e.g., one for which secure communication services are defined), the server does not return the true IP address of the target node, but instead automatically sets up a virtual private network between the target node and the user. The VPN is preferably implemented using the IP address "hopping" features of the basic invention described above, such that the true identity of the two nodes cannot be determined even if packets during the communication are intercepted. For DNS requests that are determined to not require secure services (e.g., an unregistered user), the DNS server transparently "passes through" the request to provide a normal look-up function and return the IP address of the target web server, provided that the requesting host has permissions to resolve unsecured sites. Different users who make an identical DNS request could be provided with different results.

FIG. **26** shows a system employing various principles summarized above. A user's computer **2601** includes a conventional client (e.g., a web browser) **2605** and an IP protocol stack **2606** that preferably operates in accordance with an IP hopping function **2607** as outlined above. A modified DNS server **2602** includes a conventional DNS server function **2609** and a DNS proxy **2610**. A gatekeeper server **2603** is interposed between the modified DNS server and a secure target site **04**. An "unsecure" target site **2611** is also accessible via conventional IP protocols.

According to one embodiment, DNS proxy **2610** intercepts all DNS lookup functions from client **2605** and determines whether access to a secure site has been requested. If access to a secure site has been requested (as determined, for example, by a domain name extension, or by reference to an internal table of such sites), DNS proxy **2610** determines whether the user has sufficient security privileges to access the site. If so, DNS proxy **2610** transmits a message to gatekeeper **2603** requesting that a virtual private network be created between user computer **2601** and secure target site **2604**. In one embodiment, gatekeeper **2603** creates "hopblocks" to be used by computer **2601** and secure target site **2604** for secure communication. Then, gatekeeper **2603** communicates these to user computer **2601**. Thereafter, DNS proxy **2610** returns to user computer **2601** the resolved address passed to it by the gatekeeper (this address could be different from the actual target computer) **2604**, preferably using a secure administrative VPN. The address that is returned need not be the actual address of the destination computer.

Had the user requested lookup of a non-secure web site such as site **2611**, DNS proxy would merely pass through to conventional DNS server **2609** the look-up request, which would be handled in a conventional manner, returning the IP address of non-secure web site **2611**. If the user had requested lookup of a secure web site but lacked credentials to create such a connection, DNS proxy **2610** would return a "host unknown" error to the user. In this manner, different users requesting access to the same DNS name could be provided with different look-up results.

Gatekeeper **2603** can be implemented on a separate computer (as shown in FIG. **26**) or as a function within modified DNS server **2602**. In general, it is anticipated that gatekeeper **03** facilitates the allocation and exchange of information needed to communicate securely, such as using "hopped" IP addresses. Secure hosts such as site **2604** are assumed to be equipped with a secure communication function such as an IP hopping function **2608**.

It will be appreciated that the functions of DNS proxy **2610** and DNS server **2609** can be combined into a single server for convenience. Moreover, although element **2602** is shown as combining the functions of two servers, the two servers can be made to operate independently.

FIG. **27** shows steps that can be executed by DNS proxy server **2610** to handle requests for DNS look-up for secure hosts. In step **01**, a DNS look-up request is received for a target host. In step **02**, a check is made to determine whether access to a secure host was requested. If not, then in step **03** the DNS request is passed to conventional DNS server **2609**, which looks up the IP address of the target site and returns it to the user's application for further processing.

In step **02**, if access to a secure host was requested, then in step **04** a further check is made to determine whether the user is authorized to connect to the secure host. Such a check can be made with reference to an internally stored list of authorized IP addresses, or can be made by communicating with gatekeeper **2603** (e.g., over an "administrative" VPN that is secure). It will be appreciated that different levels of security can also be provided for different categories of hosts. For example, some sites may be designated as having a certain security level, and the security level of the user requesting access must match that security level. The user's security level can also be determined by transmitting a request message back to the user's computer requiring that it prove that it has sufficient privileges.

If the user is not authorized to access the secure site, then a "host unknown" message is returned (step **05**). If the user has sufficient security privileges, then in step **06** a secure VPN is established between the user's computer and the secure target site. As described above, this is preferably done by allocating a hopping regime that will be carried out between the user's computer and the secure target site, and is preferably performed transparently to the user (i.e., the user need not be involved in creating the secure link). As described in various embodiments of this application, any of various fields can be "hopped" (e.g., IP source/destination addresses; a field in the header; etc.) in order to communicate securely.

Some or all of the security functions can be embedded in gatekeeper **2603**, such that it handles all requests to connect to secure sites. In this embodiment, DNS proxy **2610** communicates with gatekeeper **2603** to determine (preferably over a secure administrative VPN) whether the user has access to a particular web site. Various scenarios for implementing these features are described by way of example below:

Scenario #1: Client has permission to access target computer, and gatekeeper has a rule to make a VPN for the client. In this scenario, the client's DNS request would be received by the DNS proxy server **2610**, which would forward the request to gatekeeper **2603**. The gatekeeper would establish a VPN between the client and the requested target. The gatekeeper would provide the address of the destination to the DNS proxy, which would then return the resolved name as a result. The resolved address can be transmitted back to the client in a secure administrative VPN.

Scenario #2: Client does not have permission to access target computer. In this scenario, the client's DNS request would be received by the DNS proxy server **2610**, which would forward the request to gatekeeper **2603**. The gatekeeper would reject the request, informing DNS proxy server **2610** that it was unable to find the target computer. The DNS proxy **2610** would then return a "host unknown" error message to the client.

Scenario #3: Client has permission to connect using a normal non-VPN link, and the gatekeeper does not have a rule to set up a VPN for the client to the target site. In this scenario, the client's DNS request is received by DNS proxy server **2610**, which would check its rules and determine that no VPN is needed. Gatekeeper **2603** would then inform the DNS proxy server to forward the request to conventional DNS server **2609**, which would resolve the request and return the result to the DNS proxy server and then back to the client.

Scenario #4: Client does not have permission to establish a normal/non-VPN link, and the gatekeeper does not have a rule to make a VPN for the client to the target site. In this scenario, the DNS proxy server would receive the client's DNS request and forward it to gatekeeper **2603**. Gatekeeper **2603** would determine that no special VPN was needed, but that the client is not authorized to communicate with non-VPN members. The gatekeeper would reject the request, causing DNS proxy server **2610** to return an error message to the client.

C. Large Link to Small Link Bandwidth Management

One feature of the basic architecture is the ability to prevent so-called "denial of service" attacks that can occur if a computer hacker floods a known Internet node with packets, thus preventing the node from communicating with other nodes. Because IP addresses or other fields are "hopped" and packets arriving with invalid addresses are quickly discarded, Internet nodes are protected against flooding targeted at a single IP address.

In a system in which a computer is coupled through a link having a limited bandwidth (e.g., an edge router) to a node that can support a much higher-bandwidth link (e.g., an Internet Service Provider), a potential weakness could be exploited by a determined hacker. Referring to FIG. **28**, suppose that a first host computer **2801** is communicating with a second host computer **2804** using the IP address hopping principles described above. The first host computer is coupled through an edge router **2802** to an Internet Service Provider (ISP) **2803** through a low bandwidth link (LOW BW), and is in turn coupled to second host computer **2804** through parts of the Internet through a high bandwidth link (HIGH BW). In this architecture, the ISP is able to support a high bandwidth to the internet, but a much lower bandwidth to the edge router **2802**.

Suppose that a computer hacker is able to transmit a large quantity of dummy packets addressed to first host computer **2801** across high bandwidth link HIGH BW. Normally, host computer **2801** would be able to quickly reject the packets since they would not fall within the acceptance window permitted by the IP address hopping scheme. However, because the packets must travel across low bandwidth link LOW BW, the packets overwhelm the lower bandwidth link before they are received by host computer **2801**. Consequently, the link to host computer **2801** is effectively flooded before the packets can be discarded.

According to one inventive improvement, a "link guard" function **2805** is inserted into the high-bandwidth node (e.g., ISP **2803**) that quickly discards packets destined for a low-bandwidth target node if they are not valid packets. Each packet destined for a low-bandwidth node is cryptographically authenticated to determine whether it belongs to a VPN. If it is not a valid VPN packet, the packet is discarded at the high-bandwidth node. If the packet is authenticated as belonging to a VPN, the packet is passed with high preference. If the packet is a valid non-VPN packet, it is passed with a lower quality of service (e.g., lower priority).

In one embodiment, the ISP distinguishes between VPN and non-VPN packets using the protocol of the packet. In the case of IPSEC [rfc **2401**], the packets have IP protocols **420** and **421**. In the case of the TARP VPN, the packets will have an IP protocol that is not yet defined. The ISP's link guard, **2805**, maintains a table of valid VPNs which it uses to validate whether VPN packets are cryptographically valid. According to one embodiment, packets that do not fall within any hop windows used by nodes on the low-bandwidth link are rejected, or are sent with a lower quality of service. One approach for doing this is to provide a copy of the IP hopping tables used by the low-bandwidth nodes to the high-bandwidth node, such that both the high-bandwidth and low-bandwidth nodes track hopped packets (e.g., the high-bandwidth node moves its hopping window as valid packets are received). In such a scenario, the high-bandwidth node discards packets that do not fall within the hopping window before they are transmitted over the low-bandwidth link. Thus, for example, ISP **2903** maintains a copy **2910** of the receive table used by host computer **2901**. Incoming packets that do not fall within this receive table are discarded. According to a different embodiment, link guard **2805** validates each VPN packet using a keyed hashed message authentication code (HMAC) [rfc **2104**].

According to another embodiment, separate VPNs (using, for example, hopblocks) can be established for communicating between the low-bandwidth node and the high-bandwidth node (i.e., packets arriving at the high-bandwidth node are converted into different packets before being transmitted to the low-bandwidth node).

As shown in FIG. **29**, for example, suppose that a first host computer **2900** is communicating with a second host computer **2902** over the Internet, and the path includes a high bandwidth link HIGH BW to an ISP **2901** and a low bandwidth link LOW BW through an edge router **2904**. In accordance with the basic architecture described above, first host computer **2900** and second host computer **2902** would exchange hopblocks (or a hopblock algorithm) and would be able to create matching transmit and receive tables **2905**, **2906**, **2912** and **2913**. Then in accordance with the basic architecture, the two computers would transmit packets having seemingly random IP source and destination addresses, and each would move a corresponding hopping window in its receive table as valid packets were received.

Suppose that a nefarious computer hacker **2903** was able to deduce that packets having a certain range of IP addresses (e.g., addresses **100** to **200** for the sake of simplicity) are being transmitted to ISP **2901**, and that these packets are being forwarded over a low-bandwidth link. Hacker computer **2903** could thus "flood" packets having addresses falling into the range **100** to **200**, expecting that they would be forwarded along low bandwidth link LOW BW, thus causing the low bandwidth link to become overwhelmed. The fast packet reject mechanism in first host computer **3000** would be of little use in rejecting these packets, since the low bandwidth link was effectively jammed before the packets could be rejected. In accordance with one aspect of the improvement, however, VPN link guard **2911** would prevent the attack from impacting the performance of VPN traffic because the packets would either be rejected as invalid VPN packets or given a lower quality of service than VPN traffic over the lower bandwidth link. A denial-of-service flood attack could, however, still disrupt non-VPN traffic.

According to one embodiment of the improvement, ISP **2901** maintains a separate VPN with first host computer **2900**, and thus translates packets arriving at the ISP into packets having a different IP header before they are transmitted to host computer **2900**. The cryptographic keys used to authenticate VPN packets at the link guard **2911** and the cryptographic keys used to encrypt and decrypt the VPN packets at host **2902** and host **2901** can be different, so that link guard **2911** does not have access to the private host data; it only has the capability to authenticate those packets.

According to yet a third embodiment, the low-bandwidth node can transmit a special message to the high-bandwidth node instructing it to shut down all transmissions on a particular IP address, such that only hopped packets will pass through to the low-bandwidth node. This embodiment would prevent a hacker from flooding packets using a single IP address. According to yet a fourth embodiment, the high-bandwidth node can be configured to discard packets transmitted to the low-bandwidth node if the transmission rate exceeds a certain predetermined threshold for any given IP address; this would allow hopped packets to go through. In this respect, link guard **2911** can be used to detect that the rate of packets on a given IP address are exceeding a threshold rate; further packets addressed to that same IP address would be dropped or transmitted at a lower priority (e.g., delayed).

D. Traffic Limiter

In a system in which multiple nodes are communicating using "hopping" technology, a treasonous insider could internally flood the system with packets. In order to prevent this possibility, one inventive improvement involves setting up "contracts" between nodes in the system, such that a receiver can impose a bandwidth limitation on each packet sender. One technique for doing this is to delay acceptance of a checkpoint synchronization request from a sender until a certain time period (e.g., one minute) has elapsed. Each receiver can effectively control the rate at which its hopping window moves by delaying "SYNC_ACK" responses to "SYNC_REQ" messages.

A simple modification to the checkpoint synchronizer will serve to protect a receiver from accidental or deliberate overload from an internally treasonous client. This modification is based on the observation that a receiver will not update its tables until a SYNC_REQ is received on hopped address

CKPT_N. It is a simple matter of deferring the generation of a new CKPT_N until an appropriate interval after previous checkpoints.

Suppose a receiver wished to restrict reception from a transmitter to 100 packets a second, and that checkpoint synchronization messages were triggered every 50 packets, A compliant transmitter would not issue new SYNC_REQ messages more often than every 0.5 seconds. The receiver could delay a non-compliant transmitter from synchronizing by delaying the issuance of CKPT_N for 0.5 second after the last SYNC_REQ was accepted.

In general, if M receivers need to restrict N transmitters issuing new SYNC_REQ messages after every W messages to sending R messages a second in aggregate, each receiver could defer issuing a new CKPT_N until M×N×W/R seconds have elapsed since the last SYNC_REQ has been received and accepted. If the transmitter exceeds this rate between a pair of checkpoints, it will issue the new checkpoint before the receiver is ready to receive it, and the SYNC_REQ will be discarded by the receiver. After this, the transmitter will re-issue the SYNC_REQ every T1 seconds until it receives a SYNC_ACK. The receiver will eventually update CKPT_N and the SYNC_REQ will be acknowledged. If the transmission rate greatly exceeds the allowed rate, the transmitter will stop until it is compliant. If the transmitter exceeds the allowed rate by a little, it will eventually stop after several rounds of delayed synchronization until it is in compliance. Hacking the transmitter's code to not shut off only permits the transmitter to lose the acceptance window. In this case it can recover the window and proceed only after it is compliant again.

Two practical issues should be considered when implementing the above scheme:

1. The receiver rate should be slightly higher than the permitted rate in order to allow for statistical fluctuations in traffic arrival times and non-uniform load balancing.
2. Since a transmitter will rightfully continue to transmit for a period after a SYNC_REQ is transmitted, the algorithm above can artificially reduce the transmitter's bandwidth. If events prevent a compliant transmitter from synchronizing for a period (e.g. the network dropping a SYNC_REQ or a SYNC_ACK) a SYNC_REQ will be accepted later than expected. After this, the transmitter will transmit fewer than expected messages before encountering the next checkpoint. The new checkpoint will not have been activated and the transmitter will have to retransmit the SYNC_REQ. This will appear to the receiver as if the transmitter is not compliant. Therefore, the next checkpoint will be accepted late from the transmitter's perspective. This has the effect of reducing the transmitter's allowed packet rate until the transmitter transmits at a packet rate below the agreed upon rate for a period of time.

To guard against this, the receiver should keep track of the times that the last C SYNC_REQs were received and accepted and use the minimum of M×N×W/R seconds after the last SYNC_REQ has been received and accepted, 2×M×N×W/R seconds after next to the last SYNC_REQ has been received and accepted, C×M×N×W/R seconds after $(C-1)^{th}$ to the last SYNC_REQ has been received, as the time to activate CKPT_N. This prevents the receiver from inappropriately limiting the transmitter's packet rate if at least one out of the last C SYNC_REQs was processed on the first attempt.

FIG. **30** shows a system employing the above-described principles. In FIG. **30**, two computers **3000** and **3001** are assumed to be communicating over a network N in accordance with the "hopping" principles described above (e.g., hopped IP addresses, discriminator values, etc.). For the sake of simplicity, computer **3000** will be referred to as the receiving computer and computer **3001** will be referred to as the transmitting computer, although full duplex operation is of course contemplated. Moreover, although only a single transmitter is shown, multiple transmitters can transmit to receiver **3000**.

As described above, receiving computer **3000** maintains a receive table **3002** including a window W that defines valid IP address pairs that will be accepted when appearing in incoming data packets. Transmitting computer **3001** maintains a transmit table **3003** from which the next IP address pairs will be selected when transmitting a packet to receiving computer **3000**. (For the sake of illustration, window W is also illustrated with reference to transmit table **3003**). As transmitting computer moves through its table, it will eventually generate a SYNC_REQ message as illustrated in function **3010**. This is a request to receiver **3000** to synchronize the receive table **3002**, from which transmitter **3001** expects a response in the form of a CKPT_N (included as part of a SYNC_ACK message). If transmitting computer **3001** transmits more messages than its allotment, it will prematurely generate the SYNC_REQ message. (If it has been altered to remove the SYNC_REQ message generation altogether, it will fall out of synchronization since receiver **3000** will quickly reject packets that fall outside of window W, and the extra packets generated by transmitter **3001** will be discarded).

In accordance with the improvements described above, receiving computer **3000** performs certain steps when a SYNC_REQ message is received, as illustrated in FIG. **30**. In step **3004**, receiving computer **3000** receives the SYNC_REQ message. In step **3005**, a check is made to determine whether the request is a duplicate. If so, it is discarded in step **3006**. In step **3007**, a check is made to determine whether the SYNC_REQ received from transmitter **3001** was received at a rate that exceeds the allowable rate R (i.e., the period between the time of the last SYNC_REQ message). The value R can be a constant, or it can be made to fluctuate as desired. If the rate exceeds R, then in step **3008** the next activation of the next CKPT_N hopping table entry is delayed by W/R seconds after the last SYNC_REQ has been accepted.

Otherwise, if the rate has not been exceeded, then in step **3109** the next CKPT_N value is calculated and inserted into the receiver's hopping table prior to the next SYNC_REQ from the transmitter **3101**. Transmitter **3101** then processes the SYNC_REQ in the normal manner.

E. Signaling Synchronizer

In a system in which a large number of users communicate with a central node using secure hopping technology, a large amount of memory must be set aside for hopping tables and their supporting data structures. For example, if one million subscribers to a web site occasionally communicate with the web site, the site must maintain one million hopping tables, thus using up valuable computer resources, even though only a small percentage of the users may actually be using the system at any one time. A desirable solution would be a system that permits a certain maximum number of simultaneous links to be maintained, but which would "recognize" millions of registered users at any one time. In other words, out of a population of a million registered users, a few thousand at a time could simultaneously communicate with a central server, without requiring that the server maintain one million hopping tables of appreciable size.

One solution is to partition the central node into two nodes: a signaling server that performs session initiation for user log-on and log-off (and requires only minimally sized tables), and a transport server that contains larger hopping tables for the users. The signaling server listens for the millions of known users and performs a fast-packet reject of other (bogus) packets. When a packet is received from a known user, the signaling server activates a virtual private link (VPL) between the user and the transport server, where hopping tables are allocated and maintained. When the user logs onto the signaling server, the user's computer is provided with hop tables for communicating with the transport server, thus activating the VPL. The VPLs can be torn down when they become inactive for a time period, or they can be torn down upon user log-out. Communication with the signaling server to allow user log-on and log-off can be accomplished using a specialized version of the checkpoint scheme described above.

FIG. **31** shows a system employing certain of the above-described principles. In FIG. **31**, a signaling server **3101** and a transport server **3102** communicate over a link. Signaling server **3101** contains a large number of small tables **3106** and **3107** that contain enough information to authenticate a communication request with one or more clients **3103** and **3104**. As described in more detail below, these small tables may advantageously be constructed as a special case of the synchronizing checkpoint tables described previously. Transport server **3102**, which is preferably a separate computer in communication with signaling server **3101**, contains a smaller number of larger hopping tables **3108**, **3109**, and **3110** that can be allocated to create a VPN with one of the client computers.

According to one embodiment, a client that has previously registered with the system (e.g., via a system administration function, a user registration procedure, or some other method) transmits a request for information from a computer (e.g., a web site). In one variation, the request is made using a "hopped" packet, such that signaling server **3101** will quickly reject invalid packets from unauthorized computers such as hacker computer **3105**. An "administrative" VPN can be established between all of the clients and the signaling server in order to ensure that a hacker cannot flood signaling server **3101** with bogus packets. Details of this scheme are provided below.

Signaling server **3101** receives the request **3111** and uses it to determine that client **3103** is a validly registered user. Next, signaling server **3101** issues a request to transport server **3102** to allocate a hopping table (or hopping algorithm or other regime) for the purpose of creating a VPN with client **3103**. The allocated hopping parameters are returned to signaling server **3101** (path 3113), which then supplies the hopping parameters to client **3103** via path **3114**, preferably in encrypted form.

Thereafter, client **3103** communicates with transport server **3102** using the normal hopping techniques described above. It will be appreciated that although signaling server **3101** and transport server **3102** are illustrated as being two separate computers, they could of course be combined into a single computer and their functions performed on the single computer. Alternatively, it is possible to partition the functions shown in FIG. **31** differently from as shown without departing from the inventive principles.

One advantage of the above-described architecture is that signaling server **3101** need only maintain a small amount of information on a large number of potential users, yet it retains the capability of quickly rejecting packets from unauthorized users such as hacker computer **3105**. Larger data tables needed to perform the hopping and synchronization functions are instead maintained in a transport server **3102**, and a smaller number of these tables are needed since they are only allocated for "active" links. After a VPN has become inactive for a certain time period (e.g., one hour), the VPN can be automatically torn down by transport server **3102** or signaling server **3101**.

A more detailed description will now be provided regarding how a special case of the checkpoint synchronization feature can be used to implement the signaling scheme described above.

The signaling synchronizer may be required to support many (millions) of standing, low bandwidth connections. It therefore should minimize per-VPL memory usage while providing the security offered by hopping technology. In order to reduce memory usage in the signaling server, the data hopping tables can be completely eliminated and data can be carried as part of the SYNC_REQ message. The table used by the server side (receiver) and client side (transmitter) is shown schematically as element **3106** in FIG. **31**.

The meaning and behaviors of CKPT_N, CKPT_O and CKPT_R remain the same from the previous description, except that CKPT_N can receive a combined data and SYNC_REQ message or a SYNC_REQ message without the data.

The protocol is a straightforward extension of the earlier synchronizer. Assume that a client transmitter is on and the tables are synchronized. The initial tables can be generated "out of band." For example, a client can log into a web server to establish an account over the Internet. The client will receive keys etc encrypted over the Internet. Meanwhile, the server will set up the signaling VPN on the signaling server.

Assuming that a client application wishes to send a packet to the server on the client's standing signaling VPL:

1. The client sends the message marked as a data message on the inner header using the transmitter's CKPT_N address. It turns the transmitter off and starts a timer **T1** noting CKPT_O. Messages can be one of three types: DATA, SYNC_REQ and SYNC_ACK. In the normal algorithm, some potential problems can be prevented by identifying each message type as part of the encrypted inner header field. In this algorithm, it is important to distinguish a data packet and a SYNC_REQ in the signaling synchronizer since the data and the SYNC_REQ come in on the same address.
2. When the server receives a data message on its CKPT_N, it verifies the message and passes it up the stack. The message can be verified by checking message type and other information (i.e., user credentials) contained in the inner header It replaces its CKPT_O with CKPT_N and generates the next CKPT_N. It updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.
3. When the client side receiver receives a SYNC_ACK on its CKPT_R with a payload matching its transmitter side CKPT_O and the transmitter is off, the transmitter is turned on and the receiver side CKPT_R is updated. If the SYNC_ACK's payload does not match the transmitter side CKPT_O or the transmitter is on, the SYNC_ACK is simply discarded.
4. T1 expires: If the transmitter is off and the client's transmitter side CKPT_O matches the CKPTO associated with the timer, it starts timer T1 noting CKPT_O again, and a SYNC_REQ is sent using the transmitter's CKPT_O address. Otherwise, no action is taken.
5. When the server receives a SYNC_REQ on its CKPT_N, it replaces its CKPT_O with CKPT_N and generates the next CKPT_N. It updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.
6. When the server receives a SYNC_REQ on its CKPT_O, it updates its transmitter side CKPT_R to correspond to the client's receiver side CKPT_R and transmits a SYNC_ACK containing CKPT_O in its payload.

FIG. **32** shows message flows to highlight the protocol. Reading from top to bottom, the client sends data to the server using its transmitter side CKPT_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT_N into CKPT_O and updates CKPT_N. This message is successfully received and a passed up the stack. It also synchronizes the receiver i.e., the server loads CKPT_N into CKPT_O and generates a new CKPT_N, it generates a new CKPT_R in the server side transmitter and transmits a SYNC_ACK containing the server side receiver's CKPT_O the server. The SYNC_ACK is successfully received at the client. The client side receiver's CKPT_R is updated, the transmitter is turned on and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

Next, the client sends data to the server using its transmitter side CKPT_N. The client side transmitter is turned off and a retry timer is turned off. The transmitter will not transmit messages as long as the transmitter is turned off. The client side transmitter then loads CKPT_N into CKPT_O and updates CKPT_N. This message is lost. The client side timer expires and as a result a SYNC_REQ is transmitted on the client side transmitter's CKPT_O (this will keep happening until the SYNC_ACK has been received at the client). The SYNC_REQ is successfully received at the server. It synchronizes the receiver i.e., the server loads CKPT_N into CKPT_O and generates a new CKPT_N, it generates an new CKPT_R in the server side transmitter and transmits a SYNC_ACK containing the server side receiver's CKPT_O the server. The SYNC_ACK is successfully received at the client. The client side receiver's CKPT_R is updated, the transmitter is turned off and the retry timer is killed. The client side transmitter is ready to transmit a new data message.

There are numerous other scenarios that follow this flow. For example, the SYNC_ACK could be lost. The transmitter would continue to re-send the SYNC_REQ until the receiver synchronizes and responds.

The above-described procedures allow a client to be authenticated at signaling server **3201** while maintaining the ability of signaling server **3201** to quickly reject invalid packets, such as might be generated by hacker computer **3205**. In various embodiments, the signaling synchronizer is really a derivative of the synchronizer. It provides the same protection as the hopping protocol, and it does so for a large number of low bandwidth connections.

F. One-Click Secure On-line Communications and Secure Domain Name Service

The present invention provides a technique for establishing a secure communication link between a first computer and a second computer over a computer network. Preferably, a user enables a secure communication link using a single click of a mouse, or a corresponding minimal input from another input device, such as a keystroke entered on a keyboard or a click entered through a trackball. Alternatively, the secure link is automatically established as a default setting at boot-up of the computer (i.e., no click). FIG. **33** shows a system block diagram **3300** of a computer network in which the one-click secure communication method of the present invention is suitable. In FIG. 33, a computer terminal or client computer **3301**, such as a personal computer (PC), is connected to a computer network **3302**, such as the Internet, through an ISP **3303**. Alternatively, computer **3301** can be connected to computer network **3302** through an edge router. Computer **3301** includes an input device, such as a keyboard and/or mouse, and a display device, such as a monitor. Computer **3301** can communicate conventionally with another computer **3304** connected to computer network **3302** over a communication link **3305** using a browser **3306** that is installed and operates on computer **3301** in a well-known manner.

Computer **3304** can be, for example, a server computer that is used for conducting e-commerce. In the situation when computer network **3302** is the Internet, computer **3304** typically will have a standard top-level domain name such as .com, .net, .org, .edu, .mil or .gov.

FIG. **34** shows a flow diagram **3400** for installing and establishing a "one-click" secure communication link over a computer network according to the present invention. At step **3401**, computer **3301** is connected to server computer **3304** over a non-VPN communication link **3305**. Web browser **3306** displays a web page associated with server **3304** in a well-known manner. According to one variation of the invention, the display of computer **3301** contains a hyperlink, or an icon representing a hyperlink, for selecting a virtual private network (VPN) communication link ("go secure" hyperlink) through computer network **3302** between terminal **3301** and server **3304**. Preferably, the "go secure" hyperlink is displayed as part of the web page downloaded from server computer **3304**, thereby indicating that the entity providing server **3304** also provides VPN capability.

By displaying the "go secure" hyperlink, a user at computer **3301** is informed that the current communication link between computer **3301** and server computer **3304** is a non-secure, non-VPN communication link. At step **3402**, it is determined whether a user of computer **3301** has selected the "go secure" hyperlink. If not, processing resumes using a non-secure (conventional) communication method (not shown). If, at step **3402**, it is determined that the user has selected the "go secure" hyperlink, flow continues to step **3403** where an object associated with the hyperlink determines whether a VPN communication software module has already been installed on computer **3301**. Alternatively, a user can enter a command into computer **3301** to "go secure."

If, at step **3403**, the object determines that the software module has been installed, flow continues to step **3407**. If, at step **3403**, the object determines that the software module has not been installed, flow continues to step **3404** where a non-VPN communication link **3307** is launched between computer **3301** and a website **3308** over computer network **3302** in a well-known manner. Website **3308** is accessible by all computer terminals connected to computer network **3302** through a non-VPN communication link. Once connected to website **3308**, a software module for establishing a secure communication link over computer network **3302** can be downloaded and installed. Flow continues to step **3405** where, after computer **3301** connects to website **3308**, the software module for establishing a communication link is downloaded and installed in a well-known manner on computer terminal **3301** as software module **3309**. At step **3405**, a user can optionally select parameters for the software module, such as enabling a secure communication link mode of communication for all communication links over computer network **3302**. At step **3406**, the communication link between computer **3301** and website **3308** is then terminated in a well-known manner.

By clicking on the "go secure" hyperlink, a user at computer **3301** has enabled a secure communication mode of communication between computer **3301** and server computer **3304**. According to one variation of the invention, the user is not required to do anything more than merely click the "go secure" hyperlink. The user does not need to enter any user identification information, passwords or encryption keys for establishing a secure communication link. All procedures required for establishing a secure communication link between computer **3301** and server computer **3304** are performed transparently to a user at computer **3301**.

At step **3407**, a secure VPN communications mode of operation has been enabled and software module **3309** begins to establish a VPN communication link. In one embodiment, software module **3309** automatically replaces the top-level domain name for server **3304** within browser **3406** with a secure top-level domain name for server computer **3304**. For example, if the top-level domain name for server **3304** is .com, software module **3309** replaces the .com top-level domain name with a .scom top-level domain name, where the "s" stands for secure. Alternatively, software module **3409** can replace the top-level domain name of server **3304** with any other non-standard top-level domain name.

Because the secure top-level domain name is a non-standard domain name, a query to a standard domain name service (DNS) will return a message indicating that the universal resource locator (URL) is unknown. According to the invention, software module **3409** contains the URL for querying a secure domain name service (SDNS) for obtaining the URL for a secure top-level domain name. In this regard, software module **3309** accesses a secure portal **3310** that interfaces a secure network **3311** to computer network **3302**. Secure network **3311** includes an internal router **3312**, a secure domain name service (SDNS) **3313**, a VPN gatekeeper **3314** and a secure proxy **3315**. The secure network can include other network services, such as e-mail **3316**, a plurality of chatrooms (of which only one chatroom **3317** is shown), and a standard domain name service (STD DNS) **3318**. Of course, secure network **3311** can include other resources and services that are not shown in FIG. 33.

When software module **3309** replaces the standard top-level domain name for server **3304** with the secure top-level domain name, software module **3309** sends a query to SDNS **3313** at step **3408** through secure portal **3310** preferably using an administrative VPN communication link **3319**. In this configuration, secure portal **3310** can only be accessed using a VPN communication link. Preferably, such a VPN communication link can be based on a technique of inserting a source and destination IP address pair into each data packet that is selected according to a pseudo-random sequence; an IP address hopping regime that pseudorandomly changes IP addresses in packets transmitted between a client computer and a secure target computer; periodically changing at least one field in a series of data packets according to a known sequence; an Internet Protocol (IP) address in a header of each data packet that is compared to a table of valid IP addresses maintained in a table in the second computer; and/or a comparison of the IP address in the header of each data packet to a moving window of valid IP addresses, and rejecting data packets having IP addresses that do not fall within the moving window. Other types of VPNs can alternatively be used. Secure portal **3310** authenticates the query from software module **3309** based on the particular information hopping technique used for VPN communication link **3319**.

SDNS **3313** contains a cross-reference database of secure domain names and corresponding secure network addresses. That is, for each secure domain name, SDNS **3313** stores a computer network address corresponding to the secure domain name. An entity can register a secure domain name in SDNS **3313** so that a user who desires a secure communication link to the website of the entity can automatically obtain the secure computer network address for the secure website. Moreover, an entity can register several secure domain names, with each respective secure domain name representing a different priority level of access in a hierarchy of access levels to a secure website. For example, a securities trading website can provide users secure access so that a denial of service attack on the website will be ineffectual with respect to users subscribing to the secure website service. Different levels of subscription can be arranged based on, for example, an escalating fee, so that a user can select a desired level of guarantee for connecting to the secure securities trading website. When a user queries SDNS **3313** for the secure computer network address for the securities trading website, SDNS **3313** determines the particular secure computer network address based on the user's identity and the user's subscription level.

At step **3409**, SDNS **3313** accesses VPN gatekeeper **3314** for establishing a VPN communication link between software module **3309** and secure server **3320**. Server **3320** can only be accessed through a VPN communication link. VPN gatekeeper **3314** provisions computer **3301** and secure web server computer **3320**, or a secure edge router for server computer **3320**, thereby creating the VPN. Secure server computer **3320** can be a separate server computer from server computer **3304**, or can be the same server computer having both non-VPN and VPN communication link capability, such as shown by server computer **3322**. Returning to FIG. **34**, in step **3410**, SDNS **3313** returns a secure URL to software module **3309** for the .scom server address for a secure server **3320** corresponding to server **3304**.

Alternatively, SDNS **3313** can be accessed through secure portal **3310** "in the clear", that is, without using an administrative VPN communication link. In this situation, secure portal **3310** preferably authenticates the query using any well-known technique, such as a cryptographic technique, before allowing the query to proceed to SDNS **3319**. Because the initial communication link in this situation is not a VPN communication link, the reply to the query can be "in the clear." The querying computer can use the clear reply for establishing a VPN link to the desired domain name. Alternatively, the query to SDNS **3313** can be in the clear, and SDNS **3313** and gatekeeper **3314** can operate to establish a VPN communication link to the querying computer for sending the reply.

At step **3411**, software module **3309** accesses secure server **3320** through VPN communication link **3321** based on the VPN resources allocated by VPN gatekeeper **3314**. At step **3412**, web browser **3306** displays a secure icon indicating that the current communication link to server **3320** is a secure VPN communication link. Further communication between computers **3301** and **3320** occurs via the VPN, e.g., using a "hopping" regime as discussed above. When VPN link **3321** is terminated at step **3413**, flow continues to step **3414** where software module **3309** automatically replaces the secure top-level domain name with the corresponding non-secure top-level domain name for server **3304**. Browser **3306** accesses a standard DNS **3325** for obtaining the non-secure URL for server **3304**. Browser **3306** then connects to server **3304** in a well-known manner. At step **3415**, browser **3306** displays the "go secure" hyperlink or icon for selecting a VPN communication link between terminal **3301** and server **3304**. By again

displaying the "go secure" hyperlink, a user is informed that the current communication link is a non-secure, non-VPN communication link.

When software module **3309** is being installed or when the user is off-line, the user can optionally specify that all communication links established over computer network **3302** are secure communication links. Thus, anytime that a communication link is established, the link is a VPN link. Consequently, software module **3309** transparently accesses SDNS **3313** for obtaining the URL for a selected secure website. In other words, in one embodiment, the user need not "click" on the secure option each time secure communication is to be effected.

Additionally, a user at computer **3301** can optionally select a secure communication link through proxy computer **3315**. Accordingly, computer **3301** can establish a VPN communication link **3323** with secure server computer **3320** through proxy computer **3315**. Alternatively, computer **3301** can establish a non-VPN communication link **3324** to a non-secure website, such as non-secure server computer **3304**.

FIG. **35** shows a flow diagram **3500** for registering a secure domain name according to the present invention. At step **3501**, a requester accesses website **3308** and logs into a secure domain name registry service that is available through website **3308**. At step **3502**, the requestor completes an online registration form for registering a secure domain name having a top-level domain name, such as .com, .net, .org, .edu, .mil or .gov. Of course, other secure top-level domain names can also be used. Preferably, the requestor must have previously registered a non-secure domain name corresponding to the equivalent secure domain name that is being requested. For example, a requestor attempting to register secure domain name "website.scom" must have previously registered the corresponding non-secure domain name "website.com".

At step **3503**, the secure domain name registry service at website **3308** queries a non-secure domain name server database, such as standard DNS **3322**, using, for example, a whois query, for determining ownership information relating to the non-secure domain name corresponding to the requested secure domain name. At step **3504**, the secure domain name registry service at website **3308** receives a reply from standard DNS **3322** and at step **3505** determines whether there is conflicting ownership information for the corresponding non-secure domain name. If there is no conflicting ownership information, flow continues to step **3507**, otherwise flow continues to step **3506** where the requestor is informed of the conflicting ownership information. Flow returns to step **3502**.

When there is no conflicting ownership information at step **3505**, the secure domain name registry service (website **3308**) informs the requestor that there is no conflicting ownership information and prompts the requestor to verify the information entered into the online form and select an approved form of payment. After confirmation of the entered information and appropriate payment information, flow continues to step **3508** where the newly registered secure domain name sent to SDNS **3313** over communication link **3326**.

If, at step **3505**, the requested secure domain name does not have a corresponding equivalent non-secure domain name, the present invention informs the requestor of the situation and prompts the requestor for acquiring the corresponding equivalent non-secure domain name for an increased fee. By accepting the offer, the present invention automatically registers the corresponding equivalent non-secure domain name with standard DNS **3325** in a well-known manner. Flow then continues to step **3508**.

G. Tunneling Secure Address Hopping Protocol Through Existing Protocol Using Web Proxy

The present invention also provides a technique for implementing the field hopping schemes described above in an application program on the client side of a firewall between two computer networks, and in the network stack on the server side of the firewall. The present invention uses a new secure connectionless protocol that provides good denial of service rejection capabilities by layering the new protocol on top of an existing IP protocol, such as the ICMP, UDP or TCP protocols. Thus, this aspect of the present invention does not require changes in the Internet infrastructure.

According to the invention, communications are protected by a client-side proxy application program that accepts unencrypted, unprotected communication packets from a local browser application. The client-side proxy application program tunnels the unencrypted, unprotected communication packets through a new protocol, thereby protecting the communications from a denial of service at the server side. Of course, the unencrypted, unprotected communication packets can be encrypted prior to tunneling.

The client-side proxy application program is not an operating system extension and does not involve any modifications to the operating system network stack and drivers. Consequently, the client is easier to install, remove and support in comparison to a VPN. Moreover, the client-side proxy application can be allowed through a corporate firewall using a much smaller "hole" in the firewall and is less of a security risk in comparison to allowing a protocol layer VPN through a corporate firewall.

The server-side implementation of the present invention authenticates valid field-hopped packets as valid or invalid very early in the server packet processing, similar to a standard virtual private network, for greatly minimizing the impact of a denial of service attempt in comparison to normal TCP/IP and HTTP communications, thereby protecting the server from invalid communications.

FIG. 36 shows a system block diagram of a computer network 3600 in which a virtual private connection according to the present invention can be configured to more easily traverse a firewall between two computer networks. FIG. 37 shows a flow diagram 3700 for establishing a virtual private connection that is encapsulated using an existing network protocol.

In FIG. 36 a local area network (LAN) 3601 is connected to another computer network 3602, such as the Internet, through a firewall arrangement 3603. Firewall arrangement operates in a well-known manner to interface LAN 3601 to computer network 3602 and to protect LAN 3601 from attacks initiated outside of LAN 3601.

A client computer 3604 is connected to LAN 3601 in a well-known manner. Client computer 3604 includes an operating system 3605 and a web browser 3606. Operating system 3605 provides kernel mode functions for operating client computer 3604. Browser 3606 is an application program for accessing computer network resources connected to LAN 3601 and computer network 3602 in a well-known manner. According to the present invention, a proxy application 3607 is also stored on client computer 3604 and operates at an application layer in conjunction with browser 3606. Proxy application 3607 operates at the application layer within client computer 3604 and when enabled, modifies unprotected, unencrypted message packets generated by browser 3606 by inserting data into the message packets that are used for forming a virtual private connection between client computer 3604 and a server computer connected to LAN 3601 or computer network 3602. According to the invention, a virtual private connection does not provide the same level of security to the client computer as a virtual private network. A virtual private connection can be conveniently authenticated so that, for example, a denial of service attack can be rapidly rejected, thereby providing different levels of service that can be subscribed to by a user.

Proxy application 3607 is conveniently installed and uninstalled by a user because proxy application 3607 operates at the application layer within client computer 3604. On installation, proxy application 3607 preferably configures browser 3606 to use proxy application for all web communications. That is, the payload portion of all message packets is modified with the data for forming a virtual private connection between client computer 3604 and a server computer. Preferably, the data for forming the virtual private connection contains field-hopping data, such as described above in connection with VPNs. Also, the modified message packets preferably conform to the UDP protocol. Alternatively, the modified message packets can conform to the TCP/IP protocol or the ICMP protocol. Alternatively, proxy application 3606 can be selected and enabled through, for example, an option provided by browser 3606. Additionally, proxy application 3607 can be enabled so that only the payload portion of specially designated message packets is modified with the data for forming a virtual private connection between client computer 3604 and a designated host computer. Specially designated message packets can be, for example, selected predetermined domain names.

Referring to FIG. 37, at step 3701, unprotected and unencrypted message packets are generated by browser 3606. At step 3702, proxy application 3607 modifies the payload portion of all message packets by tunneling the data for forming a virtual private connection between client computer 3604 and a destination server computer into the payload portion. At step, 3703, the modified message packets are sent from client computer 3604 to, for example, website (server computer) 3608 over computer network 3602.

Website 3608 includes a VPN guard portion 3609, a server proxy portion 3610 and a web server portion 3611. VPN guard portion 3609 is embedded within the kernel layer of the operating system of website 3608 so that large bandwidth attacks on website 3608 are rapidly rejected. When client computer 3604 initiates an authenticated connection to website 3608, VPN guard portion 3609 is keyed with the hopping sequence contained in the message packets from client computer 3604, thereby performing a strong authentication of the client packet streams entering website 3608 at step 3704. VPN guard portion 3609 can be configured for providing different levels of authentication and, hence, quality of service, depending upon a subscribed level of service. That is, VPN guard portion 3609 can be configured to let all message packets through until a denial of service attack is detected, in which case VPN guard portion 3609 would allow only client packet streams conforming to a keyed hopping sequence, such as that of the present invention.

Server proxy portion 3610 also operates at the kernel layer within website 3608 and catches incoming message packets from client computer 3604 at the VPN level. At step 3705, server proxy portion 3610 authenticates the message packets at the kernel level within host computer 3604 using the destination IP address, UDP ports and discriminator fields. The authenticated message packets are then forwarded to the authenticated message packets to web server portion 3611 as normal TCP web transactions.

At step 3705, web server portion 3611 responds to message packets received from client computer 3604 in accordance with the particular nature of the message packets by generating reply message packets. For example, when a client computer requests a webpage, web server portion **3611** generates message packets corresponding to the requested webpage. At step **3706**, the reply message packets pass through server proxy portion **3610**, which inserts data into the payload portion of the message packets that are used for forming the virtual private connection between host computer **3608** and client computer **3604** over computer network **3602**. Preferably, the data for forming the virtual private connection is contains field-hopping data, such as described above in connection with VPNs. Server proxy portion **3610** operates at the kernel layer within host computer **3608** to insert the virtual private connection data into the payload portion of the reply message packets. Preferably, the modified message packets sent by host computer **3608** to client computer **3604** conform to the UDP protocol. Alternatively, the modified message packets can conform to the TCP/IP protocol or the ICMP protocol.

At step **3707**, the modified packets are sent from host computer **3608** over computer network **3602** and pass through firewall **3603**. Once through firewall **3603**, the modified packets are directed to client computer **3604** over LAN **3601** and are received at step **3708** by proxy application **3607** at the application layer within client computer **3604**. Proxy application **3607** operates to rapidly evaluate the modified message packets for determining whether the received packets should be accepted or dropped. If the virtual private connection data inserted into the received information packets conforms to expected virtual private connection data, then the received packets are accepted. Otherwise, the received packets are dropped.

While the present invention has been described in connection with the illustrated embodiments, it will be appreciated and understood that modifications may be made without departing from the true spirit and scope of the invention.

What is claimed is:

1. A system for providing a domain name service for establishing a secure communication link, the system comprising:

a domain name service system configured and arranged to be connected to a communication network, store a plurality of domain names and corresponding network addresses, receive a query for a network address, and indicate in response to the query whether the domain name service system supports establishing a secure communication link.

2. The system of claim **1**, wherein at least one of the plurality of domain names comprises a top-level domain name.

3. The system of claim **2**, wherein the top-level domain name is a non-standard top-level domain name.

4. The system of claim **3**, wherein the non-standard top-level domain name is one of .scom, .sorg, .snet, .sgov, .sedu, .smil and .sint.

5. The system of claim **2**, wherein the domain name service system is configured to authenticate the query using a cryptographic technique.

6. The system of claim **1**, wherein the communication network includes the Internet.

7. The system of claim **1**, wherein the domain name service system comprises an edge router.

8. The system of claim **1**, wherein the domain name service system is connectable to a virtual private network through the communication network.

9. The system of claim **8**, wherein the virtual private network is one of a plurality of secure communication links in a hierarchy of secure communication links.

10. The system of claim **8**, wherein the virtual private network is based on inserting into each data packet communicated over a secure communication link one or more data values that vary according to a pseudo-random sequence.

11. The system of claim **8**, wherein the virtual private network is based on a network address hopping regime that is used to pseudorandomly change network addresses in packets transmitted between a first device and a second device.

12. The system of claim **8**, wherein the virtual private network is based on comparing a value in each data packet transmitted between a first device and a second device to a moving window of valid values.

13. The system of claim **8**, wherein the virtual private network is based on a comparison of a discriminator field in a header of each data packet to a table of valid discriminator fields maintained for a first device.

14. The system of claim **1**, wherein the domain name service system is configured to respond to the query for the network address.

15. The system of claim **1**, wherein the domain name service system is configured to provide, in response to the query, the network address corresponding to a domain name from the plurality of domain names and the corresponding network addresses.

16. The system of claim **1**, wherein the domain name service system is configured to receive the query initiated from a first location, the query requesting the network address associated with a domain name, wherein the domain name service system is configured to provide the network address associated with a second location, and wherein the domain name service system is configured to support establishing a secure communication link between the first location and the second location.

17. The system of claim **1**, wherein the domain name service system is connected to a communication network, stores a plurality of domain names and corresponding network addresses, and comprises an indication that the domain name service system supports establishing a secure communication link.

18. The system of claim **1**, wherein at least one of the plurality of domain names is reserved for secure communication links.

19. The system of claim **1**, wherein the domain name service system comprises a server.

20. The system of claim **19**, wherein the domain name service system further comprises a domain name database, and wherein the domain name database stores the plurality of domain names and the corresponding network addresses.

21. The system of claim **1**, wherein the domain name service system comprises a server, wherein the server comprises a domain name database, and wherein the domain name database stores the plurality of domain names and the corresponding network addresses.

22. The system of claim **1**, wherein the domain name service system is configured to store the corresponding network addresses for use in establishing secure communication links.

23. The system of claim **1**, wherein the domain name service system is configured to authenticate the query for the network address.

24. The system of claim **1**, wherein at least one of the plurality of domain names comprises an indication that the domain name service system supports establishing a secure communication link.

25. The system of claim **1**, wherein at least one of the plurality of domain names comprises a secure name.

26. The system of claim **1**, wherein at least one of the plurality of domain names enables establishment of a secure communication link.

27. The system of claim **1**, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location transparently to a user at the first location.

28. The system of claim **1**, wherein the secure communication link uses encryption.

29. The system of claim **1**, wherein the secure communication link is capable of supporting a plurality of services.

30. The system of claim **29**, wherein the plurality of services comprises a plurality of communication protocols, a plurality of application programs, multiple sessions, or a combination thereof.

31. The system of claim **30**, wherein the plurality of application programs comprises items selected from a group consisting of the following: video conferencing, e-mail, a word processing program, and telephony.

32. The system of claim **29**, wherein the plurality of services comprises audio, video, or a combination thereof.

33. The system of claim **1**, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location.

34. The system of claim **33**, wherein the query is initiated from the first location, wherein the second location comprises a computer, and wherein the network address is an address associated with the computer.

35. The system of claim **1**, wherein the domain name service system comprises a domain name database connected to a communication network and storing a plurality of domain names and corresponding network addresses for communication, wherein the domain name database is configured so as to provide a network address corresponding to a domain name in response to a query in order to establish a secure communication link.

36. A non-transitory machine-readable medium comprising instructions executable in a domain name service system, the instructions comprising code for: connecting the domain name service system to a communication network; storing a plurality of domain names and corresponding network addresses; receiving a query for a network address; and indicating in response to the query whether the domain name service system supports establishing a secure communication link.

37. The non-transitory machine-readable medium of claim **36**, wherein the instructions comprise code for storing the plurality of domain names and corresponding network addresses including at least one top-level domain name.

38. The non-transitory machine-readable medium of claim **36**, wherein the instructions comprise code for responding to the query for the network address.

39. The non-transitory machine-readable medium of claim **36**, wherein the instructions comprise code for providing, in response to the query, the network address corresponding to a domain name from the plurality of domain names and the corresponding network addresses.

40. The non-transitory machine-readable medium of claim **36**, wherein the instructions comprise code for receiving the query for a network address associated with a domain name and initiated from a first location, and providing a network address associated with a second location, and establishing a secure communication link between the first location and the second location.

41. The non-transitory machine-readable medium of claim **36**, wherein the instructions comprise code for indicating that the domain name service system supports the establishment of a secure communication link.

42. The non-transitory machine-readable medium of claim **36**, wherein the instructions comprise code for reserving at least one of the plurality of domain names for secure communication links.

43. The non-transitory machine-readable medium of claim **36**, wherein the code resides on a server.

44. The non-transitory machine-readable medium of claim **36**, wherein the instructions comprise code for storing a plurality of domain names and corresponding network addresses so as to define a domain name database.

45. The non-transitory machine-readable medium of claim **36**, wherein the code resides on a server, and the instructions comprise code for creating a domain name database configured to store the plurality of domain names and the corresponding network addresses.

46. The non-transitory machine-readable medium of claim **36**, wherein the instructions comprise code for storing the corresponding network addresses for use in establishing secure communication links.

47. The non-transitory machine-readable medium of claim **36**, wherein the instructions comprise code for authenticating the query for the network address.

48. The non-transitory machine-readable medium of claim **36**, wherein at least one of the plurality of domain names includes an indication that the domain name service system supports the establishment of a secure communication link.

49. The non-transitory machine-readable medium of claim **36**, wherein at least one of the plurality of domain names includes a secure name.

50. The non-transitory machine-readable medium of claim **36**, wherein at least one of the plurality of domain names is configured so as to enable establishment of a secure communication link.

51. The non-transitory machine-readable medium of claim **36**, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location transparently to a user at the first location.

52. The non-transitory machine-readable medium of claim **36**, wherein the secure communication link uses encryption.

53. The non-transitory machine-readable medium of claim **36**, wherein the secure communication link is capable of supporting a plurality of services.

54. The non-transitory machine-readable medium of claim **53**, wherein the plurality of services comprises a plurality of communication protocols, a plurality of application programs, multiple sessions, or a combination thereof.

55. The non-transitory machine-readable medium of claim **54**, wherein the plurality of application programs comprises items selected from a group consisting of the following: video conferencing, e-mail, a word processing program, and telephony.

56. The non-transitory machine-readable medium of claim **53**, wherein the plurality of services comprises audio, video, or a combination thereof.

57. The non-transitory machine-readable medium of claim **36**, wherein the domain name service system is configured to enable establishment of a secure communication link between a first location and a second location.

58. The non-transitory machine-readable medium of claim **57**, wherein the instructions include code for receiving a query initiated from the first location, wherein the second location comprises a computer, and wherein the network address is an address associated with the computer.

59. The non-transitory machine-readable medium of claim 36, wherein the domain name service system comprises a domain name database connected to a communication network and storing a plurality of domain names and corresponding network addresses for communication, wherein the domain name database is configured so as to provide a network address corresponding to a domain name is response to the query in order to establish a secure communication link.

60. A method of providing a domain name service for establishing a secure communication link, the method comprising:

connecting a domain name service system to a communication network;

storing a plurality of domain names and corresponding network addresses; and

upon receiving a query for a network address for communication, indicating whether the domain name service system supports establishing a secure communication link.

* * * * *

Electronic Acknowledgement Receipt

EFS ID:	14960605
Application Number:	13336790
International Application Number:	
Confirmation Number:	6217
Title of Invention:	SYSTEM AND METHOD EMPLOYING AN AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES
First Named Inventor/Applicant Name:	Victor Larson
Customer Number:	23630
Filer:	Toby H. Kusmer.
Filer Authorized By:	
Attorney Docket Number:	77580-151(VRNK-1CP3CNFT1)
Receipt Date:	15-FEB-2013
Filing Date:	23-DEC-2011
Time Stamp:	20:54:48
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Foreign Reference	C26.pdf	986113 428ec891268fa87080eaaa8cfa7a2ad0d0b28397	no	19

Warnings:

Information:

2	Foreign Reference	C27.pdf	8199200 a8fa582ed32622fc0adc879c32cedd5aa6ed289d	no	48
Warnings:					
Information:					
3	Foreign Reference	C28.pdf	1128947 e9375fdfb63581635b68d6263a321db7d4e8aa4b	no	12
Warnings:					
Information:					
4	Non Patent Literature	D1254.pdf	425228 1b734af37f6db3bfa2fbf6717e7151bad0b70d34	no	10
Warnings:					
The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing					
Information:					
5	Non Patent Literature	D1255.pdf	574470 3c086015267fe9e1f49f487afd41588bdd654736	no	5
Warnings:					
Information:					
6	Non Patent Literature	D1256.pdf	636843 b6b8ab6df274e68eef427d02fb95bbda25380ec4	no	6
Warnings:					
Information:					
7	Non Patent Literature	D1257.pdf	232366 f6759c8e9c9b23f90a7382e0e3d5b8373da6bae8	no	3
Warnings:					
The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing					
Information:					
8	Non Patent Literature	D1258.pdf	572382 1d56e80d0d4ab9cc18915e8bfae50b02e4e3afc2	no	18
Warnings:					
Information:					
9	Non Patent Literature	D1259.pdf	564429 f10d4a399953e5606cb2bc3e6903797b7c2178e2	no	13
Warnings:					
Information:					

10	Non Patent Literature	D1260.pdf	573928 7d171a014e47245cff263cc06a841882716c cada	no	18
Warnings:					
Information:					
11	Non Patent Literature	D1261.pdf	720589 1475f1547e4ccbd2d1a4897f948a13f7a7fcc 69f	no	13
Warnings:					
Information:					
12	Non Patent Literature	D1262.pdf	409565 4ee4b0ba0eff1434bb3347ffe0f2bf3744ad0 b3f	no	6
Warnings:					
Information:					
13	Non Patent Literature	D1263.pdf	6778779 36d79200c0a10f7c5edf08c16307b9a77a6e d1d8	no	73
Warnings:					
Information:					
14	Non Patent Literature	D1264.pdf	9664167 f5da575d446cdead0d2127f0b614fa2cd8f3 f044	no	12
Warnings:					
Information:					
15	Non Patent Literature	__D1265part1.pdf	8176028 705f7bd46afc5cf46b74d0d4ecfdd09bad3f d799	no	38
Warnings:					
Information:					
16	Foreign Reference	C25.pdf	3008098 065807bf4b73a003d18827ca2f8ec6ff3380 1a92	no	31
Warnings:					
The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing					
Information:					
17	Non Patent Literature	_D1265part2.pdf	20822780 87437db960c80b7db007f961ad872d5d6a 482f3b	no	121
Warnings:					
Information:					
18	Non Patent Literature	_D1265part3.pdf	1044749 79f66514e9a9e48a757016032d707b2b6fa c7f56	no	10

Warnings:					
Information:					
19	Non Patent Literature	_D1265part4.pdf	11202918 efbe19ee92fe05850b4124f71c38806784a719e9	no	59
Warnings:					
The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing					
Information:					
20	Non Patent Literature	_D1265part5.pdf	23595379 b97b21c254a1b63dc26e9817986e4b080b81c9be	no	142
Warnings:					
Information:					
21	Non Patent Literature	_D1265part6.pdf	10515808 604abdc3cffb26abd6eba0ab646c42022cfd5326	no	79
Warnings:					
Information:					
22	Non Patent Literature	_D1265part7.pdf	9678583 66cd806eee07c785926af98a5008b328230aa9fe	no	56
Warnings:					
Information:					
23	Non Patent Literature	_D1265part8.pdf	9599984 8e1635cf42acec857c0ef5cbe505cbafa6b5970d	no	60
Warnings:					
Information:					
24	Non Patent Literature	D1266.pdf	5201699 0ae584f5461e98ea2eaabd0c81aaac30cd2ed346	no	74
Warnings:					
The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing					
Information:					
25	Non Patent Literature	D1268part1.pdf	9966686 039ca211365c81e14240a28fc3a77ca79908afb8	no	100
Warnings:					
Information:					
26	Non Patent Literature	D1268part2.pdf	8109308 168315ce928ceb4c47339663fa2ff1d375c04d5a	no	100
Warnings:					

Information:					
27	Non Patent Literature	D1268part3.pdf	9980777 fc39c68654853afb3057282101bb3edee29530a7	no	100
Warnings:					
Information:					
28	Non Patent Literature	D1268part4.pdf	9200871 31db4222f327e349cf1bba51e43d4be5bde8461c	no	100
Warnings:					
Information:					
29	Non Patent Literature	D1268part5.pdf	8739047 42e687eab1c4f97e8cdbec4cebe8292574c1a238	no	100
Warnings:					
Information:					
30	Non Patent Literature	D1268part6.pdf	8550043 8980a273a27cf872df5b8a3a8939e6376f4d666b	no	100
Warnings:					
Information:					
31	Non Patent Literature	D1268part7.pdf	8274468 aed7b8b8e6747f3ad646a1eb88c2f4f2ac1afdf2	no	100
Warnings:					
Information:					
32	Non Patent Literature	D1268part8.pdf	7260885 586f6dac84c0958e6fa0692857e7d156886144ec	no	100
Warnings:					
Information:					
33	Non Patent Literature	D1268part9.pdf	8073282 e3ae4bb0871c2747cbf8a35d0daa98d14965a368	no	100
Warnings:					
Information:					
34	Non Patent Literature	D1268part10.pdf	8560232 2009ff8be267bf3dd62fa4a1c385a9c00bff9e5b	no	100
Warnings:					
Information:					
35	Non Patent Literature	D1268part11.pdf	8338473 d9ce938c1efea922374df44140d34e5b5878400c	no	100
Warnings:					

Information:

36	Non Patent Literature	D1268part12.pdf	5506776 e6982247f8da96c9d0980e9ae459b9c87af5a366	no	70

Warnings:

Information:

37	Non Patent Literature	D1267.pdf	736755 b1992fed98554e4b59532cbba41e8797f215668f	no	10

Warnings:

The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing

Information:

38	Non Patent Literature	D1269.pdf	8972105 be1666f598735d40a0c5ab28e0d874aefe5ec4c9	no	63

Warnings:

Information:

39	Non Patent Literature	D1271part2.pdf	4029269 db626da48ada2f728a9a49a060b0f74bb0dc8a4c	no	200

Warnings:

Information:

40	Non Patent Literature	D1271part3.pdf	3504952 bf75da8954442d651d810b504a387458f2e8e525	no	200

Warnings:

Information:

41	Non Patent Literature	D1271part4.pdf	3837133 b3916a75633f2c530780acb19ea45f9eb68be34d	no	200

Warnings:

Information:

42	Non Patent Literature	D1271part5.pdf	3361641 f9e8b446a2b7e8aee842f3feea04052d569efc15	no	200

Warnings:

Information:

43	Non Patent Literature	D1271part6.pdf	2331490 bf576bddb9a97962ac71f7c6d385486f4d390fa4	no	200

Warnings:

Information:

44	Non Patent Literature	D1271part7.pdf	2371238 8de6405f301a972e7856fb1b121840007e405650	no	159
Warnings:					
Information:					
45	Non Patent Literature	_D1271part1.pdf	19751839 cb684627c86c7341832093dc7d47948ef14a30a0	no	100
Warnings:					
Information:					
46	Non Patent Literature	1271part1_2.pdf	1435324 c32b6bb1ecebfe325e922434b3997f3a55361647	no	100
Warnings:					
Information:					
47	Non Patent Literature	_D1272.pdf	9466849 04e0a60844bd23f7253e871fd4b7a6f5f2ba1389	no	78
Warnings:					
Information:					
48	Non Patent Literature	D1274.pdf	5328775 4c0a4192f4afc37e4fd00cb14d1873a12fb34d87	no	3
Warnings:					
Information:					
49	Non Patent Literature	_D1273.pdf	1871264 a554b552e7cf04b46c7ec2729cfcfba16b42e27a	no	12
Warnings:					
Information:					
50	Non Patent Literature	D1270.pdf	1203134 b14266da2fa45887796f7084e87e9a1c30781256	no	19
Warnings:					
The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing					
Information:					
51	Non Patent Literature	D1275part1.pdf	3706485 2ee9a4f0a2505d16367e2b28156a66167bdfa8f8	no	100
Warnings:					
The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing					
Information:					

52	Non Patent Literature	D1275part2.pdf	3726943 87f5abe5af083c48b57ea8af0289b08d877d194e	no	117
Warnings:					
The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing					
Information:					
53	Non Patent Literature	D1276.pdf	4716307 f483105a7d5d81ac0d42ab25b8e8bab719077221	no	274
Warnings:					
The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing					
Information:					
54	Non Patent Literature	D1277.pdf	6592950 bf2f5e7e664399d90f8c8da0db5a6c6331072f91	no	154
Warnings:					
The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing					
Information:					
55	Non Patent Literature	D1278.pdf	7384245 661f70d16db974409d7840194049f9c522082c0b	no	163
Warnings:					
The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing					
Information:					
56	Non Patent Literature	D1279.pdf	6257208 b2830e53df4ca72e2c1d8844d8f95516f878ca1f	no	141
Warnings:					
The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing					
Information:					
57	Non Patent Literature	D1280.pdf	3705608 8ce24f484faec3da2707a6f375682fd491bf241b	no	181
Warnings:					
The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing					
Information:					
58	Non Patent Literature	D1281.pdf	7982752 c9033342ba1e9325ee97570ee7318936fdc8f975	no	198
Warnings:					

The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing					
Information:					
59	Non Patent Literature	D1282.pdf	3126761 47e08f5ed556192529173e50275d195e404eb754	no	154
Warnings:					
The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing					
Information:					
Total Files Size (in bytes):	350274907				

Subst. for form 1449/PTO	Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT *(Use as many sheets as necessary)*	Application Number	**13/336,790**
	Filing Date	**12-23-2011**
	First Named Inventor	**Victor Larson**
	Art Unit	**2453**
	Examiner Name	**Krisna Lim**
	Docket Number	**77580-151(VRNK-0001CP3CNFT1)**

U.S. PATENTS

EXAMINER'S INITIALS	CITE NO.	Patent Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
	A166	5,007,051	04/09/1991	Dolkas et al.	

U.S. PATENT APPLICATION PUBLICATIONS

EXAMINER'S INITIALS	CITE NO.	Patent Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear

FOREIGN PATENT DOCUMENTS

EXAMINER'S INITIALS	CITE NO.	Foreign Patent Document Country Codes - Number - Kind Codes *(if known)*	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines Where Relevant Figures Appear	Translation	
						Yes	No
	C25	JP 09-270803	10/14/1997	Furukawa Electric Co. Ltd.		English Abstract	
	C26	JP 10-111848	04/28/1998	AT&T Corp.		English Abstract	
	C27	JP 10-215244	08/11/1998	Sony Corp.		English Abstract	
	C28	JP 04-117826	04/17/1992	Matsushita Electric Ind. Co. Ltd.		English Abstract	

OTHER ART (Including Author, Title, Date, Pertinent Pages, Etc.)

EXAMINER'S INITIALS	CITE NO.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	
	D1254	Eastlake, "Domain Name System Security Extensions," Network Working Group, RFC: 2535 pages 2-11 (March 1999)	
	D1255	Press Release; VirnetX and Aastra Sign a Patent License Agreement, 4 pages, May 2012, Printed from Website: http://virnetx.com/virnetx-and-aastra-sign-a-patent-license-agreement/	
	D1256	Press Release; VirnetX and Mitel Networks Corporation Sign a Patent License Agreement, 5 pages, July 2012, Printed from Website: http://virnetx.com/virnetx-and-mitel-networks-corporation-sign-a-patent-license-agreement/	
	D1257	Press Relese; Virnetx and NEC Corporation and NEC Corporation of America Sign a Patent License Agreement, 5 pages, August 2012, Printed from Website: http://virnetx.com/virnetx-and-nec-corporation-and-nec-corporation-of-america-sign-a-patent-license-agreement/	
	D1258	Supplemental Declaration of Angelos D. Keromytis, Ph.D from Control No.: 95001789 pp. 1-18, dated December 20, 2012	
	D1259	Supplemental Declaration of Angelos D. Keromytis, Ph.D from Control No.: 95001851 pp. 1-13, dated December 30, 2012	

Subst. for form 1449/PTO **INFORMATION DISCLOSURE STATEMENT BY APPLICANT** *(Use as many sheets as necessary)*	**Complete if Known**	
	Application Number	**13/336,790**
	Filing Date	**12-23-2011**
	First Named Inventor	**Victor Larson**
	Art Unit	**2453**
	Examiner Name	**Krisna Lim**
	Docket Number	**77580-151(VRNK-0001CP3CNFT1)**

	D1260	Supplemental Declaration of Angelos D. Keromytis, Ph.D from Control No.: 95001788 pp. 1-18, dated December 18, 2012	
	D1261	Supplemental Declaration of Angelos D. Keromytis, Ph.D from Control No.: 95001856 pp. 1-13, dated December 30, 2012	
	D1262	VirnetX vs Apple Transcript of Trial, Afternoon Session, 12:05 p.m., dated November 5, 2012	
	D1263	Certified Copy dated September 18, 2012 of U.S. Patent Number 6,502,135, 73 pages	
	D1264	Certified Copy dated December 30, 2009 of Assignment for Patent Application Number 95/047,83 12 pages	
	D1265	Certified Copy dated March 11, 2008 of Patent Application Number 09/504,783, 1500 pages	
	D1266	Certified Copy dated March 30, 2011 of U.S. Patent Number 7,418,504, 74 pages	
	D1267	Certified Copy dated October 17, 2012 of Assignment for Patent Application Number: 10/714,849, 10 pages	
	D1268	Certified Copy dated April 4, 2011 of Patent Application Number 10/714,849, 1170 pages	
	D1269	Certified Copy dated March 30, 2011 of U.S. Patent Number 7,490,151, 63 pages	
	D1270	Certified Copy dated October 17, 2012 of Assignment for Patent Application Number 10/259,494, 19 pages	
	D1271	Certified Copy dated April 4, 2011 of Application Number 10/259,454, 1359 pages	
	D1272	Certified Copy dated April 12, 2011 of U.S. Patent Number 7,921,211, 78 pages	
	D1273	Certified Copy dated October 17, 2012 of Assignment for Application Number 11/840,560, 12 pages	
	D1274	Certified Copy dated April 20, 2011 of Application Number 11/840,560, 3 pages	
	D1275	iPhone User Guide for iPhone OS 3.1 Software, 217 pages, 2009	
	D1276	iPhone User Guide for iOS 4.2 and 4.3 Software, 274 pages, 2011	
	D1277	iPhone User Guide for iPhone and iPhone 3G, 154 pages, 2008	
	D1278	iPhone User Guide for iOS 5.0 Software, 163 pages, 2011	
	D1279	iPad User Guide for iOS 5.0 Software, 141 pages, 2011	
	D1280	iPad User Guide for iOS 4.2 Software, 181 pages, 2010	
	D1281	iPad User Guide for iOS 4.3 Software, 198 pages, 2011	
	D1282	iPad User Guide, 154 pages, 2010	
	D1283	iPod Touch User Guide for iOS 5.0 Software, 143 pages, 2011	
	D1284	iPod Touch User Guide, 122 pages, 2008	
	D1285	iPod Touch User Guide for iPhone OS 3.0 Software, 153 pages, 2009	
	D1286	iPod Touch User Guide for iPhone OS 3.1 Software, 169 pages, 2009	
	D1287	iPod Touch User Guide for iOS 4.3 Software, 230 pages, 2011	
	D1288	iPod Touch Features Guide, 98 pages, 2008	

Subst. for form 1449/PTO **INFORMATION DISCLOSURE STATEMENT BY APPLICANT** ***(Use as many sheets as necessary)***	**Complete if Known**
Application Number	**13/336,790**
Filing Date	**12-23-2011**
First Named Inventor	**Victor Larson**
Art Unit	**2453**
Examiner Name	**Krisna Lim**
Docket Number	**77580-151(VRNK-0001CP3CNFT1)**

	D1289	VPN Server Configuration for iOS; Networking & Internet Enterprise Deployment, 12 pages, 2011	
	D1290	iPhone Configuration Utility User Guide, 26 pages, 2010	
	D1291	iPhone Configuration Utility; Networking & Internet: Enterprise Deployment, 26 pages, 2011	
	D1292	iPhone Configuration Utility; Networking>Internet & Web, 24 pages, 2010	
	D1293	iOS Configuration Profile Reference; Networking & Internet: Enterprise Deployment, 24 pages, 2011	
	D1294	iPhone OS Enterprise Deployment Guide; Second Edition, for Version 3.1 or Later, 91 pages, 2009	
	D1295	iPhone OS; Enterprise Deployment Guide; Second Edition, for Version 3.2 or Later, 90 pages, 2010	
	D1296	CFHost Reference; Developer, 20 pages, 2008	
	D1297	CFNetwork Programming Guide; Developer, 60 pages, 2011	
	D1298	CFStream Socket Additions; Developer, 22 pages, 2010	
	D1299	Mac OS X Devloper Library; CFHostSample.c, 1 page	
	D1300	Mac OS X Developer Library; CFHostSample, 1 page, 2004	
	D1301	Mac OS X Developer Library; Document Revision History, 1 page, 2004	
	D1302	CFStream Socket Additions; Developer, 22 pages, 2010	
	D1303	Apple Push Notification Service; Distribution Service, Version 1.0, 6 pages, 2009	
	D1304	iOS Human Interface Guidelines; Developer, 184 pages, 2012	
	D1305	Networking & Internet Starting Point, 3 pages, 2011	
	D1306	Server Admin. 10.5 Help; Viewing a VPN Overview, 1 page	
	D1307	iOS: Supported Protocols for VPN, 2 pages, 2010	
	D1308	IPhone in Business Virtual Private Networks (VPN), 3 pages, 2010	
	D1309	iPhone and iPad in Business Deployment Scenarios, 26 pages, 2011	
	D1310	Deploying iPhone and iPad Virtual Private Networks, 3 pages, 2011	
	D1311	Deploying iPhone and iPad; Security Overview, 6 pages, 2011	
	D1312	Pad in Business; "Ready for Work," 2012, 5 pages	
	D1313	iOS: Using FaceTime, 2 pages, 2011, Printed from website http://support.apple.com/kb/HT4317	
	D1314	MobileMe: "Secure Chat" is Unavailable in OS X Lion, 2 pages, 2012, Printed from Website: http://support.apple.com/kb/TS3902	
	D1315	iPhone 4 and iPod Touch (4th Generation): Using FaceTime, 2 pages, 2010, Printed from Website: http://support.apple.com/kb/HT4319	
	D1316	IPhone; "Picking Up Where Amazing Left Off," 11 pages, 2012, Printed from Website: http://www.apple.com/iPhone/features/facetime	
	D1317	FaceTime for Mac; "Say Hello to FaceTime for Mac," 4 pages, 2012, Printed from Website: http://www.apple.com/mac/facetime	

Subst. for form 1449/PTO **INFORMATION DISCLOSURE STATEMENT BY APPLICANT** *(Use as many sheets as necessary)*	**Complete if Known**	
	Application Number	**13/336,790**
	Filing Date	**12-23-2011**
	First Named Inventor	**Victor Larson**
	Art Unit	**2453**
	Examiner Name	**Krisna Lim**
	Docket Number	**77580-151(VRNK-0001CP3CNFT1)**

	D1318	iPad; "Your New Favorite Way to do Just About Everything," 8 pages, 2012, Printed from Website" http://www.apple.com/ipad/built-in-apps/	
	D1319	iPod Touch; FaceTime, "Oh, I see what you're saying," 2 pages	
	D1320	Apple Press Info; Apple Presents iPhone 4, Printed from Website: http://www.apple.com/pr/library/apple-presents-iphone	
	D1321	iPod Touch; FaceTime, "Oh I See What You're Saying,", 3 pages, 2012, Printed from Website: http://www.apple.com/iPodtouch/built-in-apps/facetime.htm	
	D1322	IOS 4, The World's Most Advanced Mobile Operating System, 5 pages, Printed from Website: http://www.apple.com/iphone/ios4	
	D1323	Apple Press Info; Apple Reinvents the Phone with iPhone, 3 pages, 2007, Printed from Website: http://www.apple.com/pr/library/2007/01/09Apple-reinvents-the-phone	
	D1324	Apple Press Info; Apple Announces the New iPhone 3Gs-The Fastest, Most Powerful iPhone Yet, 3 pages, 2009, Printed from the Website: http://www.apple.com/pr/library/2009/06/08Apple-Announces-the-new-iphone3GS	
	D1325	Apple Press Info; Apple Launches iPhone 4S, ios 5 & iCloud, iPhone 4S Features Dual-Core A5 Chip, All New Camera, full 1080p HD Video Recording & Introduces Siri, 2011, 2 pages, Printed from website: http://www.apple.com/pr/library/2011/10/04Apple-Launches-iPhone-4S-iOS-5-iCloud.html	
	D1326	Apple Press Info; Apple Introduces New iPod Touch, Features Retina Display, A4 Chip, FaceTime Video Calling, HD Video Recording & Game Center, 2 pages, 2010, Printed from Website http://www.apple.com/pr/library/2010/09/01Apple-Introduces-New-iPod-touch.html	
	D1327	Apple Press Info; Apple Launches iPad, Magical & Revolutionary Device at an Unbelievable Price, 2 pages, 2010, Printed from Website: http://www.apple.com/pr/library/2010/01/27Apple-Launches-iPad.html	
	D1328	Apple Press Info; Apple Launces New iPad, New iPad Features Retina Display, A5X Chip, 5 Megapixel iSight Camera & Ultrafast 4G LTE, 2012, 3 pages, Printed from the Website: http://www.apple.com/pr/library/2012/03/07Apple-Launches-New-iPad.html	
	D1329	FaceTime; "Phone Calls Like You've Never Seen Before," 3 pages	
	D1330	Apple Press Info; Apple Brings FaceTime to the Mac, 1 pages, Printed from Website https://www.apple.com/pr/library/2010/10/20Apple-Brings-FaceTime-to-the-Mac.html	
	D1331	iPad at Work; "Mobile Meetings Made Easy," 4 pages, 2011	
	D1332	IPad – Technical Specifications, 49 pages, Printed from Website: http://support.apple.com/kb/sp58C	
	D1333	Stirling Design, 8 pages, 2008	
	D1334	Quick Guide: SSL VPN Technical Primer, 11 pages, 2010	
	D1335	Silva, "Secure iPhone Access to Corporate Web Applications," Technical Brief, 10 pages	
	D1336	Defendant Apple Inc.'s Third Supplemental Responses to VirnetX Inc.'s First Request for Admission to Apple Inc. dated, April 13, 2012, 207 pages	
	D1337	Apple Support Communities, 4 pages, Printed from Website https://discussions.apple.com/thread/486096?start=0&tstart=0	
	D1338	VirnetX – Products; License and Service Offerings, 1 page	

Subst. for form 1449/PTO	Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT *(Use as many sheets as necessary)*	Application Number	**13/336,790**
	Filing Date	**12-23-2011**
	First Named Inventor	**Victor Larson**
	Art Unit	**2453**
	Examiner Name	**Krisna Lim**
	Docket Number	**77580-151(VRNK-0001CP3CNFT1)**

	D1339	VirnetX Contact Information, 4 pages, 2011	
	D1340	VirnetX Launches Secure Domain Name Initiative; 4G/LTE Security, 1 page, 2010	
	D1341	VirnetX Gabriel Connection; Enabling Safe Network Neighborhoods, 2 pages, 2012	
	D1342	Baugher et al., "The Secure Real-Time Transport Protocol (SRTP)," Network Working Group, RFC:3711, 39 pages, 2004	
	D1343	Jennings et al., "Resource Location and Discovery (Reload) Draft-Bryan-P2PSIP-Reload-04," Internet-Draft, 12/12/08, pages 1-127	
	D1344	Barnes et al., "Verification Involving PSTN Reachability: Requirements and Architecture Overview," Internet Draft, 27 pages, 2012	
	D1345	April Inc. Form 10-K (Annual Report) filed 12/01/05 for the Period Ending 09/24/05, Edgar Online, 1400 pages, 2011	
	D1346	Phone, Facetime; "Be in Two Places at Once," 3 pages, Printed from the Website http://www.apple.com/ios/facetime/	
	D1347	Apple Press Info; Apple Presents iPhone 4, All-New Design with FaceTime Video Calling, Retina, Display, 5 Megapixel Camera & HD Video Recording, 3 pages, 2010	
	D1348	NYSE AMEX:VHC; Cowen and Co. 39th Annual Technology Media & Telecom Conference, 36 pages, June 2, 2011	
	D1349	Pindyck et al., "Market Power: Monopoly and Monopsony," Microeconomics, Sixth Edition, pages 370-371	
	D1350	Press Release; IpCapital Group Completes VirnetX IP Licensing Evaluation, 3 pages	
	D1351	Microsoft Real-Time Communications: Protocols and Technologies, Microsoft TechNet, 22 pages, 2010	
	D1352	Filing Receipt dated September 23, 2011 for Application Number: 13/223,259	
	D1353	Email Communications Regarding Apple Product Innovations, 6 pages, 2010	
	D1354	Mathy et al., "Traversal Using Relays Around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)," Internet Engineering Task Force (IETF), RFC: 5766, 67 pages, 2010	
	D1355	Egevang et al., "The IP Network Address Translator (NAT)," Network Working Group, RFC: 1631, 10 pages, 1994	
	D1356	Srisuresh et al., "IP Network Address Translator (NAT) Terminology and Considerations," Network Working Group, RFC:2663, 30 pages, 1999	
	D1357	Sisalem, et al., "Introduction to Cryptographic Mechanisms," SIP Security, 356 pages, 2009	
	D1358	Curriculum Vitae, Mark T Jones, 9 pages	
	D1359	Curriculum Vitae, Roy Weinstein, 5 pages	
	D1360	How To Configure IPSec Tunneling in Windows 2000, 8 pages	
	D1361	Press Relese; Virnetx and NEC Corporation and NEC Corporation of America Sign a Patent License Agreement, 5 pages, August 2012, Printed from Website: http://virnetx.com/virnetx-and-nec-corporation-and-nec-corporation-of-america-sign-a-patent-license-agreement/	

Subst. for form 1449/PTO **INFORMATION DISCLOSURE STATEMENT BY APPLICANT** *(Use as many sheets as necessary)*	**Complete if Known**	
	Application Number	**13/336,790**
	Filing Date	**12-23-2011**
	First Named Inventor	**Victor Larson**
	Art Unit	**2453**
	Examiner Name	**Krisna Lim**
	Docket Number	**77580-151(VRNK-0001CP3CNFT1)**

	D1362	iPhone, FaceTime; "Be in Two Places at Once," 3 pages, Printed from Website: http://www.apple.com/ios/facetime/	
	D1363	iPhone, "It Does Everything Better,"6 pages, Printed from Website: http://www.apple.com/iPhone/built-in-apps	
	D1364	My Apple ID, "What's an Apple ID," 1 pages, Printed from Website: https://appleid.apple.com/cgi-bin/webobjects/myappleid.woa	
	D1365	Rosenberg et al., "Session Initiation Protocol (SIP): Locating SIP Servers," Network Working Group, RFC: 3263, 17 pages, 2002	
	D1366	Certified Copy dated September 21, 2012 of Reexamination Certificate Number 6,502,135 issued June 6, 2011, 11 pages	
	D1367	Certified Copy dated September 20, 2012 of Patent Application Number 95/001,269, 4999 pages	
	D1368	Chatterjee et al., "Bargaining Under Incomplete Information," Operations Research, 31:835-851, 1983	
	D1369	Nash, "The Bargaining Problem," Econometrica, 18:155-162, 1950	
	D1370	Nash, "Two-Person Cooperative Games," Econometrica, 21:128-140, 1953	
	D1371	Choi et al., "An Analytical Solution to Reasonable Royalty Rate Calculations," IDEA: The Journal of Law and Technology, 13 pages, 2001	
	D1372	The Prize in Economics 1994 - Press Release dated October 11, 1994, 4 pages, Printed from Website: http://www.nobelprize.org/nobel_prizes/economics/laureates/1994/press.html	
	D1373	Putnam et al., "Bargaining and the Construction of Economically Consistent Hypothetical License Negotiations," The Licensing Journal, pages 8-15, 2004	
	D1374	Scherling et al., "Rational Reasonable Royalty Damages: A Return to the Roots," Landslide, Volume 4, 4 pages, 2011	
	D1375	Jarosz et al., "Application of Game Theory to Intellectual Property Royalty Negotiations," Chapter 17, pages 241-265	
	D1376	Goldscheider, Licensing Best Practices; Strategic, Territorial, and Technology Issues, 2 pages, 2006	
	D1377	iPhone Configuration Utility, 19 pages, 2012	
	D1378	VPN Server Configuration for iOS Devices, 6 pages, 2012	
	D1379	Samuelson et al., Economics, Fourteenth Edition, pages 258-259, 1992	
	D1380	Stigler et al., The Theory of Price, Forth Edition, pages 215-216, 1987	
	D1381	Truett et al., "Joint Profit Maximization, Negotiation, and the Determinacy of Price in Bilateral Monopoly," Journal of Economic Education, pages 260-270	
	D1382	Binmore et al., "Noncooperative Models of Bargaining," The Handbook of Game Theory, 1:(7)181-225,1992	
	D1383	Spindler et al., "Endogenous Bargaining Power in Bilateral Monopoly and Bilateral Exchange," Canadian Journal of Economics-Revue Canadienne D Economie, pages 464-474, 1974	
	D1384	Myerson, "Game Theory; Analysis or Conflict," Harvard University Press, pages 375-392	

Subst. for form 1449/PTO	Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT *(Use as many sheets as necessary)*	Application Number	**13/336,790**
	Filing Date	**12-23-2011**
	First Named Inventor	**Victor Larson**
	Art Unit	**2453**
	Examiner Name	**Krisna Lim**
	Docket Number	**77580-151(VRNK-0001CP3CNFT1)**

	D1385	Binmore, "The Nash Bargaining Solution in Economic Modelling," The Rand Journal of Economics, 17:176-188, 1996	
	D1386	Rubinstein et al., "On the Interpretation of the Nash Bargaining Solution and its Extension to Non-Expected Utility Preferences," Econometrica, 60:1171-1186, 1992	
	D1387	Greenleaf et al., "Guarantees in Auctions: The Auction House as Negotiator and Managerial Decision Maker," Management Science, 39:1130-1145, 1993	
	D1388	Chan, "Trade Negotiations in a Nash Bargaining Model," Journal of International Economics, 25:253-363, 1987	
	D1389	Lemley et al., "Patent Holdup and Royalty Stacking," Texas Law Review, 85:1991-2049	
	D1390	Cauley, "Winning the Patent Damages Case; A Litigator's Guide to Economic Models and Other Damage Strategies," Oxford Press, pages 29-30, 2044	
	D1391	Degnan et al., "A Survey of Licensed Royalties," Les Nouvelles, pages 91, 93, 94, 1997	
	D1392	Kahn, "The Review of Economics and Statistics," pages 157-164, 1993	
	D1393	Microsoft Company Information; Including Stocks and Financial Information, 83 pages	
	D1394	Apple Press Info: Apple Updates MacBook Pro with Next Generation Processors, Graphics & Thunderbolt I/O Technology, 3 pages, Printed from Website: http://www.apple.com/pr/library/2011/02/24Apple-Updates-MacBook-Pro-with-Next-Generation-Processors-Graphics-Thunderbolt-I-O-Technology.html	
	D1395	Apple Press Info: Apple to Ship Mac OS X Snow Leopard on August 28, 2 pages, Printed from the Website: http://www.apple.com/pr/library/2009/08/24/apple-to-ship-mac-os-x	
	D1396	iPad, Facetime; "Once Again, iPad gets the World Talking," 3 pages, Printed from the Website: http://www.apple.com/ipad/built-in-apps/facetime/html	
	D1397	Apple iOS: Setting up VPN, 2 pages, Printed from Website: http/support.apple.com/kb/HT1424	
	D1398	Apple iPhone User Guide for iOS 5.1 Software, 179 pages, 2012	
	D1399	Apple, Communicating with HTTP Servers, CFNetworking Programming Guide, 6 pages, 2011, Printed from the Website: https://developer.apple.com/library/ios/documentation/networking/conceptual/CFNetwork/CFHT	
	D1400	VirnetX, Gabriel Connection Technology ^{TM} White Paper, 7 pages, 2012	

Subst. for form 1449/PTO	Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT *(Use as many sheets as necessary)*	Application Number	**13/336,790**
	Filing Date	**12-23-2011**
	First Named Inventor	**Victor Larson**
	Art Unit	**2453**
	Examiner Name	**Krisna Lim**
	Docket Number	**77580-151(VRNK-0001CP3CNFT1)**

	D1401	VirnetX, Technology, 4 pages, 2012	
	D1402	Certified Copy dated January 15, 2008 of U.S. Patent Number 6,502,135, 64 pages	
	D1403	Inter Partes Reexamination Certificate dated June 7, 2011 for Patent Number 6,502,135	
	D1404	Proceedings of The Symposium on Network and Distributed System Security, 7 pages, February 22-23, 1996	
	D1405	In-Q-Tel; Corporate Overview, 2 pages, 2004	
	D1406	Davies, Supervisor of Translation: Tadahiro Uezono, Security for Computer Networks, Japan, Nikkei-McGraw-Hill Inc., First Edition, First Copy, p 126-129 (December 5, 1985) – (English Version and Japanese Version Submitted)	
	D1407	Comer, "Translated by Jun Murai and Hiroyuki Kusumoto, "Internetworking with TCP/IP Vol. 1: Principles, Protocols, and Architecture, Third Edition," Japan Kyoritsu Shuppan Co., Ltd., First Edition, First Copy, p 161-193 (August 10, 1997) (English Version and Japanese Version Submitted)	
	D1408	Lynch et al., Supervisor of Translation: Jun Murai, "Internet System Handbook," Japan Impress Co. Ltd. First Edition p 152-157 and p 345-351 (August 11, 1996) (English Version and Japanese Version Submitted)	
	D1409	Office Action dated December 27, 2012 from Corresponding Canadian Patent Application Number 2723504	
	D1410	Office Action dated December 5, 2012 from Corresponding Japanese Patent Application Number 2011-081417	
	D1411	Office Action dated December 13, 2012 from Corresponding Japanese Patent Application Number 2011-085052	
	D1412	Office Action dated December 13, 2012 from Corresponding Japanese Patent Application Number 2011-083415	

EXAMINER	DATE CONSIDERED

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.
1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

Subst. for form 1449/PTO				**Complete if Known**	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT *(Use as many sheets as necessary)*				Application Number	**13/336,790**
				Filing Date	**12-23-2011**
				First Named Inventor	**Victor Larson**
				Art Unit	**2453**
				Examiner Name	**Krisna Lim**
				Docket Number	**77580-151(VRNK-0001CP3CNFT1)**

CERTIFICATION STATEMENT

This Information Disclosure Statement is being filed to replace the Information Disclosure Statement filed with the United States Patent and Trademark Office on February 15, 2013.

The following items were added:

1. Japanese Office Action dated December 13, 2012 from Corresponding Japanese Patent Application Number 2011-083415 has been added (DI412) to note where prior art reference C25 came from; and

2. The certification was added to note that the remaining references contained in the information disclosure statement were cited in a communication from a foreign patent office in a counterpart foreign application, and, to the was known to any individual designated in § 1.56(c) more than three months prior to the filing of the information disclosure statement.

Please See 37 CFR 1.97 and 1.98 to make the appropriate selection(s)

[ ] Information Disclosure Statement is being filed with the filing of the application or before the receipt of a first office action.

[X] That each item of information contained in the information disclosure statement was first cited in any communication from a foreign patent office in a counterpart foreign application not more than three months prior to the filing of the information disclosure statement; or; Cited reference A166 from Canadian office action dated December 27, 2012; Cited reference C25 from Japanese office action dated 12/13/12; Cited references C26, D1254 from Japanese office action dated 12/13/12; C27-C28, D1406-1408 from Japanese office action dated 12/05/12.

[X] That no item of information contained in the information disclosure statement was cited in a communication from a foreign patent office in a counterpart foreign application, and, to the knowledge of the person signing the certification after making reasonable inquiry, no item of information contained in the information disclosure statement was known to any individual designated in § 1.56(c) more than three months prior to the filing of the information disclosure statement. Cited references D1255-D1405 all received by the client on January 31, 2013.

[ ] The Commissioner is hereby authorized to charge any required fees to Deposit Account 50-1133.

[ ] Information Disclosure Statement is being filed with the Request for Continued Examination. The Commissioner is hereby authorized to charge the fee pursuant to 37 CFR 1.17(P) in the amount of $810.00, or further fees which may be due, to Deposit Account 50-1133.

SIGNATURE

A signature of the applicant or representative is required in accordance with CFR 1.33, 10.18. Please see CFR 1.4(d) for the form of the signature.

/Toby H. Kusmer/ Date: February 19, 2013
Toby H. Kusmer; Reg. No.:26,418
McDermott Will & Emery LLP
28 State Street
Boston, MA 02109
Tel. (617) 535-4000
Fax (617) 535-3800

DM_US 41241724-1.077580.0151

Electronic Acknowledgement Receipt

EFS ID:	14986704
Application Number:	13336790
International Application Number:	
Confirmation Number:	6217
Title of Invention:	SYSTEM AND METHOD EMPLOYING AN AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES
First Named Inventor/Applicant Name:	Victor Larson
Customer Number:	23630
Filer:	Toby H. Kusmer./Kerrie Jones
Filer Authorized By:	Toby H. Kusmer.
Attorney Docket Number:	77580-151(VRNK-1CP3CNFT1)
Receipt Date:	19-FEB-2013
Filing Date:	23-DEC-2011
Time Stamp:	12:03:43
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Information Disclosure Statement (IDS) Form (SB08)	IDS.pdf	83150 2a87d42f09d19c9a78a60e975e72ff7ded0f156f	no	9

Warnings:

Information:

This is not an USPTO supplied IDS fillable form

2	Non Patent Literature	D1412.pdf	371629 702ab63954ec638403ee41f30cc00dde5b241fa7	no	4

Warnings:

Information:

Total Files Size (in bytes):	454779

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

<u>New Applications Under 35 U.S.C. 111</u>
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

<u>National Stage of an International Application under 35 U.S.C. 371</u>
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

<u>New International Application Filed with the USPTO as a Receiving Office</u>
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

US006286047B1

(12) **United States Patent**
Ramanathan et al.

(10) **Patent No.: US 6,286,047 B1**
(45) **Date of Patent: Sep. 4, 2001**

(54) METHOD AND SYSTEM FOR AUTOMATIC DISCOVERY OF NETWORK SERVICES

(75) Inventors: **Srinivas Ramanathan**, Sunnyvale; **Deborah L. Caswell**, Santa Clara, both of CA (US)

(73) Assignee: **Hewlett-Packard Company**, Palo Alto, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/151,134**

(22) Filed: **Sep. 10, 1998**

(51) **Int. Cl.**⁷ **G06F 15/16**

(52) **U.S. Cl.** **709/224**; 709/202; 709/217; 709/226; 345/329; 370/229; 370/254; 706/51; 707/501

(58) **Field of Search** 709/202, 203, 709/204, 205, 217, 218, 220, 224, 226, 227, 229; 345/329; 370/229, 230, 254, 258; 707/501; 706/51

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,185,860		2/1993	Wu	395/200
5,276,789		1/1994	Besaw et al.	395/200
5,485,579	*	1/1996	Hitz et al. .	
5,644,720	*	7/1997	Boll et al. .	
5,678,045	*	10/1997	Bettels .	
5,727,147	*	3/1998	Van Hoff .	
5,740,362	*	4/1998	Buickel et al. .	
5,758,083	*	5/1998	Singh et al. .	
5,774,689	*	6/1998	Curtis et al. .	
5,781,534	*	7/1998	Perlman et al. .	
5,781,743	*	7/1998	Matsuno et al. .	
5,790,548	*	8/1998	Sistanizadeh et al. .	
5,793,965	*	8/1998	Vanderbilt et al.	709/203
5,796,951	*	8/1998	Hamner et al. .	
5,802,291	*	9/1998	Balick et al.	709/202
5,805,820	*	9/1998	Bellovin et al. .	

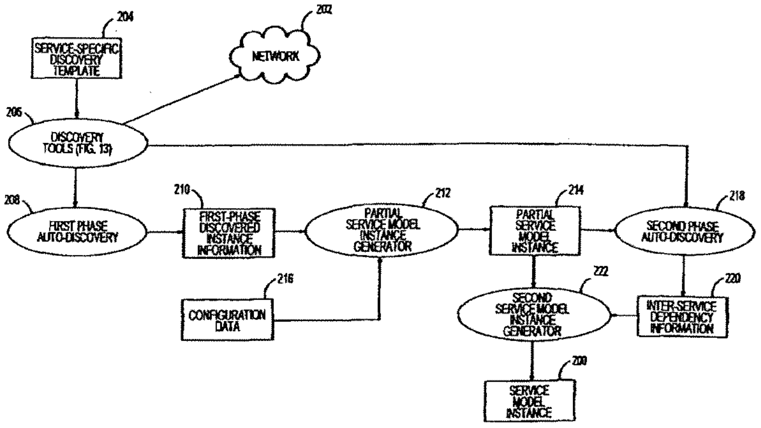
(List continued on next page.)

Primary Examiner—Robert B. Harrell
Assistant Examiner—Stephan Willett

(57) **ABSTRACT**

A method for identifying services, service elements and dependencies among the services and service elements includes executing first and second phases of discovery. In the first phase, the services and service elements are detected, as well as a first set of dependencies. The second phase is based on results of the first phase and is focused upon detecting inter-service dependencies, i.e., conditions in which proper operation of one service relies upon at least one other service. Various techniques may be used in executing the first phase, including accessing information in a domain name service (DNS) of the network to identify dependencies, as well as services and service elements. Discovery within the first phase may also be based upon recognizing naming conventions. Regarding the second phase, one approach to discovering inter-service dependencies is to deploy discovery agents implemented in computer software to access content of configuration files of applications detected in the first phase. Discovery agents may also be used to monitor connections completed via specified service elements detected in the first phase, such that other inter-service dependencies are identified. As an alternative or additional approach, network probes may be deployed to access information of data packets transmitted ted between service elements detected in the first phase, with the accessed packet information being used to detect inter-service dependencies. When information of the DNS is accessed in the first phase, the information is used as a basis for determining at least some of (1) groups of service elements that are generally equivalent with respect to executing a particular service within the network, (2) hosts supporting virtual hosting, (3) hosts supporting virtual servers, and (4) name servers.

20 Claims, 13 Drawing Sheets


204
SERVICE-SPECIFIC DISCOVERY TEMPLATE
202
NETWORK
206
DISCOVERY TOOLS (FIG. 13)
208
FIRST PHASE AUTO-DISCOVERY
210
FIRST-PHASE DISCOVERED INSTANCE INFORMATION
212
PARTIAL SERVICE MODEL INSTANCE GENERATOR
214
PARTIAL SERVICE MODEL INSTANCE
218
SECOND PHASE AUTO-DISCOVERY
216
CONFIGURATION DATA
222
SECOND SERVICE MODEL INSTANCE GENERATOR
220
INTER-SERVICE DEPENDENCY INFORMATION
200
SERVICE MODEL INSTANCE

U.S. PATENT DOCUMENTS

5,812,771	*	9/1998	Fee et al. .	
5,835,718	*	11/1998	Blewett	709/218
5,854,901	*	12/1998	Cole et al. .	
5,862,339	*	1/1999	Bonnaure et al.	709/227
5,867,667	*	2/1999	Butman et al. .	
5,870,545	*	2/1999	Davis et al. .	
5,884,033	*	3/1999	Duvall et al. .	
5,913,041	*	6/1999	Ramanathan et al. .	
5,926,463	*	7/1999	Ahearn et al.	370/254
5,944,783	*	8/1999	Nieten	709/202
5,946,464	*	8/1999	Kito et al.	709/202
5,958,012	*	9/1999	Battat et al.	709/224
5,958,052	*	9/1999	Bellovin et al. .	
5,968,116	*	10/1999	Day, II et al.	709/202
5,978,594	*	11/1999	Bonnell et al. .	
5,983,233	*	11/1999	Potonniee .	
5,999,940	*	12/1999	Ranger .	
6,009,467	*	12/1999	Ratcliff et al.	709/220
6,012,066	*	1/2000	Discount et al. .	
6,014,686	*	11/2000	Elnozahy et al.	709/202
6,044,224	*	3/2000	Radia et al. .	
6,046,988	*	4/2000	Schenkel et al.	370/254
6,054,987	*	4/2000	Richardson .	
6,055,562	*	4/2000	Devarakonda et al.	709/202
6061721	*	5/2000	Ismael et al. .	
6,063,129	*	5/2000	Bentley et al. .	

* cited by examiner

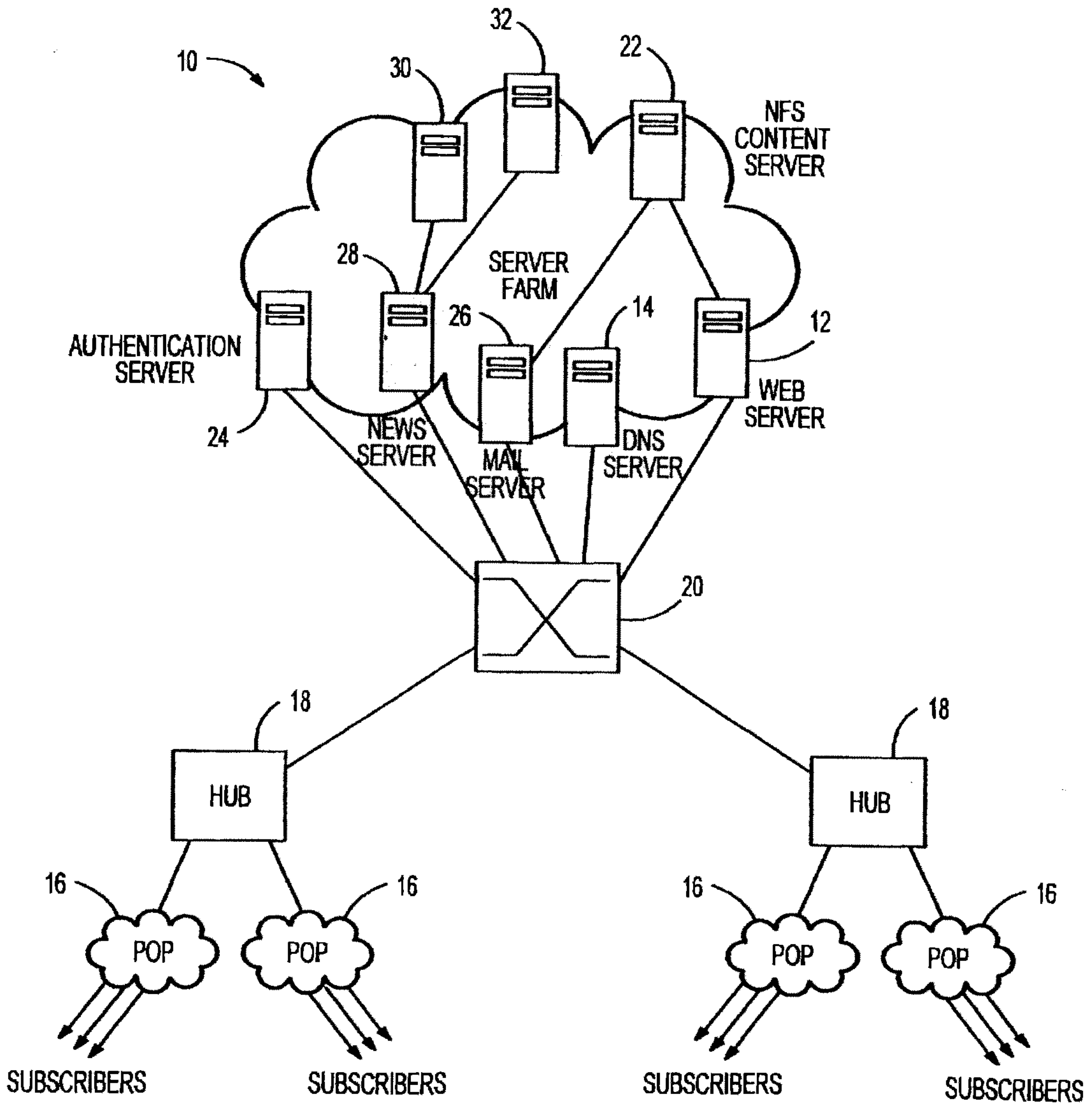


FIG. 1
(PRIOR ART)

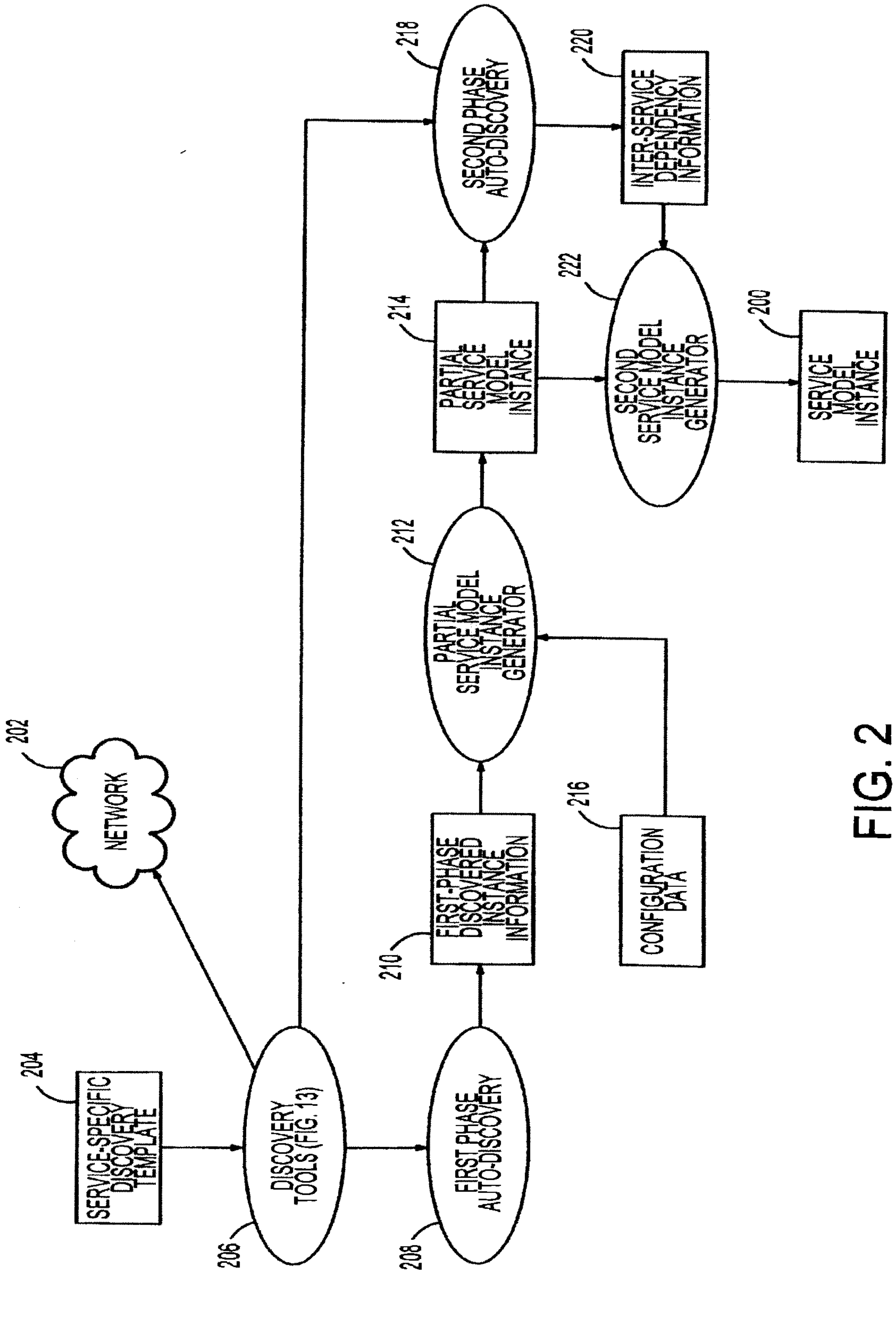

SERVICE-SPECIFIC DISCOVERY TEMPLATE
204
NETWORK
202
DISCOVERY TOOLS (FIG. 13)
206
FIRST PHASE AUTO-DISCOVERY
208
FIRST-PHASE DISCOVERED INSTANCE INFORMATION
210
CONFIGURATION DATA
216
PARTIAL SERVICE MODEL INSTANCE GENERATOR
212
PARTIAL SERVICE MODEL INSTANCE
214
SECOND PHASE AUTO-DISCOVERY
218
INTER-SERVICE DEPENDENCY INFORMATION
220
SECOND SERVICE MODEL INSTANCE GENERATOR
222
SERVICE MODEL INSTANCE
200

FIG. 2

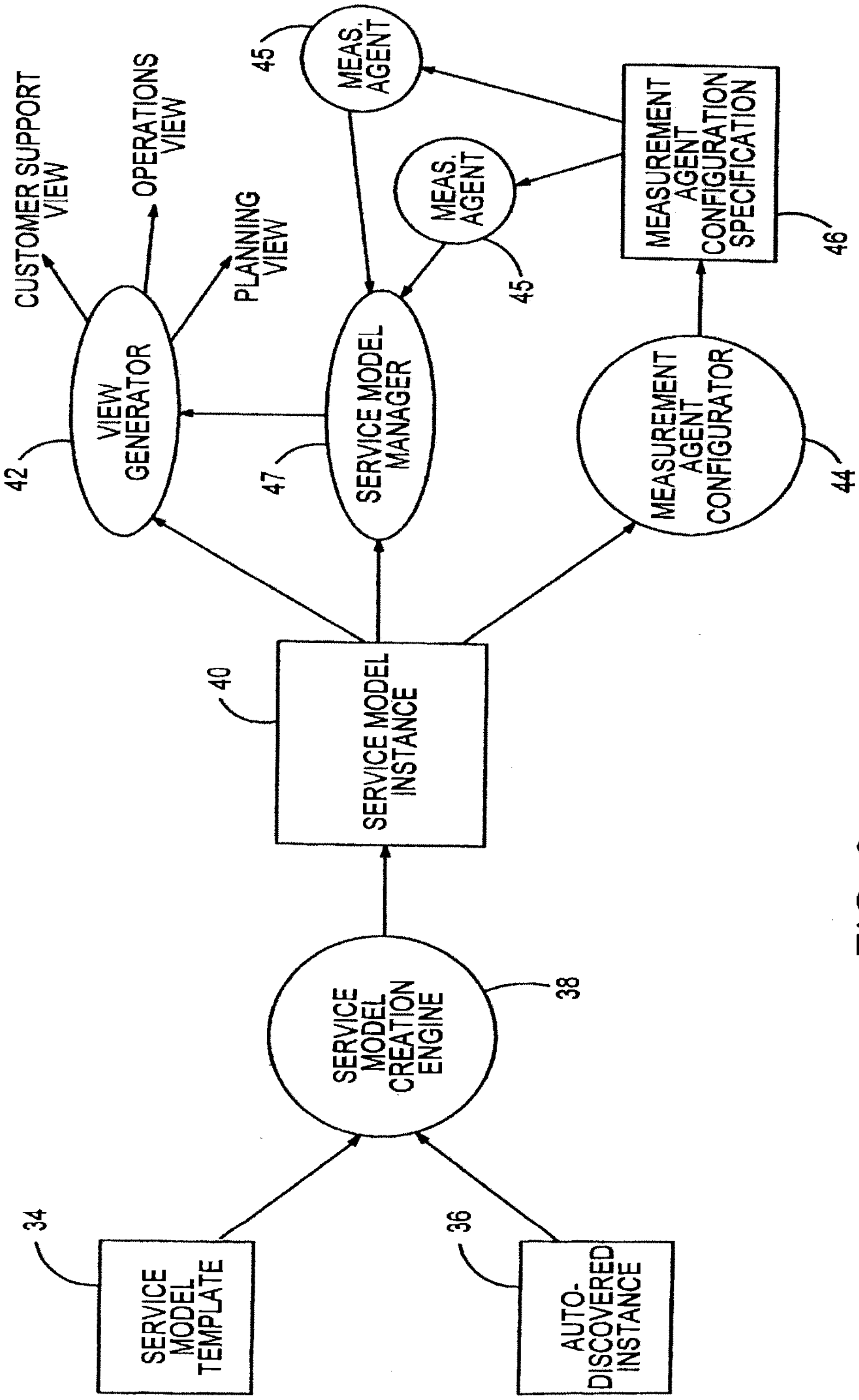

SERVICE MODEL TEMPLATE
34
AUTO-DISCOVERED INSTANCE
36
SERVICE MODEL CREATION ENGINE
38
SERVICE MODEL INSTANCE
40
VIEW GENERATOR
42
CUSTOMER SUPPORT VIEW
OPERATIONS VIEW
PLANNING VIEW
SERVICE MODEL MANAGER
47
MEAS. AGENT
45
MEAS. AGENT
45
MEASUREMENT AGENT CONFIGURATOR
44
MEASUREMENT AGENT CONFIGURATION SPECIFICATION
46

FIG. 3

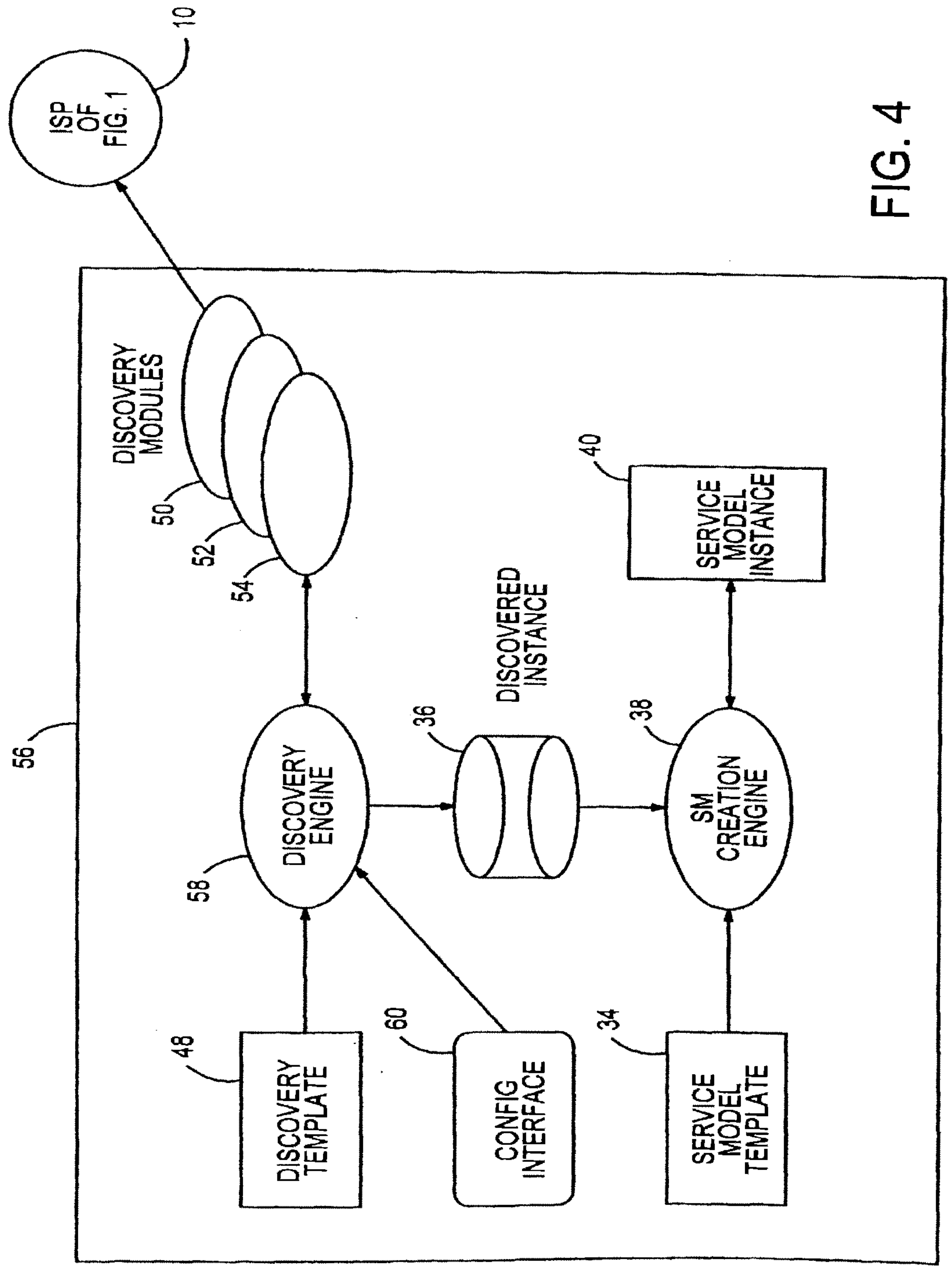

ISP OF FIG. 1
10
DISCOVERY MODULES
50
52
54
56
DISCOVERY ENGINE
58
36
DISCOVERED INSTANCE
SM CREATION ENGINE
38
SERVICE MODEL INSTANCE
40
DISCOVERY TEMPLATE
48
CONFIG INTERFACE
60
SERVICE MODEL TEMPLATE
34

FIG. 4

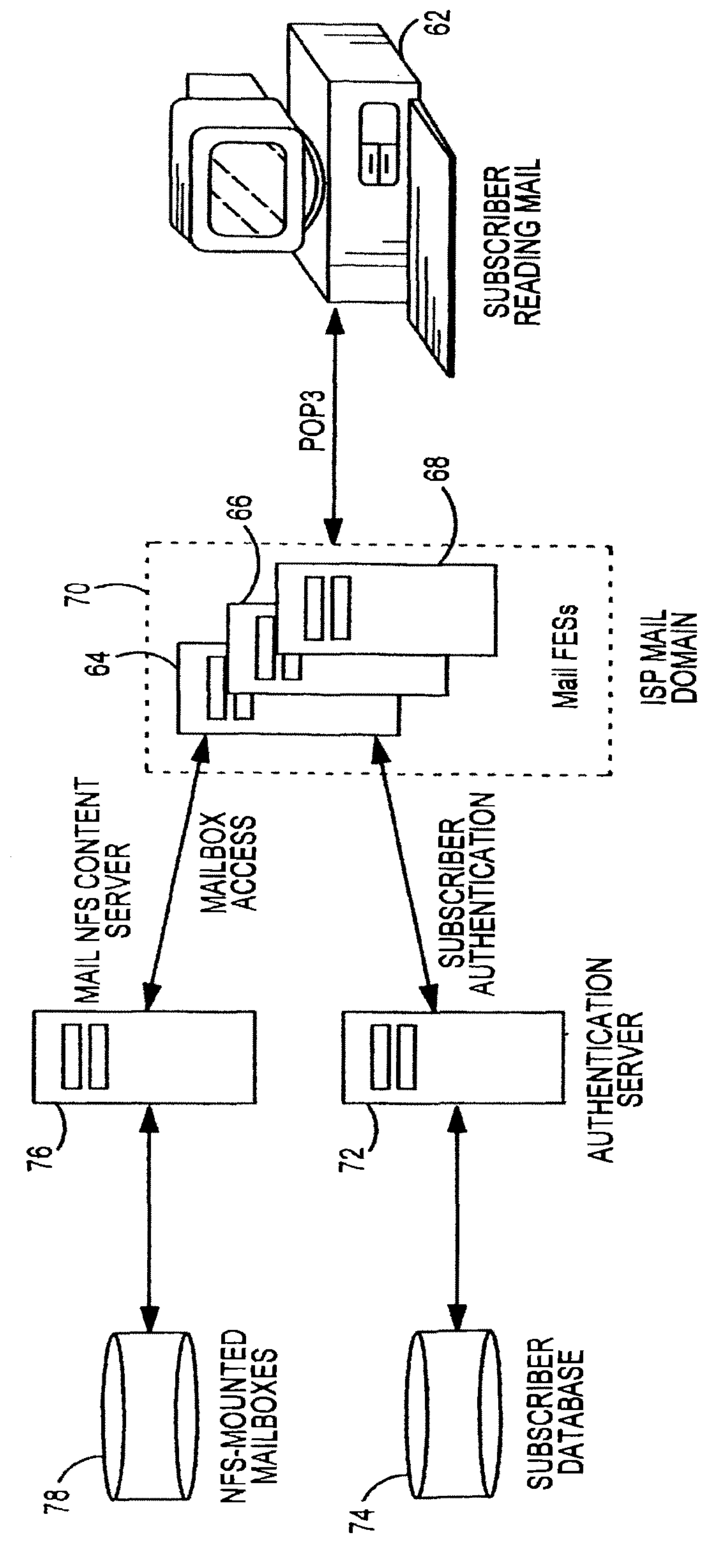

62
SUBSCRIBER READING MAIL
POP3
66
68
70
64
Mail FESs
ISP MAIL DOMAIN
MAIL NFS CONTENT SERVER
MAILBOX ACCESS
SUBSCRIBER AUTHENTICATION
76
72
AUTHENTICATION SERVER
78
NFS-MOUNTED MAILBOXES
74
SUBSCRIBER DATABASE

FIG. 5

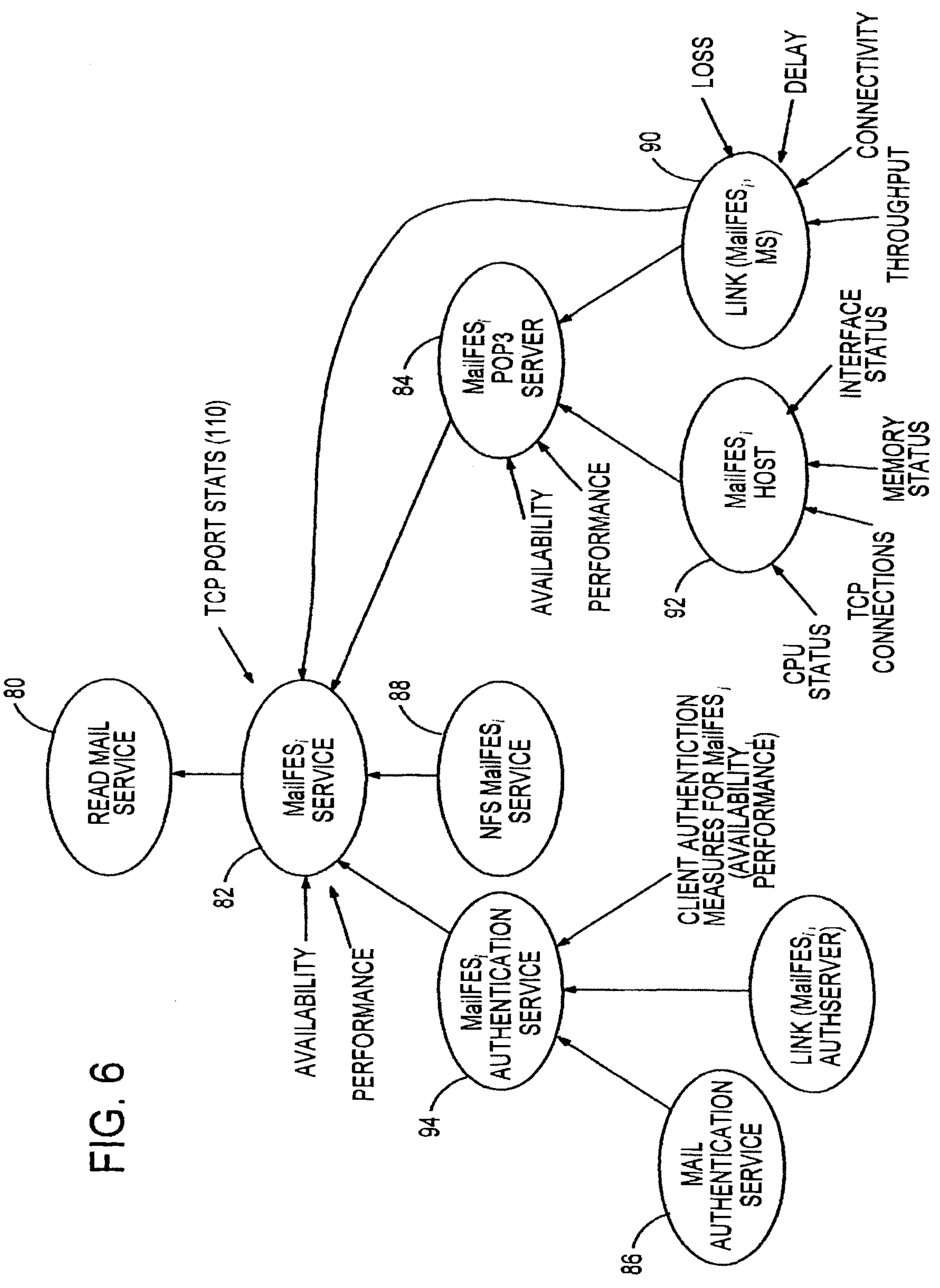

READ MAIL SERVICE
80
MailFES$_i$ SERVICE
82
TCP PORT STATS (110)
AVAILABILITY
PERFORMANCE
NFS MailFES$_i$ SERVICE
88
MailFES$_i$ POP3 SERVER
84
AVAILABILITY
PERFORMANCE
LINK (MailFES$_i$, MS)
90
LOSS
DELAY
CONNECTIVITY
THROUGHPUT
MailFES$_i$ HOST
92
INTERFACE STATUS
MEMORY STATUS
TCP CONNECTIONS
CPU STATUS
MailFES$_i$ AUTHENTICATION SERVICE
94
CLIENT AUTHENTICTION MEASURES FOR MailFES$_i$ (AVAILABILITY, PERFORMANCE)
LINK (MailFES$_i$, AUTHSERVER)
MAIL AUTHENTICATION SERVICE
86

FIG. 6

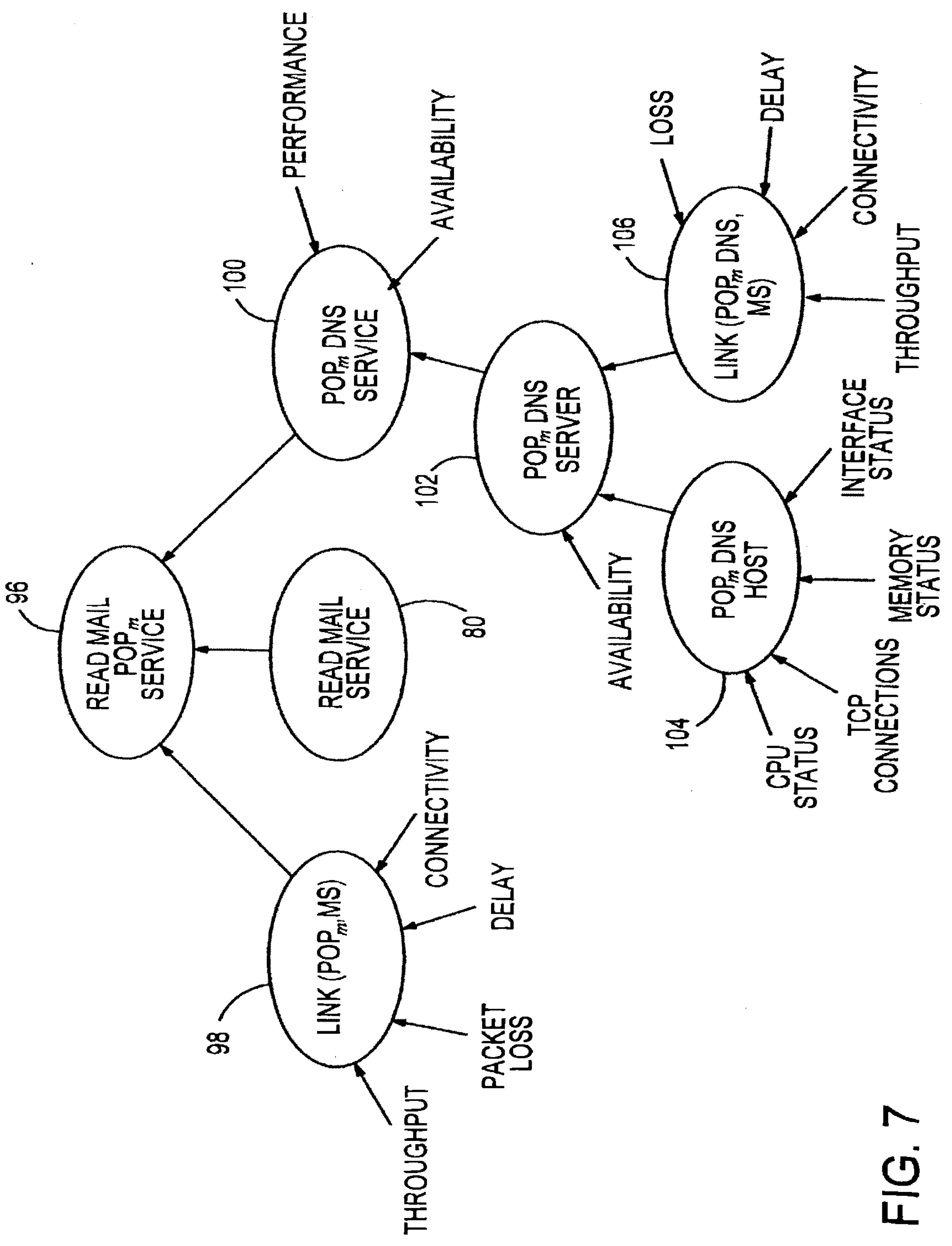

PERFORMANCE
AVAILABILITY
100
POP$_m$ DNS SERVICE
LOSS
DELAY
CONNECTIVITY
106
LINK (POP$_m$ DNS, MS)
THROUGHPUT
POP$_m$ DNS SERVER
102
INTERFACE STATUS
POP$_m$ DNS HOST
MEMORY STATUS
AVAILABILITY
TCP CONNECTIONS
104
CPU STATUS
96
READ MAIL POP$_m$ SERVICE
READ MAIL SERVICE
80
CONNECTIVITY
DELAY
LINK (POP$_m$,MS)
98
PACKET LOSS
THROUGHPUT

FIG. 7

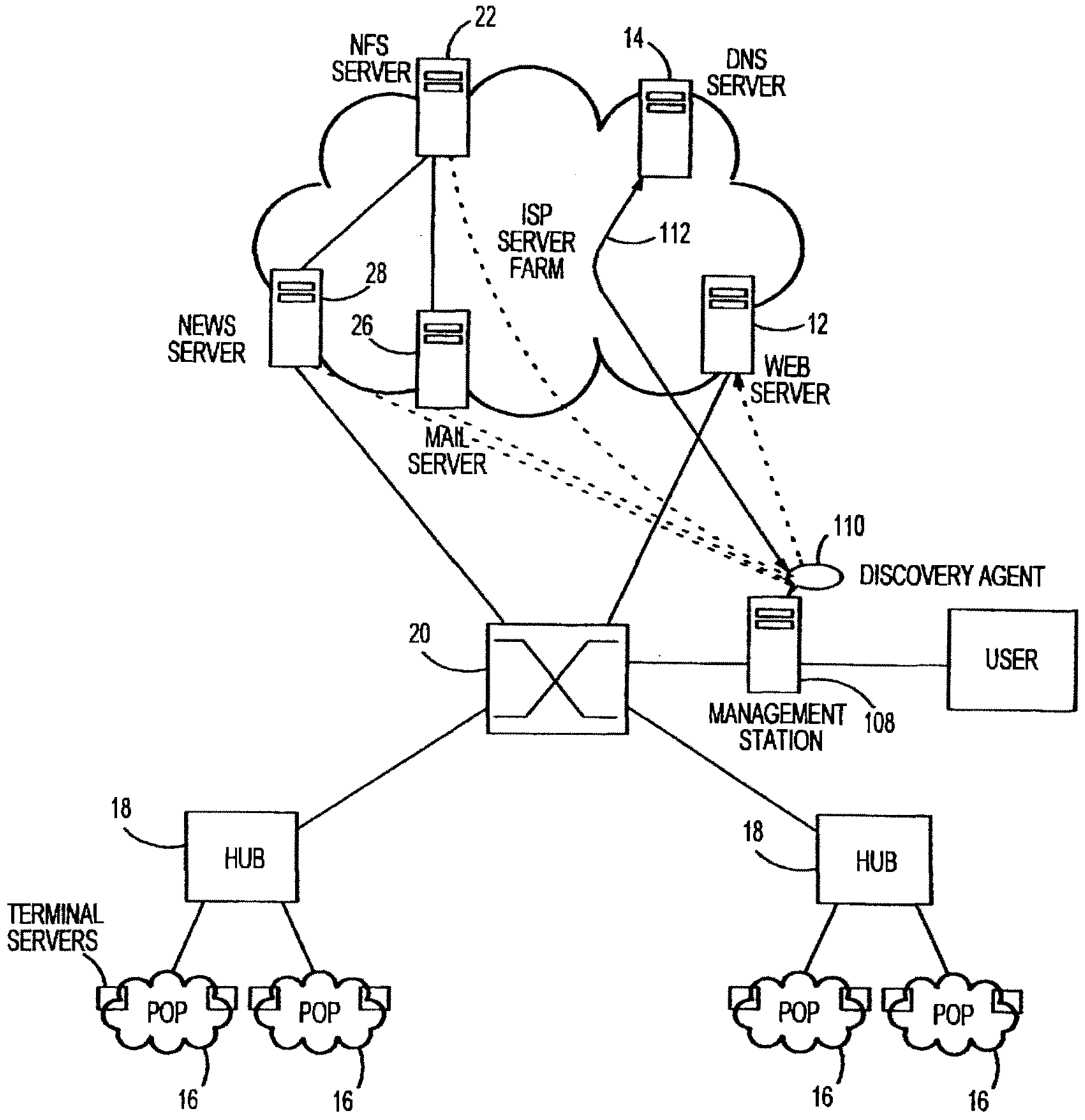

NFS SERVER
22
14
DNS SERVER
ISP SERVER FARM
112
28
NEWS SERVER
26
12
WEB SERVER
MAIL SERVER
110
DISCOVERY AGENT
20
USER
MANAGEMENT STATION
108
18
HUB
18
HUB
TERMINAL SERVERS
POP
POP
POP
POP
16
16
16
16

FIG. 8

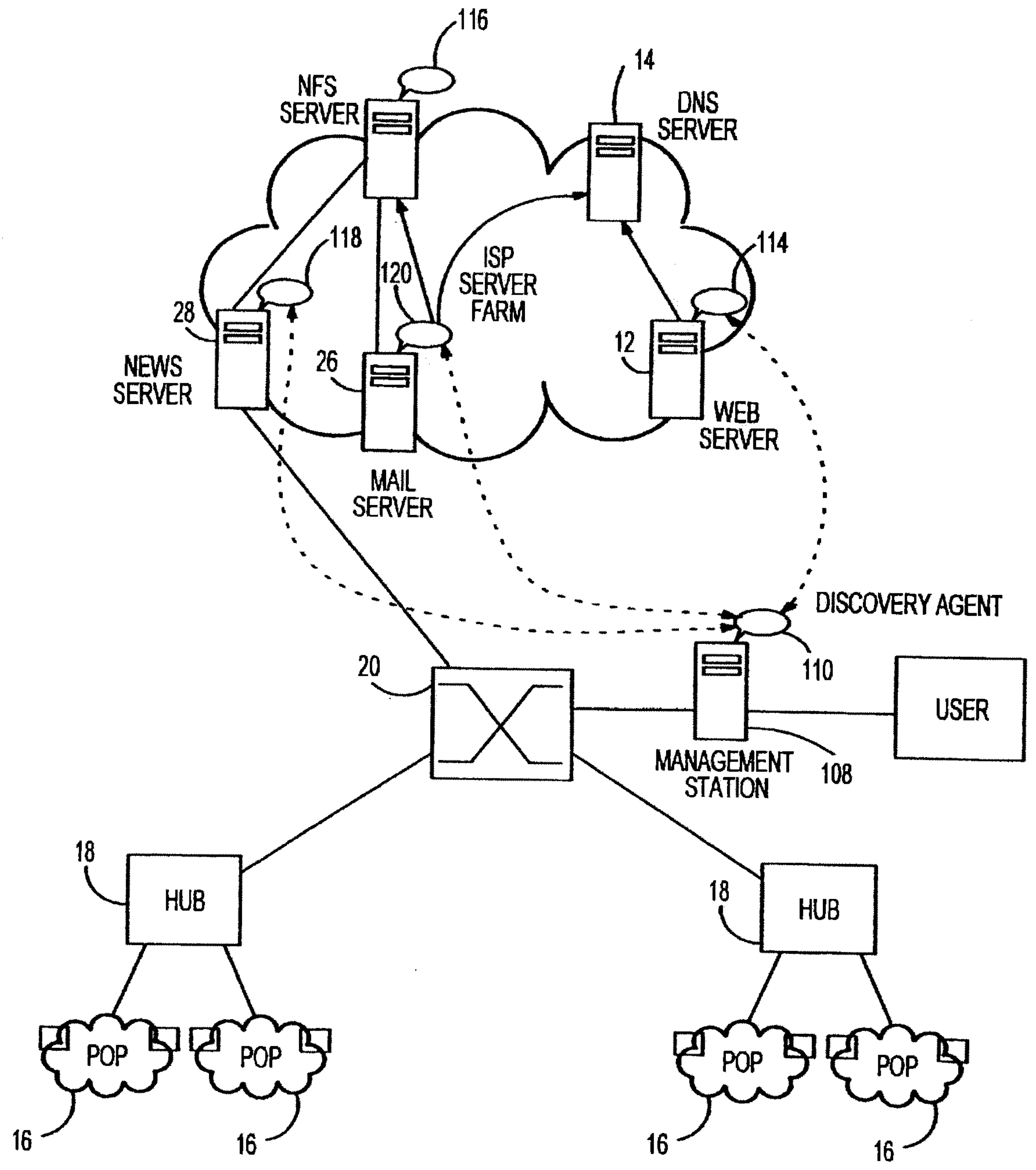

116
NFS SERVER
14
DNS SERVER
118
120
ISP SERVER FARM
114
28
NEWS SERVER
26
12
WEB SERVER
MAIL SERVER
DISCOVERY AGENT
110
20
USER
MANAGEMENT STATION
108
18
HUB
18
HUB
POP
POP
POP
POP
16
16
16
16

FIG. 9

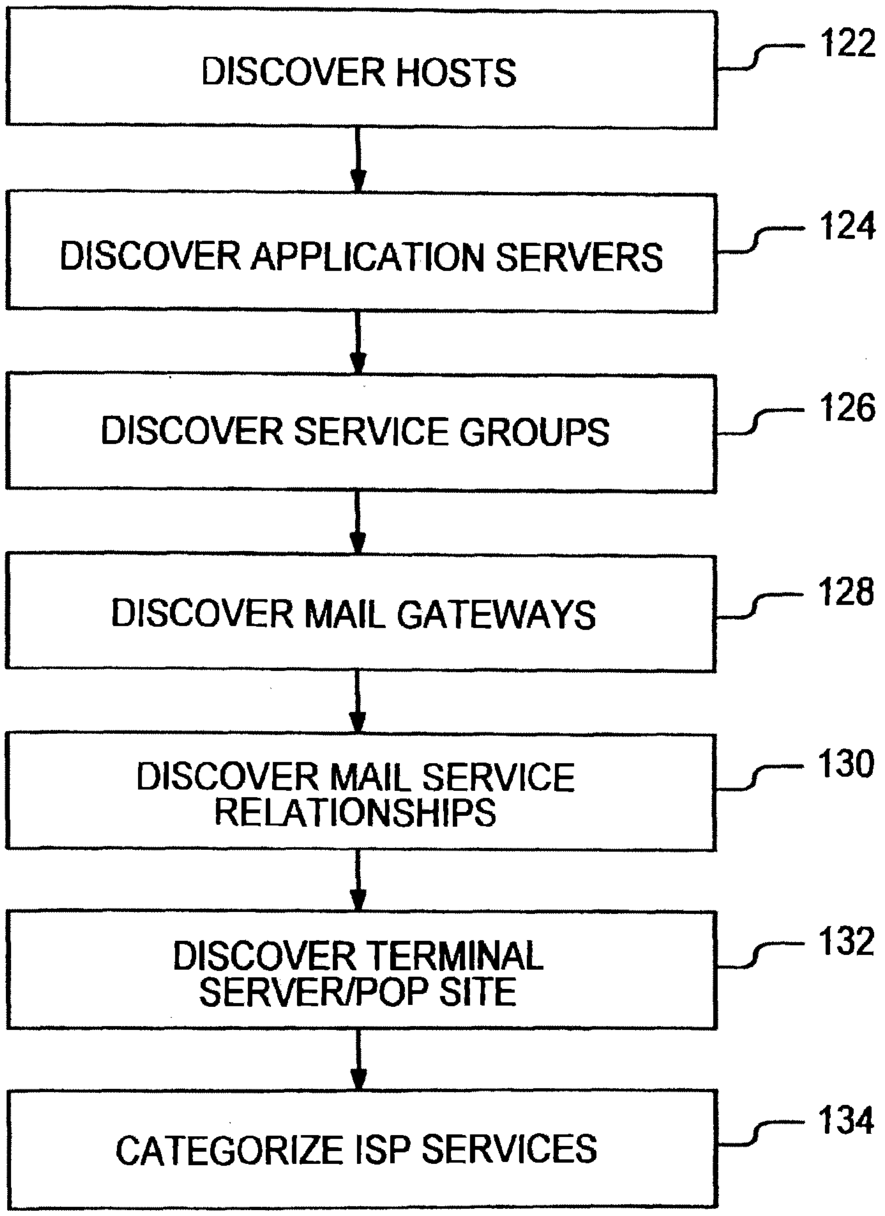

DISCOVER HOSTS
122
DISCOVER APPLICATION SERVERS
124
DISCOVER SERVICE GROUPS
126
DISCOVER MAIL GATEWAYS
128
DISCOVER MAIL SERVICE RELATIONSHIPS
130
DISCOVER TERMINAL SERVER/POP SITE
132
CATEGORIZE ISP SERVICES
134

FIG. 10

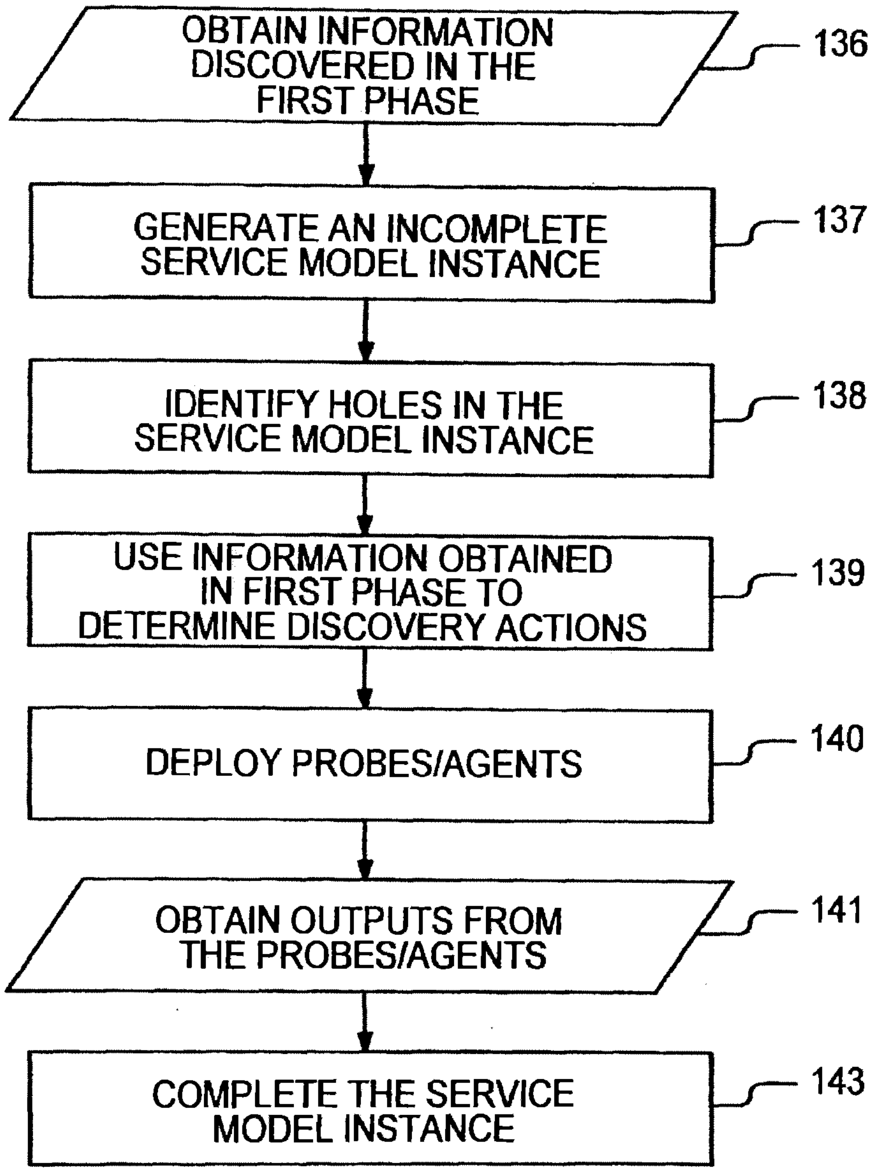

OBTAIN INFORMATION DISCOVERED IN THE FIRST PHASE
136
GENERATE AN INCOMPLETE SERVICE MODEL INSTANCE
137
IDENTIFY HOLES IN THE SERVICE MODEL INSTANCE
138
USE INFORMATION OBTAINED IN FIRST PHASE TO DETERMINE DISCOVERY ACTIONS
139
DEPLOY PROBES/AGENTS
140
OBTAIN OUTPUTS FROM THE PROBES/AGENTS
141
COMPLETE THE SERVICE MODEL INSTANCE
143

FIG. 11

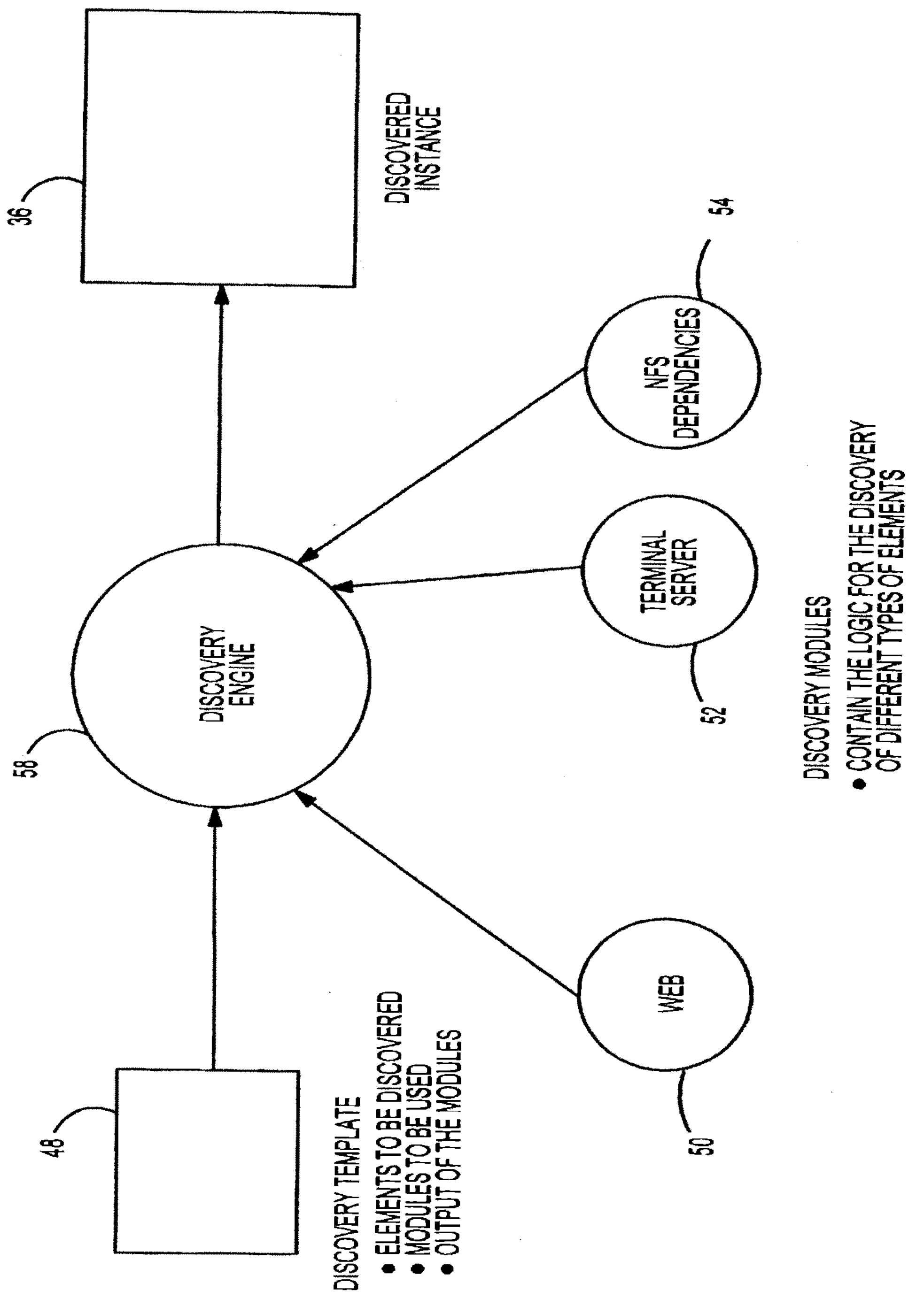

36
DISCOVERED INSTANCE
58
DISCOVERY ENGINE
48
DISCOVERY TEMPLATE
• ELEMENTS TO BE DISCOVERED
• MODULES TO BE USED
• OUTPUT OF THE MODULES
WEB
50
TERMINAL SERVER
52
NFS DEPENDENCIES
54
DISCOVERY MODULES
• CONTAIN THE LOGIC FOR THE DISCOVERY OF DIFFERENT TYPES OF ELEMENTS

FIG. 12

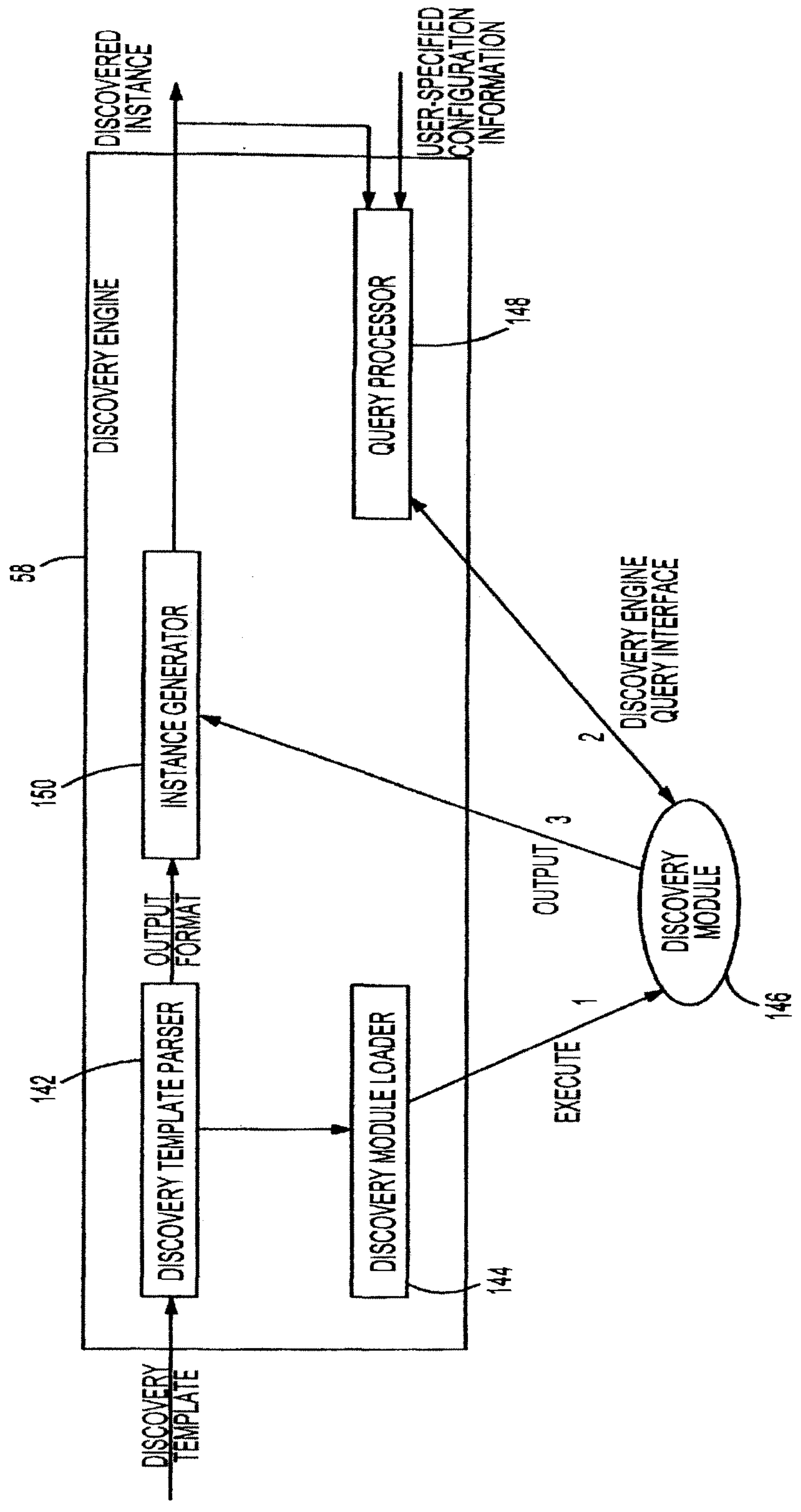

DISCOVERY TEMPLATE
DISCOVERY ENGINE
58
DISCOVERY TEMPLATE PARSER
142
DISCOVERY MODULE LOADER
144
OUTPUT FORMAT
INSTANCE GENERATOR
150
QUERY PROCESSOR
148
DISCOVERED INSTANCE
USER-SPECIFIED CONFIGURATION INFORMATION
EXECUTE
1
OUTPUT
3
2
DISCOVERY ENGINE QUERY INTERFACE
DISCOVERY MODULE
146

FIG. 13

METHOD AND SYSTEM FOR AUTOMATIC DISCOVERY OF NETWORK SERVICES

TECHNICAL FIELD

The invention relates generally to methods and systems for discovering service elements that enable services available via a network, as well as for discovering interrelationships among the services. Such information may be used for constructing models of services, permitting network personnel to assess the health of the service and to diagnose problems associated with the service.

BACKGROUND ART

Originally, computer networks were designed to have a centralized, network topology. In such a topology, a centralized mainframe computer is accessed by users at computer terminals via network connections. Applications and data are stored at the mainframe computer, but may be accessed by different users. However, a current trend in network design is to provide a topology that enables distributed processing and peer-to-peer communications. Under this topology, network processing power is distributed among a number of network sites that communicate on a peer-to-peer level. Often, there are a number of servers within the network and each server is accessible by a number of clients. Each server may be dedicated to a particular service, but this is not critical. Servers may communicate with one another in providing a service to a client.

Networks vary significantly in scope. A local area network (LAN) is limited to network connectivity among computers that are in close proximity, typically less than one mile. A metropolitan area network (MAN) provides regional connectivity, such as within a major metropolitan area. A wide area network (WAN) links computers located in different geographical areas, such as the computers of a corporation having business campuses in a number of different cities. A global area network (GAN) provides connectivity among computers in various nations. The most popular GAN is the network commonly referred to as the Internet.

The decentralization of computer networks has increased the complexity of tracking network topology. The network components (i.e., "nodes") may be linked in any one of a variety of schemes. The nodes may include servers, hubs, routers, bridges, and the hardware for linking the various components. Systems for determining and graphically displaying the topology of a computer network are known. U.S. Pat. Nos. 5,276,789 to Besaw et al. and 5,185,860 to Wu, both of which are assigned to the assignee of the present invention, describe such systems. As described in Besaw et al., the system retrieves a list of nodes and their interconnections from a database which can be manually built by a network administrator or automatically constructed using computer software. The system can be configured to provide any one of three views. An internet view shows nodes and interconnections of different networks. A network view shows the nodes and interconnections of a single network within the internet view. A segment view displays nodes connected within one segment of one of the networks. Selected nodes on the network, called discovery agents, can convey knowledge of the existence of other nodes. The network discovery system queries these discovery agents and obtains the information necessary to form a graphical display of the topology. The discovery agents can be periodically queried to determine if nodes have been added to the network. In a Transmission Controller Protocol/Internet Protocol (TCP/IP) network, the discovery agents are nodes

that respond to queries for an address translation table which translates Internet Protocol (IP) addresses to physical addresses.

The Besaw et al. and Wu systems operate well for graphically displaying hardware components and hardware connections within a network. From this information, a number of conclusions can be drawn regarding the present capabilities and future needs of the network. However, these systems do not discover services, their elements and interdependencies. Moreover, the interdependencies of the components in providing a particular service are not apparent from the graphical display that is presented by the system. The complexities of such interdependencies continue to increase in all networks, particularly the Internet. Moreover, these systems are designed in a monolithic manner. This does not allow the management system to be extended to discover and manage new service elements or new services.

Another approach is described by J. L. Hellerstein in an article entitled "A Comparison of Techniques for Diagnosing Performance Problems in Information Systems: Case Study and Analytic Models," *IBM Technical Report*, September, 1994. Hellerstein proposes a measurement navigation graph (MNG) in which network measurements are represented by nodes and the relationships between the measurements are indicated by directed arcs. The relationships among measurements are used to diagnose problems. However, the approach has limitations, since MNGs only represent relationships among measurements. An ISP operator must understand the details of the measurements (when, where, and how each measurement is performed) and their relationships to the different service elements. This understanding is not readily available using the MNG approach. Automatic discovery capabilities do not exist in this system to discover relationships among measurements.

The emergence of a variety of new services, such as World Wide Web (WWW) access, electronic commerce, multimedia conferencing, telecommuting, and virtual private network services, has contributed to the growing interest in network-based services. However, the increasing complexity of the services offered by a particular network causes a reduction in the number of experts having the domain knowledge necessary to diagnose and fix problems rapidly. Within the Internet, Internet Service Providers (ISPs) offer their subscribers a number of complex services. An ISP must handle services that involve multiple complex relationships, not only among their service components (e.g., application servers, hosts, and network links), but also within other services. One example is the web service. This service will be described with reference to FIG. 1. Although it may appear to a subscriber of the ISP 10 that the web service is being exclusively provided by a web application server 12, there are various other services and service elements that contribute to the web service. For instance, to access the web server 12, a Domain Name Service (DNS) server 14 is accessed to provide the subscriber with the IP address of the web site. The access route includes one of the Points of Presence (POP) 16, a hub 18, and a router 20. Each POP houses modem banks, telco connections, and terminal servers. A subscriber request is forwarded to and handled by a web server application. The web page or pages being accessed may be stored on a back-end Network File System (NFS) 22, from which it is delivered to the web server on demand. When the subscriber perceives a degradation in the Quality of Service (QoS), the problem may be due to any of the web service components (e.g., the web application server 12, the host machine on which the web application server is executing, or the network links interconnecting the subscriber to the web server), or may be due to the other infrastructure services on which the web service depends (e.g., DNS or NFS). The ISP system **10** of FIG. **1** is also shown to include an authentication server 24 for performing a subscriber authentication service, a mail server **26** for enabling email service (for login and email access), and front-end and back-end servers **28**, **30** and **32** for allowing Usenet access.

Subscribers demand that ISPs offer reliable, predictable services. To meet the expectations of subscribers and to attract new subscribers, ISPs must measure and manage the QoS of their service offerings. This requires a variety of tools that monitor and report on service-level metrics, such as availability and performance of the services, and that provide health reports on the individual service components. Unfortunately, the majority of management systems have not kept pace with the service evolution. Available management systems lack the capability to capture and exploit the inter-relationships that exist among services available in a network environment, such as the Internet. Typically, these management systems discover and manage service elements in isolation. Moreover, these systems are implemented in a monolithic manner and are not easily extensible to discovering and managing new network services and service elements. Adding new discovery and management capabilities to these systems requires extensive redesign and modification of the management system.

Each network is unique in various respects, such as the configuration of servers, the types of application servers, the service offerings, the organizational topology, and the inter-service dependencies. Therefore, in order to accurately understand the operations of the network, specific models must be crafted for the services provided within the network. However, handcrafting models of network services requires an enormous effort on the part of a human expert or group of experts.

What is needed is a comprehensive method and system that discovers services and service elements of a network, and discovers the dependencies among the services and service elements. This information can then be used to automatically generate models of the discovered services.

SUMMARY OF THE INVENTION

A method of identifying service elements, services and dependencies among the elements and services of a network includes executing a two-phase discovery process. In the first phase of discovery, the services and service elements are detected, as well as a first set of dependencies. The second phase of discovery is focused upon the dependencies, specifically inter-service dependencies in which one discovered service is reliant upon one or more other discovered services.

The term "service" is defined herein as "a functionality offered by a network to a user to perform a specific set of tasks." Performance of the service involves a number of cooperating service elements, such as network routers, servers, applications and the like. Services include application services (such as web and email access) and network services (such as a Virtual Private Network).

The first phase of discovery may be considered as a "black-box approach." At the beginning of the phase, the system is likely to have no information regarding the services and service elements that exist in a network, such as an Internet Service Provider (ISP). Using a variety of techniques, this first phase obtains information relating to the services and service elements in the network. Execution,

component, organizational and some inter-service dependencies are discovered during the first phase. Since a black-box approach is adopted, this first phase of discovery can be executed from a management station that is separate from elements of the network being discovered. For example, in an ISP environment, subject to the availability of a network connection from the management station to the server farm of the ISP, this first phase can be executed when the management station is outside of the server farm. Thus, this first phase may be referred to as "external discovery." However, it should be understood that the first phase can be executed exclusively within the network of interest.

The second phase of discovery is targeted at identifying inter-service dependencies. This phase uses the knowledge of the services and service elements obtained in the first phase, and targets the discovered services to obtain dependency information relating to the services. This dependency information likely requires direct access to the elements of the network. Therefore, the second phase may be considered to be "internal discovery." The step of executing the second phase to identify the dependencies is a software-based automated process. The discovery system may include discovery agents that are configured to access the content of configuration files of applications that were detected in the first phase of discovery. As an example of processing the content of a configuration file to discover inter-service dependencies, a configuration file of a web server may be accessed to discover whether the web server has a dependency on an NFS service. Discovery agents may also be deployed to monitor connections completed via service elements that were detected in the first phase of discovery. Information received during the monitoring may be utilized to identify inter-service dependencies. For example, discovery agents may be deployed by a discovery engine of the system to identify Transmission Control Protocol (TCP) connections of at least one host that was detected in the first phase. This technique exploits the fact that most TCP implementations enforce a three-minute delay for connections in a TIME_WAIT state of TCP, so that a connection persists for approximately three minutes after it is no longer in use.

As an additional or alternative source of information for the second phase, network probes may be deployed by the discovery engine to access information embedded within data packets transmitted by service elements detected in the first phase. Since most TCP/IP communication is based on source/destiny port numbers, by processing the headers of captured packets, a software probe can deduce many of the relationships that exist among services.

Returning to the first phase of identifying services and service elements, the domain name service (DNS) may be accessed to obtain information that specifies services, service elements, component dependencies, organizational dependencies, and some inter-service dependencies. The DNS of the network may be used to identify some, and preferably all, of: (1) any external name servers (organizational dependency); (2) round-robin service groups that are utilized to provide scalability and redundancy for the network (component dependency); (3) naming conventions that are employed by the network (organizational dependencies); (4) any external mail gateways of the network (organizational dependencies); and (5) any SMTP servers corresponding to hosts that run POP3 servers (inter-service dependencies).

The recognition of the naming conventions used by the network provides evidence of any virtual hosts or virtual servers. For example, an ISP may use a single host machine to support multiple customer web sites. While each customer web site may be associated with a unique IP address, there will be a naming pattern that identifies the common host machine. Naming conventions may also be used to recognize associations between terminal servers of an ISP and POP sites.

An advantage of the invention is that the discovery process may be used to automatically detect services, service elements and dependencies with regard to a selected core service of a network (e.g., Read Mail Service of an ISP). That is, the method and system may be used to model the selected core service, permitting network personnel to assess the health of the service and to diagnose problems associated with the service.

Another advantage is that the system is extensible to discover services or service elements that are added to a network. When a new service or service element is introduced, a user can add a new discovery module (e.g., a discovery agent) for the service or service element. The discovery engine is then able to discover instances of the added service or service element without any changes or enhancements to the discovery engine. This approach permits third-party discovery modules to be introduced into the system.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a schematic view of an exemplary prior art Internet Service Provider (ISP) system.

FIG. 2 is a schematic view of a process for modeling a selected service available via a network.

FIG. 3 is a block diagram of components of a system for modeling a selected service available via a network, such as the ISP of FIG. 1.

FIG. 4 is a block diagram of a management system having components of FIG. 3.

FIG. 5 is an exemplary layout of components for utilizing the Read Mail service of the ISP of FIG. 1.

FIG. 6 is a graph of nodes of a Read Mail service model configured using the system of FIG. 3.

FIG. 7 is an alternative Read Mail service model graph.

FIG. 8 is a schematic view of first phase internal discovery processing using the system of FIG. 3.

FIG. 9 is a schematic view of second phase external discovery processing using the system of FIG. 3.

FIG. 10 is a process flow of steps of the first phase discovery of FIG. 8.

FIG. 11 is a process flow of steps of the second phase discovery of FIG. 9.

FIG. 12 is a block diagram of the operation of the discovery engine of FIG. 4.

FIG. 13 is a block diagram of the discovery engine of FIG. 12.

BEST MODE FOR CARRYING OUT THE INVENTION

The invention relates to the discovery process for enabling automated detection of service elements and/or services that are utilized by a specific network to provide a particular service. One application of the invention is to model the particular service, so that dependencies among services and service elements (e.g., servers, hosts and network links) may be readily determined by network personnel. While the discovery process will be described primarily with reference to application to an ISP system, the process

has other applications. That is, the process may be used in other network environments (e.g., a corporation's network).

FIG. 2 is an overview of the auto-discovery process for generating a service model instance **200** for a particular network **202**. In FIG. 2, the rectangles represent data stores and the ellipses represent processing elements. A service-specific discovery template **204** is accessed upon initiation of the auto-discovery process. The discovery template defines the services and service elements that are anticipated as cooperating to provide the core service. Moreover, the discovery template identifies particular discovery tools (i.e., discovery modules and/or agents) that are to be used for each service and service element, as well as any dependencies that a particular discovery tool may have on outputs from other discovery tools. The template preferably includes instructions for configuring the outputs of the discovery tools. A more thorough description of the discovery template will follow, particularly with reference to Table 2, which is an example of a discovery template specification. A more thorough description of the discovery tools **206** will also be provided below, with reference to FIGS. **12** and **13**.

The auto-discovery approach is dividable into two phases. In the first phase auto-discovery procedure **208**, a "black-box approach" is adopted. Initially, no information about the services or service elements may be immediately available. Using a variety of techniques, information is obtained for the purpose of identifying relevant services, service elements, component dependencies, execution dependencies, organizational dependencies and some inter-service dependencies. In one of these techniques, the information that is available in the domain name system of an ISP is used to discover the existence and the relationships among different services and service elements. In order to allow subscribers to access a host using its host name, rather than requiring specification of an IP address that is more difficult to remember, DNS is used to map the host names of the IP addresses for all hosts in the ISP system. Moreover, the exchange of email messages across hosts occurs using the mail exchange records (MX records) maintained by the DNS. In summary, the domain name system stores a wealth of information that is used in the first phase of the auto-discovery process to discover Internet services and relationships among the services. Other techniques may be used in this first phase to supplement the information acquired from the domain name system. Since these techniques may be exploited separately from the servers of the network **202**, this first phase may be referred to as "external discovery." However, the process may occur entirely within the network environment.

Instance information **210** is acquired in the first phase procedure **208**. In addition to the identification of services and service elements, the instance information **210** identifies dependencies. There are a number of inter-dependencies of concern to the auto-discovery process. These inter-dependencies include execution dependencies, component dependencies, inter-service dependencies, and organizational dependencies. An execution dependency relates directly to an application server process being executed on a host machine. The types of application servers that are executed on host machines include web, email, news, DNS and NFS. A component dependency occurs in order to ensure scalability and redundancy of a service. For example, a web service may be provided collectively by a number of "front-end servers" (FESs), with round-robin DNS scheduling being used to provide subscribers with a domain name that maps to one of the FESs. ISPs often replicate web, email and news content across a number of servers. The round-robin scheduling balances the load among the servers. The servers are grouped together and assigned a single domain name in the DNS database. When the DNS server receives a request for the domain name, the IP address of one of the servers is acquired in the round-robin scheme.

An inter-service dependency occurs when one service accesses another service for its proper operation. For example, a web service depends on a DNS service to allow the subscriber to connect the web server host using its IP address, and an NFS service is used to access the web content. As another example, a Read Mail service depends on an authentication service to verify the identity of the subscriber, uses a DNS service to log the subscriber's IP address, and uses an NFS service to access the mail content.

Finally, an organization dependency occurs when there are different ISP operations personnel (e.g., subject matter experts) who are responsible for different services and service elements. For example, an ISP may have a first supervisor managing the web service, a second supervisor managing DNS, and a third supervisor managing NFS. Operational responsibilities may be delegated based upon the geographical location of the service elements. Since the precise organization structure may vary from one ISP to another, the auto-discovery mechanism provides a means by which it can be quickly customized for a particular ISP system.

The first phase auto-discovery procedure **208** provides discovered instance information **210** that identifies most of the execution, component and organization dependencies, as well as some of the inter-service dependencies. A partial service model instance generator **212** is then used to provide a partial service model instance **214**. Optionally, configuration data **216** may be used to allow an operator to customize a service model instance **200**. Using a configuration interface which will be described with reference to FIG. 4, the operator can specify categorization criteria for services and service elements. Thus, the service model instance **200** can be configured to meet the specific needs of the operator.

In a second phase auto-discovery procedure **218**, an "internal" view of the network **202** is acquired. The internal discovery of the second phase is intended to fill any "holes" in the partial service model **214**, and particularly focuses on identifying inter-service dependencies. There are two basic approaches to the internal discovery of the second phase. One approach is the use of network probes, which are implemented in software and are installed at strategic locations on the network to acquire information from headers of packet transmissions. By processing the headers of packets, a software probe can deduce many of the relationships that exist among servers. The second basic approach is to use special-purpose discovery agents that are installed on the ISP hosts to discover relationships among services. Rather than examining headers of packet transmissions, the special-purpose agents use a number of operating systems and application-specific mechanisms (such as processing service configuration information and application-dependent monitoring tables) to discover inter-service dependencies.

The inter-service dependency information **220** from the second phase auto-discovery procedure **218** is combined with the partial service model instance **214** using a second service model instance generator **222**. The output of the second service model instance generator is the completed service model instance **200**.

The invention will be described primarily in its preferred embodiment of using the discovery template to detect both services and service elements to form a service model. However, the discovery process may be used without the discovery template and without restricting the process to identifying only the services, service elements and dependencies relevant to a single core service.

One Embodiment for Using the Instance Information

As previously noted, the use of the discovery template **204** is optionally combined with a service model template in the process of generating the service model instance **200**. An overview of the template-driven procedure is illustrated in FIG. 3. The service model template is a generic specification of the service topology and measurement topology for the service of interest (e.g., Read Mail service). Depending on the service being modeled and the service elements that are likely to be involved, the template defines nodes of various types (e.g., hosts, servers, network links, and services) and their associated measurements. Moreover, the template indicates the dependencies among the nodes, such as the dependency of the service on other services (e.g., the Read Mail service which refers to a subscriber accessing his/her mailbox depends on the authentication and NFS services). In the preferred embodiment, the template also includes default state computation rules for specified nodes, so that the state (i.e., "health") of a node can be computed based upon measurements associated with the node and upon states of dependencies of the node.

The service model template **34** is not specific to any ISP system in the sense that it does not specifically reference any hosts, servers, network links, service groups, or other services or service elements in the ISP system. The template may be considered as a "lifeless tree." There is no association between nodes in the service model template and running elements, such as hosts and servers. However, the information contained in the service template for a node may include the element type, the element dependencies, the measurement definitions (e.g., agent to run, the format of the run string, the number and type of parameters, and the format of the output), default state computation rules, default thresholds, default baselines, and default alarm generation and correlation rules.

In database terminology, the service model template **34** is the schema. On the other hand, an instance defines the records in the database and the values of the static information. In FIG. 3, the discovered instance **36** is determined using auto-discovery, as will be explained fully below. Information regarding the services and service elements (e.g., servers, hosts and links) that exist in the ISP system or other service provider systems may be auto-discovered. The store representing auto-discovered information shall be referred to as the auto-discovered instance **36**.

The service model creation engine **38** of FIG. 3 is used to generate a service model instance **40** based on the service model template **34** and auto-discovered instance **36**. Ideally, all of the discovered information is available prior to instantiation. However, in many cases, discovery has to be performed in multiple phases, such as the two outlined above. In such cases, instantiation may occur partially after each phase of discovery. The main advantage of providing a partial service model instance is that the partially completed service model can provide a guide to identify the additional information needed in subsequent phases of discovery. The service model creation engine **38** encompasses the functions of the partial service model instance generator **212** and the second service model instance generator **222** of FIG. 2. Since new elements and services may be added to the ISP system over time, the service model instantiation process has to be repeated periodically to augment the initially created service model instance.

Unlike the service model template 34, the service model instance 40 is specific to an ISP system. The process of constructing the service model instance using the service model template and the auto-discovered instance 36 is referred to as the "instantiation" of the service model. As previously noted, the relationship between the service model template and the auto-discovered instance is analogous to the relationship between the schema and records in a database.

The service model instance 40 maps services and service elements that exist in a particular ISP system with nodes in the service model template. In doing so, the service model instance also specifies a set of measurements that must be made against each of the services and service elements in the specific ISP system.

The service model instance 40 may be used by a view generator 42. In the preferred embodiment, the service model instance is represented as a graph of the nodes and edges that identify dependencies of the nodes. Different management functions of an ISP will benefit from viewing different subsets of nodes and edges of the service model instance 40. Thus, the view generator 42 may be used by operations personnel to provide an "operations view" of only the services and service elements that fall in a particular domain of control of the personnel. On the other hand, a "customer support view" may provide an end-to-end view for the customer support personnel of the ISP. A "planning view" may be used to graphically present information about usage and performance trends, so that future requirements can be ascertained.

Even when different management functions are interested in the same subset of nodes and edges of the service model instance 40, there may be different interests with regard to rules to be used for state computations. For instances, a management application that visually depicts a color-coded hierarchical representation of the service model instance to operations personnel may require that hierarchical state propagation rules be employed in the service model instance. In this manner, viewing the state of the top-level nodes of the service model, an ISP operator can determine whether any problem has been detected in the ISP system. In contrast, an application that is responsible for automatically detecting, diagnosing and reporting the root-causes of problems in the ISP system may prefer not to use a hierarchical state propagation rule, since it is interested in the state of each node of the service model independently of the state of other nodes.

The view generator 42 may be used to define the subset of nodes of a service model instance that are of interest to a management application, as well as the method for evaluating the states of the nodes of interest. The measurements that are associated with the nodes in the service model instance are common across the different views. However, the rules that apply to computing the state of a particular node based upon relevant measurements and the states of dependent nodes may be different for the different views.

A measurement agent configurator 44 may be used to extract information from the service model instance 40, when the information is relevant to scheduling tests and retrieving test results. At the completion of the service model instance 40, static configuration information as to how the tests of various elements are to be run may be merged with default algorithm descriptions into a service model instance file. State computation rules, thresholds and alarm correlation rules defaults may be overridden using the measurement agent configurator. Specific measurements may be enabled or disabled. A final measurement agent specification 46 is generated by the configurator for the specific ISP. The measurements that are identified in the specification 46 are executed using software agents 45. The results of the measurements are analyzed and used by a service model manager 47 to compute the states of different nodes in the service model.

Overview of Instantation

FIG. 4 represents an overview of the process for generating the discovered instance that is used to form the service model instance 40 of FIG. 3. As previously noted, a requirement of the instantiation of a service model is the generation of the service model template for the particular service or services of interest. The template may be handcrafted, preferably by a domain expert of the service. The template includes a specification of the elements involved in providing the service, such as servers and network links, and the dependencies of the service on other services, such as the dependency of Read Mail service on the authentication and NFS services. As will be explained more fully below, the discovery process includes designation of a discovery template 48. The discovery template specifies the types of services and the service elements to be discovered. The discovery template also includes specifications of discovery modules 50, 52 and 54 to be invoked in the discovery of the specified services and service elements.

The service model template 34, the discovery template 48, and the discovery modules 50, 52 and 54 are utilized within a management system 56 for forming the service model instance 40. Another key component of the management system is a discovery engine 58 that processes information contained within the discovery template 48. The discovery engine invokes the appropriate discovery modules 50–54, interprets the outputs of the invoked modules, and stores the outputs in memory (or in a database) as the discovered instance 36, which is made accessible to the service model creation engine 38. The discovery engine 58 supports a configuration interface 60 that allows ISP operations personnel to control and customize the discovery process. Through the configuration interface 60, an ISP operator can restrict the discovery to specified IP address ranges or host name patterns. Furthermore, the operator can specify naming conventions used by the ISP (e.g., for naming terminal servers and POP sites), which are used by some of the discovery modules 50–54.

The configuration interface 60 serves as a way for an ISP operator to quickly customize the service model instance 40 that is generated by the process. Using the configuration interface, the operator can also specify categorization criteria for services and service elements. For instance, all the mail services could fall in one category, while the DNS servers could fall in another. The categories assigned to services and service elements can represent the organizational structure of the ISP, the geographical location of the servers offering the services (e.g., servers in San Francisco fall in one category, while servers in New York fall in another), or differences in business functions of the service (e.g., web servers that an ISP is hosting on behalf of business customers, as opposed to local web servers that the ISP offers for providing access to the dial-up subscribers).

In the preferred embodiment, the configuration interface 60 is implemented using a graphical user interface (GUI) at an operator computing station. An example of a configuration specification that is entered using the interface **60** is shown in Table 1.

TABLE 1

```
Hosts          exclude   ip     10.1.204.1–10.1.205.1
# exclude all hosts with IP addresses in this range. These hosts represent subscriber
#    home PCs
WebServers   category name   *.com.fakeisp.net WebHost
# servers with names of the form *.com.fakeisp.net are Web hosting servers
TerminalServers   extract   name    max[0–9]{3}<PopSite>[0–9]t.fakeisp.net
#Terminal servers must match the naming pattern above. Extract the POP site name
#       from the terminal server's name
```

The discovery modules **50–54** obtain the configuration criteria from the discovery engine **58**. The modules execute the appropriate discovery techniques, and as part of their outputs, record the categories to which the different services and service elements belong. The category information is stored at the discovery instance **36** and is interpreted by the service model creation engine **38** in a manner that is reflective of the service model template **34**.

The following sections provide details regarding implementation of the process described with reference to FIGS. **2** and **3**. Since the service models template **34** depends on the syntax specified in the discovery template **48**, the discovery template specification is considered first. All the templates and instances presented as examples use the INI file format that is commonly used to represent the content of system files in Microsoft Windows-based personal computer systems. However, the process may be implemented using other specification and schema definition technologies (e.g., the Common Information Model specified by the Desktop Management Task Force). Per the INI format, a template or an instance is organized as different sections, each of which is specified by a unique stream enclosed within a pair of square brackets ("[<section name>]").

Discovery Template Specification

An example discovery template specification is shown in Table 2. Each section of the discovery template defines a specific service or service element. The first four sections represent templates for discovery of external name servers, hosts, Mail SMTP servers, and Mail POP3 servers. The "module" variable specification in each section identifies the discovery module **50–54** of FIG. **4** that is to be used for the particular service or service element. The "arguments" variable represents arguments that are to be passed by the discovery engine **58** to the discovery module during deployment. The "outputs" variable defines the number and names of the columns in the output of the discovery module. The "instanceKey" variable denotes the index that is to be used to access the discovery instance **36** corresponding to each row of output generated by the discovery module. The name or names specified within angled brackets (<>) on the right side of the instanceKey assignment must correspond to one of the column names specified in the output assignment. Finally, the "dependencies" variable indicates the dependencies that a discovery module has on other modules. The discovery engine may use this information to select a sequence in which it processes the discovery templates.

TABLE 2

```
[ExternalNameServers]
discovery-module=DiscoverExternalNameServers.class
discovery-arguments=-url http://ism-pc2/scripts/discEngineAPI.pl
discovery-outputs=ipAddress, hostName, domainName, category
discovery-instanceKey=<ipAddress>:DNS
discovery-dependencies=
[Hosts]
discovery-module=DiscoverHosts.class
discovery-arguments=-url http://ism-pc2/scripts/discEngineAPI.pl
discovery-outputs=ipAddress, hostName, state, category
discovery-instanceKey=<ipAddress>:Host
discovery-dependencies=ExternalNameServers
[SMTPServers]
discovery-module=DiscoverSmtpServers.class
discovery-arguments=-url http://ism-pc2/scripts/discEngineApI.pl
discovery-outputs=ipAddress, hostName, category
discovery-instanceKey=<ipAddress>:Smtp_Mail
discovery-dependencies=Hosts, ExternalNameServers
[POP3Servers]
discovery-module=DiscoverPop3Servers.class
discovery-arguments=-url http://ism-pc2/scripts/discEngineAPI.pl
discovery-outputs=ipAddress, hostName, relatedSmtpServer, category
discovery-instanceKey=<ipAddress>:Pop3_Mail
discovery-dependencies=Hosts, ExternalNameServers
[HTTPServers]
discovery-module=DiscoverHttpServers.class
discovery-arguments=-url http://ism-pc2/scripts/discEngineApI.pl
discovery-outputs=ipAddress, hostName, serverType, category
discovery-instanceKey=<ipAddress>:Web
discovery-dependencies=Hosts
[NFSServers]
discovery-module=DiscoverNFSServers.class
discovery-arguments=-url http://ism-pc2/scripts/discEngineApI.pl
discovery-outputs=ipAddress, hostName, category
discovery-instanceKey=<ipAddress>:NFS
discovery-depenendencies=Hosts
[Host-Groups]
discovery-module=DiscoverHostGroups.class
discovery-arguments=-url http://ism-pc2/scripts/discEngineApI.pl
discovery-outputs=groupName, groupComponentsList, category
discovery-instanceKey=<groupName>:HostGroup
discovery-dependencies=ExternalNameServers
[WebServiceGroups]
hostName=mailfes21.fakeisp.net
relatedSmtpServer=smtp.fakeisp.net
category=
[10.137.196.54:Pop3_Mail]
ipAddress=10.137.196.54
hostName=mailfes22.fakeisp.net
relatedSmtpServer=smtp.fakeisp.net
category=
[10.137.196.56:Pop3_Mail]
ipAddress=10.137.196.56
```

TABLE 2-continued

```
hostName=mailfes23.fakeisp.net
relatedSmtpServer=smtp.fakeisp.net
category=
[10.137.196.58:Web]
ipAddress=10.137.196.58
hostName=www.fakeisp.net
serverType=NCSA/1.5
category=InternalWeb
[10.137.196.69:Web]
ipAddress=10.137.196.69
hostName=www.xyz.com.fakeisp.net
serverType=Apache/1.2
category=WebHost
[10.137.196.70:Web]
ipAddress=10.137.196.70
hostName=www.abc.com.fakeisp.net
serverType=Apache/1.2
category=WebHost
[10.174.173.23:NFS]
ipAddress=10.174.173.23
hostName=dns1.fakeisp.net
[pop3.fakeisp.net:HostGroup]
groupName=pop3.fakeisp.net
groupComponentsList=10.137.196.52:10.137.196.54:10.137.196.56
[pop3.fakeisp.net:Pop3_MailServiceGroup]
serviceGrpName=pop3.fakeisp.net
serviceGrpComponentsList=10.137.196.52:10.137.196.54:10.137.196.56
[ExternalDnsServer:Category]
categoryName=ExternalDnsServer
[InternalWebServer:Category]
categoryName=InternalWeb
[WebHostedServer:Category]
categoryName=WebHost
```

Because the ISP environment has a heterogeneous set of elements (e.g., host nodes, routers, application servers, services and inter-service dependencies), sections of the discovery template must be processed in an order in which elements having no dependencies are considered first. Sections have dependencies can be processed after the dependencies have been completed. There are at least three approaches to ordering the processing of sections. One approach is to order the sections in the template to reflect the dependencies among sections. Thus, a section appears in the template only after the appearance of all sections on which it depends. The discovery engine **58** can then process the sections in order. Another approach is to allow the discovery engine to dictate the sequence of discovery. By considering the values of the dependencies variable within the sections of the discovery template, the discovery engine can determine the order in which the sections must be processed. Sections of the template having no dependencies are processed first. After all such sections are processed, the discovery engine iterates through the list of template sections, choosing sections which have not been processed and which have their dependencies determined by earlier processing. This procedure is repeated until all of the sections have been processed. In the third approach, the sequencing is driven by the discovery modules **50–54** themselves. In this embodiment, the discovery engine processes the template once, invoking the discovery modules simultaneously. The discovery modules determine when the different elements of the ISP system are discovered. When a new instance is detected by a discovery module, it forwards the results to the discovery engine. Based on the dependencies on the discovery module, as specified in the discovery template, the engine forwards the results to other discovery modules for which the results are relevant. The availability of the new results may trigger discovery by other modules. This procedure is repeated until all of the sections have been fully processed. In an alternative implementation, the discovery

modules can communicate with one another without involving the discovery engine.

Discovered Instance Specification

Table 3 is a portion of an example of the discovered instance **36** of FIG. **4**. Section names in the discovered instance correspond to the instanceKey variable specifications and the discovery template of Table 2. For each of the output variables in the discovery template, there is an assignment in the discovered instance of Table 3. The ISP system being discovered in this example is assumed to have the domain name fakeisp.net.

TABLE 3

```
[10.174.173.23:DNS]
ipAddress=10.174.173.23
hostName=dns1.fakeisp.net
domainName=fakeisp.net
category=ExternalDnsServer
[10.174.173.23:Host]
ipAddress=10.174.173.23
hostName=dns1.fakeisp.net
state=Alive
category=
[10.137.196.52:Host]
ipAddress=10.137.196.52
hostName=mailfes21.fakeisp.net:smtp.fakeisp.net
state=Alive
category=
[10.137.196.54:Host]
ipAddress=10.137.196.54
hostName=mailfes22.fakeisp.net
state=Alive
category=
[10.137.196.56:Host]
ipAddress=10.137.196.56
hostName=mailfes23,fakeisp.net
state=Alive
category=
[10.137.196.58:Host]
ipAddress=10.137.196.58
hostName=www.fakeisp.net
state=Alive
category=
[10.137.196.69:Host]
ipAddress=10.137.196.69
hostName=www.xyz.com.fakeisp.net
state=Alive
category=
[10.137.196.70:Host]
ipAddress=10.137.196.70
hostName=www.abc.com.fakeisp.net
state=Alive
category=
[10.137.196.52:Smtp_Mail]
ipAddress=10.137.196.52
hostName=mailfes21.fakeisp.net:smtp.fakeisp.net
category=ExternalSmtpServer
[10.137.196.52:Pop3_Mail]
ipAddress=10.137.196.52
hostName=mailfes21.fakeisp.net
relatedSmtpServer=smtp.fakeisp.net
category=
[10.137.196.54:Pop3_Mail]
ipAddress=10.137.196.54
hostName=mailfes22.fakeisp.net
relatedSmtpServer=smtp.fakeisp.net
category=
[10.137.196.56:Pop3_Mail]
ipAddress=10.137.196.56
hostName=mailfes23.fakeisp.net
relatedSmtpServer=smtp.fakeisp.net
category=
```

Next, a portion of a service model template specification is presented in Table 4 as an example service model template **34**. The service model template contains the intelligence to map the discovered instance into the service model nodes.

The discovery modules (and hence the discovered instance) are designed independently of the service model instance being generated. This enables the same discovered instance to be used in the generation of different service models (e.g., for different services). Each section in the service model template represents a type of node in the service model instance 40 and contains a series of instructions for creating a node in the service model instance. The service model creation engine 38 processes sections of the service model template, one at a time, attempting to match the template with elements in the discovered instance 36. Each element in the discovered instance corresponds to a section of the discovered instance specification. Lines beginning with a ";" represent comments. These lines are ignored by the service model creation engine 38 when it processes the service model template.

TABLE 4

```
[SM-Host]

; Host Node in the service model
match=[*:Host]
; for each discovered Host
instanceKey=<ipAddress>:SM-Host
measurements=TCP-ConnectionRate(<ipAddress>),
        VMstat(<ipAddress>), IFstat(<ipAddress>)
; these are measurements of the host
; next copy its attributes to the SM node
hostname=<hostName>
ipAddress=<ipAddress>
state=<state>
category=<category>
[SM-Web]

; SM node for a web server
match=[*:Web]
; match all discovered Web server instances
instanceKey=<ipAddress>:SM-Web
components=<ipAddress>:SM-Host, <ipAddress>-msIP:SM-Link
measurements=HTTP-TCPConnectionTime(<ipAddress>)
; next copy all the attributes from the server discovered instance
stateComputationRule=measurementsOnly
serverType=<serverType>
hostName=<hostName>
ipAddress=<ipAddress>
category=append(<category>,<ipAddress>:SM-Host?<category>)
[SM-WebService]

; a web service node
match=[*:SM-Web]
; there is a web service node corresponding to each web server node
instanceKey=<ipAddress>:SM-WebService
measurements=HTTP-Availability (<ipAddress>), HTTP-Total ResponseTime(<ipAddress>),
        HTTP-DnsTime (<ipAddress>)
components=<ipAddress>:SM-Web, <<ipAddress>,Web>:SM-NFS, <<ipAddress>,
        Web>:SM-DNS
; NFS and DNS service dependencies will be determined by phase 2 discovery
category=<category>
[SM-WebServiceGroup]

match=[*:WebServiceGroup]
instanceKey=<serviceGrpName>:SM-WebServiceGroup
components=list(<serviceGrpComponentsList>) :SM-WebService
category=<category>
category=append(list(<serviceGrpComponentsList>) :SM-WebService?<category>)
[SM-TopLevel-Web]

instancekey=Web:SM-TopLevelWeb
components=*:SM-WebService, *:SM-WebServiceGroup
; components are all web services and all web service groups
[SM-DNS]

match=[*:DNS]
instanceKey=<ipAddress>:SM-DNS
components=<ipAddress>-msIP:SM-Link
measurements=DNS-Availability(<ipAddress>),
        DNS-CacheHitResponseTime(<ipAddress>)
stateComputationRule=measurementsOnly
ipAddress=<ipAddress>
hostName=<hostName>
domainName=<domainName>
category=<category>
[SM-NFS]
match=[*:NFS]
instanceKey=<ipAddress>:SM-NFS
components=<ipAddress>:SM-Host
```

TABLE 4-continued

```
measurements=NFS-TotalCalls(<ipAddress>), NFS-DupReqs(<ipAddress>),
        NFS-TimeOutPercent(<ipAddress>)
stateComputationRule=measurementsOnly
ipAddress=<ipAddress>
hostName=<hostName>
[SM-Pop3_Mail]

match=[*:Pop3_Mail]
instanceKey=<ipAddress>:SM-POP3_Mail
components=<ipAddress>:SM-Host
measurements=POP3-TCPConnectionTime(<ipAddress>)
hostName=<hostName>
category=<category>
relatedSmtpServer=<relatedSmtpServer>
ipAddress=<ipAddress>
[SM-ReadMailService]

match=[*:SM-Pop3_Mail]
instanceKey=<ipAddress>:SM-ReadMailService
measurements=POP3-Availability (<ipAddress>),
        POP3-TotalResponseTime(<ipAddress>),
        POP3-AuthenticationTime (<ipAddress>)
stateComputationRule=default
components=<ipAddress>:SM-Pop3_Mail, <<ipAddress>,Pop3_Mail:SM-NFS>,
        <<ipAddress>,Pop3_Mail:SM-Auth>
category=<category>
[SM-ReadMailServiceGroup]

match=[*:Pop3MailServiceGroup]
instanceKey=<serviceGrpName>:SM-ReadMailServiceGroup
components=list(<serviceGrpComponentsList>) :SM-ReadMailService
category=<category>
[SM-TopLevel-ReadMail]

instanceKey=ReadMail:SM-TopLevel-ReadMail
components=*:SM-ReadMailService, *:SM-ReadMailServiceGroup
[SM-Category]

match=[*:Category]
instanceKey=<categoryName>:SM-Category
components=*:SM-*?category=<categoryName>
```

Most sections of the service model template **34** begin with a "match" criteria. The match criteria for a section of the service model template specifies the discovery instances that are relevant to the section under consideration. For instance, the match criteria corresponding to the host node's specification (SM-Host) in the discovery template indicates that the corresponding discovered instances are those of type Host. The match criteria is specified as a regular expression that is matched against section names (instance keys) of the discovered instance **36**. For each object (section) in the discovered instance that matches the regular expression specified in the match criteria, a corresponding node is instantiated in the service model instance **40**.

Each section of the service model template **34** can match discovered instances of at least one type. When a discovered instance satisfies the match criteria specified in a section of the service model template **34**, any of the attributes of the discovered instance can be referred to in the subsequent instructions of the service model template's section. The absence of a match criteria in the specification of a section of the service model template indicates that there is only one instance of that type for the particular ISP.

The instancekey variable in Table 4 denotes the key that is to be used to refer to a service model node that is instantiated by the section of the template under consideration. The attributes enclosed within the angled brackets ("<>") must be one of the attributes of the elements of the discovered instance for which there is a reference by the match criteria.

The "components" instruction specifies the parent-child relationship in a service model instance. Various types of dependencies (i.e., execution dependencies, component dependencies, inter-service dependencies and organizational dependencies) are captured by this specification. The components list specified must make reference to the node in the service model instance **40** that is to be generated from the service model template **34**. The components list refers to specific nodes, all nodes of a specific type, and all nodes of a different specific type that have a specific attribute value. Sections of the service model template that refer to leaf nodes of the service model instance do not have component specifications.

The "measurements" instruction specifies a list of measurements that must be targeted at the corresponding node in the service mode instance **40**. By processing the measurement specifications of nodes in the service model instance, the measurement agent configurator **44** of FIG. **3** can determine the agents that must be scheduled for execution against each element of the discovered instance **36**. It should be noted that not all nodes in the service model instance have measurement specifications.

The "StateComputationRule" instruction covers how the states of the corresponding nodes in the service model instance **40** are computed. By default, the state of a node in the service model instance is determined based on the states of all of the measurements associated with a node and the states of all of its dependencies (children nodes) in the service model instance. The service model creation engine **38** may support additional state computation policies. For example, a "measurementsOnly" policy indicates that only the states of the measurements associated with a node must be taken into account in determining the state of that node.

Regarding attribute value settings, each service model node may derive attributes from the discovered instance 36 to which it refers. The service model template syntax also allows for hierarchical aggregation of attributes. This is demonstrated in the append construct used for defining category attributes for the service model nodes.

Service Model Creation Engine

The service model creation engine 38 incorporates the logic for processing a service model template 34 with the discovered instance 38 to generate the service model instance 40. There are alternatives with regard to the order in which the engine 38 processes the service model template. In a sequential processing approach, it is assumed that the service model template was constructed such that the sections of the service model template are in an order in which they need to be processed. The engine can then process the sections sequentially. The sequential processing enables simplification of the service model creation engine. However, this approach burdens the template developer with a requirement of manually determining the placement of each section in the template based upon the order of processing. Moreover, since processing typically starts from the host nodes, this approach may result in a number of service model host nodes that do not have additional nodes above them when a service model instance graph is generated in a manner to be described below. To avoid such "orphaned" nodes, the created service model instance must be further processed.

In the alternative hierarchical processing, a service model creation engine 38 can use the "components" specifications in the different sections of the service model template 34 to determine the order for processing the sections of the template. Since the components list specifies the dependencies of a node, before processing a section, all of the sections corresponding to each of the nodes types in the components list must be processed. Based on this rule, sections of the templates which have no specified components are processed first. Although this hierarchical approach requires more complexity than the sequential approach, it does not result in any orphaned nodes in the service model instance 40.

Service Model Instance Specification

Table 5 is an example of a portion of a service model instance specification for the service model template of Table 4 and the discovered instance of Table 3. The variables of the table are consistent with the variables of Tables 2–4. Referring now to FIG. 3, the service model instance 40 may be used by the view generator 42 to provide any of the three views. Preferably, the service model instance is represented as a graph of the nodes and edges that identify dependencies among the nodes. The service model instance is also used by the measurement agent configurator. Information from the instance may be extracted by the configurator to merge test information relevant to particular elements with default algorithm descriptions in order to generate a measurement agent specification 46 for the ISP of interest.

TABLE 5

```
[10.174.173.23-fakeisp.net:DNS]
ipAddress=10.174.173.23
hostName=dns1.fakeisp.net
domainName=fakeisp.net
category=ExternalDnsServer
[10.174.173.23:SM-Host]
measurements=TCP-Connection-Rate (10.174.173.23), VMstat (10.174.173.23),
        IFstat (10.174.173.23)
ipAddress=10.174.173.23
hostName=dns1.fakeisp.net
state=Alive
category=
[10.137.196.52:SM-Host]
measurements=TCP-Connection-Rate (10.137.196.52), VMstat (10.137.196.52),
        IFstat (10.137.196.52)
ipAddress=10.137.196.52
hostName-mailfes21.fakeisp.net
state-Alive
category=
[10.137.196.54:SM-Host]
measurements=TCP-Connection-Rate(10.137.196.54), VMstat(10.137.196.54),
        IFstat(10.137.196.54)
ipAddress=10.137.196.54
hostName=mailfes22.fakeisp.net
state=Alive
category=
[10.137.196.56:SM-Host]
measurements=TCP-Connection-Rate(10.137.196.56), VMstat(10.137.196.56),
IFstat(10.137.196.56)
ipAddress=10.137.196.56
hostName=mailfes23.fakeisp.net
state=Alive
category=
[10.137.196.58:SM-Host]
measurements=TCP-Connection-Rate(10.137.196.58), VMstat(10.137.196.58),
        IFstat(10.137.196.58)
ipAddress=10.137.196.58
hostName=www.fakeisp.net
state=Alive
category=
[10.137.196.69:SM-Host]
```

TABLE 5-continued

```
measurements=TCP-Connection-Rate(10.137.196.69), VMstat(10.137.196.69),
        IFstat(10.137.196.69)
ipAddress=10.137.196.69
hostName=www.xyz.com.fakeisp.net
state=Alive
category=
[10.137.196.70:SM-Host]
measurements=TCP-Connection-Rate(10.137.196.70), VMstat(10.137.196.70),
        IFstat(10.137.196.70)
ipAddress=10.137.196.70
hostName=www.abc.com.fakeisp.net
state=Alive
category=
```

Customizing the Service Model to an ISP'S Orangaizational Structure

As previously noted, the service model instance **40** of FIG. 4 is customized to an ISP's organizational structure, so that ISP operations personnel only view the status of services and service elements that are of relevance to the personnel. A straightforward approach to customizing the service model instance to an ISP's organizational structure is to edit the service model template and explicitly include nodes that capture the organizational dependencies. For example, the nodes may be grouped according to categories (e.g., InternalWeb services and WebHosting services). Each of these categories may be managed by different operations personnel. To accommodate this case, the service model template could be modified to define nodes that represent the individual categories and dependencies that indicate the components of the different categories. Since the organizational structure varies from one ISP to another, the approach would require that each ISP edit the service model templates to match their organizational structure. Editing the service model template to define the categories and the components relationships can be a tedious task, especially for a large ISP.

An alternative approach is to allow an ISP to specify its organizational structure using the configuration interface **60** previously described with reference to FIG. 4. The service model template **34** is pre-specified to exploit the configuration specification and to generate a service model instance that is customized to each ISP. The main advantage of this approach is that the ISP operator only has to primarily edit the configuration specification, which is much less complex than editing the service model template **34**.

The application of this less complex approach can be described with reference to Tables 1–5. Through the configuration interface **60**, an ISP operator specifies ways in which the discovered services and service models are to be categorized in the discovered instance **36**. In Table 1, the configuration specification indicates that the ISP uses the naming pattern *.com.fakeisp.net to identify web servers that are hosted for the businesses. Web servers that do not match this pattern are internal web servers that are used by the ISP's residential customers. Each of the discovery modules **50**, **52** and **54** then uses the configuration specification to determine a categorization of the services and service models that are discovered. A discovery module is also included in the discovery template **48** to discover and report on all the categories that have been discovered in the ISP's system. This information is used by the service model creation engine **38** to construct a service model instance **40** that represents the ISP's organizational structure. To enable this, the service model template of Table 4 has a section that generates a node in the service model instance for each category in the discovered instance. The components list in this section maps all the services and service elements that are associated with a category to the corresponding node in the service model instance. Thus, by merely specifying the categorization of services and service elements using the configuration specification, an ISP operator can derive a service model instance that is customized for the ISP.

Example Read Mail Service Model

As an example implementation of the system of FIG. 3, the email service of "Read Mail" will be considered. The Read Mail service refers to a subscriber accessing his/her mailbox from the mail system of the ISP. FIG. 5 illustrates a service topology for this service. Using a client application that supports the Post Office Protocol-Version 3 (POP3), a subscriber at a desktop computer **62** attempts to access mail. Internal to the ISP system, the request from the subscriber's computer **62** may be received and processed by one of many servers **64**, **66** and **68** that constitute a mail service group **70**. The servers within the group are front-end servers (FESs).

Before the subscriber can access the appropriate mailbox, the mail server **64**–**68** that handles the request contacts an authentication server **72** to verify the identity of the subscriber. Typically, password identification is used in the Read Mail service. A subscriber database **74** is accessed in this process. Following the authentication process, the mail FES **64**–**68** accesses a mailbox **78** of the subscriber from a back-end content server **76** using the NFS service. The retrieved mail messages are transmitted to the computer **62** of the subscriber.

There are several active and passive measurements that can be made to assess the health of the different elements involved in supporting the Read Mail service. A measurement system (MS) may be installed in the server farm of the ISP to perform measurements using agents executing on the MS and on the different ISP hosts. The different measurements that characterize the Read Mail service include an active service quality measurement of availability and response time made from the MS in the service farm, an active measurement of network throughput from the MS in the service farm to the POP sites, passive measurements of CPU and memory utilization passive measurements of TCP connection traffic and packet traffic to the mail servers obtained from agents executing on the mail servers, and passive measurements of NFS statistics (e.g., number of calls, timeouts, and duplicate transmissions) on the mail servers and the mail content servers. The active measurements attempt to assess the service quality as viewed from subscribers connecting to the POP sites, while the passive measurements may be used to assess resource utilization and traffic statistics that are critical for problem diagnosis.

FIG. 6 is an illustration of an example of a view that may be presented to an ISP operator using the view generator 42 of FIG. 3. While the oval-shaped nodes in the service model graph represent the different services and service elements, the arrows represent measurements of services and service elements. The root of the service model graph is the Read Mail service, represented by oval 80. The state of this node represents the overall health of the Read Mail service, as assessed by the MS located in the service farm of the ISP. That is, the overall health is assessed without considering the state of the network links from the server farm to the POP sites. In one embodiment, the overall health is represented by color coding the oval 80. For example, oval 80 may be shaded green to designate a positive health of the Read Mail service, and may be shaded red if the Read Mail service has degraded in its availability or performance.

Typically, there is no direct measure of the overall health of the Read Mail service. Instead, the state of the service must be inferred, based on the states of the different mail FESs 64–68 that together enable the Read Mail service. Direct active measures of availability and performance, and passive measurements of TCP statistics to the POP3 service port, together contribute to the determination of the status of the Read Mail service. The active and passive measurements are performed at each of the mail FESs, as indicated by the arrows corresponding to the second level service oval 82 in FIG. 6.

The next level of the service model graph reflects the dependencies of the Read Mail service on one element and two services. Oval 84 is the dependency of the Read Mail service on a POP3 server executing on the mail FES. Oval 86 represents the authentication service for verifying the identity of the subscriber from which a Read Mail request is received. Oval 88 represents the NFS service used by the particular mail FES. Considering the POP3 server 84 first, the health of the server is measured based on the ability to establish a TCP connection as part of the active Read Mail service quality measurement and the time required to establish the connection. In turn, the health of the POP3 server may be impacted by the link, represented by oval 90, interconnecting the mail FES to the measurement station (from which the active test is run) and the health of the mail FES host, represented by oval 92. As shown in FIG. 6, both the link 90 and the host 92 include four performance parameters that are measurable in determining the health of the nodes. While not shown in FIG. 6, the state of the various nodes may be represented by color coding or by other display means for distinguishing the states of the nodes.

Regarding the dependency of the authentication service 86 on the mail service 82, since it is possible that the authentication service is healthy but a specific mail FES is failing to perform authentication, the authentication service is first represented from the point of view of each of the mail FESs 94. Direct measures of the authentication delay when accessing through a mail FES are used to determine the state of the mail authentication service 86 in relation to that mail FES 94. The service model for an authentication service node, whose state affects the state of the mail FES-specific authentication node, is not expanded in the service model graph of FIG. 6. Likewise, the NFS service 88 is not shown as being expanded in the service model graph. The service model dependency is handled in much the same way as the authentication service dependency.

As previously noted, the service model graph of FIG. 6 represents the Read Mail service in isolation. To represent the end-to-end service, a service model must take into account the state of the DNS service used by subscribers to

resolve the ISP mail domain to each one of the mail FESs, and the state of the network links between the different POP sites and the ISP server farm. Clearly, since the different POP sites use different routes to connect to the service farm, a subscriber's perception of the end-to-end service may vary, depending upon the POP site that the subscriber is using. FIG. 7 is a service model graph for the Read Mail service as perceived by a subscriber connected to the ISP system via POP_m. The Read Mail POP_m service is represented by oval 96 and has the Read Mail service 80 of FIG. 6 as one of its dependencies. However, the service 80 is not expanded. While not shown in FIG. 7, the graphing preferably includes color coding or other designation for nodes, such as the service 80, which are not fully expanded.

The other dependencies on the Read Mail POP_m service 96 include the link 98 and the DNS service 100. The health of the link is determined by measurements of the performance parameters throughput, packet loss, delay and connectivity. The health of the DNS service 100 is determined by measurements of availability and performance. The DNS service 100 is shown as having the dependency POP_m DNS server 102. In turn, the server 102 has two dependencies, namely POP_m DNS host 104 and the link 106.

There are several ISP management functions that can exploit the capabilities of the service models. For example, a service model for operational monitoring may be configured to indicate the status of the different elements providing a service. When a failure occurs, the service model indicates which element or elements have been affected by the failure. Moreover, by traversing the service model graph top-down, an operator can determine the root-cause of the failure. For example, in FIG. 6, when a problem is noticed with the overall Read Mail service 80, an operator can traverse the service model graph to determine whether the problem is caused by a specific mail FES or whether all of the mail FESs are experiencing a problem. Assuming a similar scenario, moving down the service model graph, the operator can further determine whether the problem is related to authentication failure, NFS failure, or a failure of the POP application server 84.

Since services and service elements can be organized based on domains of responsibility in a service model, ISP operations personnel need only monitor and diagnose the services that fall in their domain of responsibility. In the Read Mail service example of FIG. 5, an email operations expert who is responsible for the mail service and the mail servers uses the service model depicted in FIG. 6, since the expert is mainly interested in the states of the email services and servers. The authentication and NFS services are included in the sèrvice model representation, since these services can adversely impact the Read Mail service. In contrast, the links between the service farm and the POP sites are not included in the model, since they do not affect the Read Mail service from the perspective of the email expert.

Measurement Topology and State Represention

As can be seen in FIGS. 5 and 6, the service model maps different measurements for some of the nodes in the service model graphs. The node to which a measurement maps depends on the semantics of the measurement (i.e., which logic node or nodes are targeted by the measurement) and the location or locations from which the measurement is made. In the simplest case, each measurement directly maps to one of the nodes in the service model. In some cases, measurements may span multiple service elements and there may not be a direct mapping of a measurement to a node in the service model. In such cases, composite measurements, which are combinations of measurements being made in the ISP system, may have to be composed and mapped to the nodes in the service model. For example, suppose Link (x,y) is a network link interconnecting hosts x and y in an ISP system, and suppose Link (x,y) is comprised of Link (x,z) and Link (z,y). If measurements are made from x, Link (x,y) and Link (x,z) can be directly measured. The status of Link (z,y) has to be derived from the status of Link (x,y) and Link (x,z).

Each of the nodes in the service model has an instantaneous state associated with it. As previously noted, the state of a node reflects the health of the service or service element that it represents. A number of policies can be used to compute the state of service model nodes. One policy computes the state of a node based on all of its measurements. Another policy assigns weights to the different measurements based on an estimate of their reliability, as gauged by a domain expert. Yet another policy determines the state of a node based on its measurements in combination with the states of its dependencies in the service model graph. The states of the measurements associated with a node may be determined by applying baselining and thresholding techniques to the measurement results.

Discovery Methodologies

Returning to FIGS. 2 and 4, the preferred embodiment of the management system **56** includes a discovery engine **58** that automatically discovers services, service elements and dependencies among services and service elements. FIGS. 7 and **8** illustrate applicable discovery methodologies. Similar to existing management system implementations, a first phase of discovery is performed from a management station **108**, which may be internal or external to the service farm of the ISP system. Predominantly, this phase involves active tests that generate test traffic and query all of the ISP hosts to detect the existence of different types of servers **12, 14, 22, 26** and **28**. That is, the phase detects execution dependencies, as defined above. Component and organizational dependencies are also detected during this phase. Moreover, some of the inter-service dependencies are discovered, but the second phase is focused on the inter-service dependencies, since it may not be possible to discover all the inter-service dependencies that exist using tests executed from outside the ISP host. The second phase uses an internal view of the ISP system. Preferably, the two phases are executed sequentially, with the second phase utilizing the discovered information output by the first phase to direct its operations. Different mechanisms can be employed in the internal discovery phase. Both phases of discovery must be executed periodically, so as to discover new services and service elements that may have been introduced into the ISP environment.

In FIG. **8**, a single discovery agent **110** is used in the first phase discovery process. The solid line **112** represents the discovery agent contacting the DNS server **14** to get a list of hosts in the ISP system. The dashed lines from the discovery agent **110** indicate active tests being executed by the discovery agent to detect various types of dependencies that exist.

FIG. **9** is an illustration of the second phase discovery process. In this phase, a number of internal discovery agents **114, 116, 118** and **120** are utilized. The solid lines having arrows at one end indicate the discovery of inter-service dependencies. The dashed lines indicate the flow of discovered instance information back to the management station **108**.

Both phases of the discovery process may employ one or more different basic categories of discovery techniques. Under the service-independent techniques, hosts and networking equipment (routers **20**, hubs **18**, and POP sites **16**) which are not specific to any service are discovered. As a second category, service-generic techniques (which may be the same techniques, but with appropriate parameterization for each service) may be used to discover instances of different services. In order to do so, typically, such discovery techniques exploit common characteristics of different services to discover instances of the services. An example of this second category is a discovery technique for News, Web and email services. Since all of these services rely upon the same transport protocol (i.e, TCP), a discovery technique can discover the existence of application servers by monitoring the TCP ports corresponding to the services.

Another category of techniques may be referred to as the service-specific-but-application-independent techniques. Techniques in this category are specific to the service. They are intended to monitor, but may be used for discovery that is independent of the specific application server that is being used to implement the service. For example, the discovery of the relationship between the services offered by POP3 email servers and the service provided by SMTP-based email servers is possible using application-independent techniques, since the relationship is accessible from the domain name service in the form of mail exchange (MX) records.

A fourth category may be referred to as application-specific techniques. Many inter-service dependencies may need to be discovered in a manner that is very specific to the application servers providing the services. For example, to discover the dependency of the web service on NFS, the discovery technique may have to query the configuration file in the web application server that is providing the web service. Since the format and contents of a web application server's configuration file are specific to the application, the discovery technique is application-specific.

First Phase Discovery

An often under utilized component of an ISP system is the domain name service (DNS). In order to allow subscribers to access hosts using their host names, rather than their more difficult to remember IP addresses, DNS stores the host name-to-IP address mapping for all of the hosts in the ISP system. Moreover, the exchange of email messages across hosts occurs using the mail exchange records maintained by the DNS. Name server (NS) records in the DNS database serve to identify authoritative name service for the ISP domain—these are name servers that are externally accessible from the global Internet and are the authorities that are contacted when users in the global Internet attempt to access any hosts in the ISP system. Moreover, service groups, such as an email service group, are enabled via round-robin scheduling mechanisms implemented in the DNS servers. In summary, the domain name system holds a wealth of information that is critical for auto-discovery of ISP services. However, additional mechanisms are necessary to complement DNS-based discovery mechanisms.

One of the first steps in discovery is to determine all of the hosts that exist in the ISP system. Most existing network management systems have taken one of two approaches. A first approach is to scan an address range that is either specified by an operator or is determined based on the local subnet of the measurement host. The address range is scanned using ping to solicit responses from all IP-enabled hosts. The second approach is to use the default router configured for a host to boot-strap the discovery. Communication using SNMP with the router and using its routing tables may be used to discover hosts in the local subnet. Also, routing table information may be used to discover other routers that can provide additional host information.

An alternative approach that is complementary to either of the two traditional approaches is to obtain a list of all of the hosts in the ISP system using information available with the domain name service. From the host name of the management station in which the discovery process executes, the ISP domain name can be deduced. Using the default name service configured for the management system, the discovery process queries DNS to obtain the NS records that list the authoritative name servers for the ISP domain. One or more of these name servers are contacted by the discovery process to obtain a list of named hosts in the ISP system. Zone transfer capabilities supported by all DNS servers are used for this purpose, as is known in the art. While ISPs usually manage a single domain, some ISPs may have multiple domains under their control. To discover all of the hosts that exist in the different domains, the discovery process must be informed of the existence of all the different domain names. This information cannot be automatically deduced by the discovery process.

The steps that are executed in the first phase of discovery are identified in FIG. 10. It is not critical that the steps be followed in the order shown in the figure. Following the step 122 of discovering the hosts, the application servers are discovered in step 124. The existence of application servers of different types (e.g., Web, Email, News, FTP, DNS, NFS, and Radius) is verified by active tests that emulate typical client requests directed at the different TCP and UDP ports corresponding to the different service types. A response to an emulated request has to be interpreted in a service-specific manner. By observing the header in a response from the web service, the discovery process determines the type of web server that is executing on the host machine. Likewise, by processing the response returned by the email and News servers, the discovery process determines the type of servers. This information can be used to customize the second phase of discovery. For instance, discovery agents installed on the ISP host machines in the second phase of discovery may process a web application server's configuration files to discover NFS dependencies that the server may have. Since web server configuration files are typically specific to the type of server, the server type information provided by the first phase of discovery can be used to determine the processing capabilities of the discovery agent or agents that must be deployed in the second phase. The server type information may also be used to determine specific measurements which must be targeted for the server to monitor its status.

In step 126, the existence of Web, Email, News, DNS and other service groups has to be determined using the DNS. By querying the DNS, the discovery process determines a list of domain names that have multiple IP addresses associated with them. For each name in the list, the discovery process then determines whether each of its IP addresses hosts a common application server. If so, the name likely represents a DNS round-robin service group that supports a common service. For example, suppose that all of the IP addresses corresponding to the name www.fakeisp.net host web servers. In this case, www.fakeisp.net represents a web service group. Note that in this process, a host that has two network interfaces, and therefore is assigned to different IP addresses, may be listed as a service group. Using the virtual host/

server discovery heuristics discussed below, all such hosts can be removed from the service group list.

In step 128, the MX records for the ISP domain are accessed from the DNS system. The MX records indicate the mail servers that must be contacted to deliver mail to the ISP domain. The list of SMTP-based mail servers thus determined represent the servers that handle delivery of incoming mail to the subscribers of the ISP. Discovery of these servers is essential to automatically generating a service model for the email service that represents the delivery of mail from the Internet to the subscribers. Moreover, the mail gateways may be managed by a different entity than the one that manages mail servers that are internal to the ISP.

One of the critical measures of the performance of an email service of an ISP is the round-trip delay between the transmission of an email message from a source host and its reception at the intended destination. This measurement can be used to assess email delivery times in the ISP domain, from the ISP domain to locations on the Internet, and from locations on the Internet to the ISP domain. Since the email service uses different protocols and, hence, different application servers to send mail and to receive mail, in order to initiate round-trip delay measurements of email, it is essential to determine relationships between the different types of email servers on the ISP domain (e.g., which SMTP server can be used to send mail to a POP3/IMAP-based server). This discovery is executed at step 130. Since mail forwarding is predominantly based on the MX records maintained in the DNS database, by querying the DNS system for MX records corresponding to each of the POP3/IMAP servers, the discovery process determines the mail service relationships in the ISP domain.

Various approaches can be adopted to discover the terminal servers that exist in an ISP POP site in step 132. The most straightforward approach uses SNMP queries to obtain the MIB-II system description from all the hosts in the ISP network. Based on the replies, the discovery process can identify the hosts that are terminal servers. An alternative approach is based on the observation that because they need to operate and manage thousands of terminal servers, most ISPs have specific naming conventions that they use when naming their terminal servers. In fact, the naming convention more often indicates the association between terminal servers and POP sites, so that when a problem is reported by a subscriber using a POP site, the ISP operations staff can quickly decide which of the terminal servers needs to be checked in order to diagnose the problem. With this approach, an ISP provides a regular expression representing the naming convention used as input to the discovery process. By matching the list of hosts that are discovered with the naming convention, the discovery process not only determines the terminal servers that exist, but also determines the POP site to which a terminal server is assigned. Another key advantage of this approach is that it performs discovery without generating any additional network traffic.

The approach of exploiting name conventions may also be used in step 134 to categorize the other services of the ISP. As an example, for each web site that it hosts for its business customers, a particular ISP may assign internal names of the form *.com.fakeisp.net, so that a hosted web site named www.customer-domain.com will have a corresponding entry for www.customer-domain.com.fakeisp.net in the internal DNS database of the ISP. As in the case of terminal servers, by permitting an ISP to specify its naming conventions, the discovery process composes a categorization of services that is customized to the target ISP system. This categorization can be based on geographical locations of services, based on business relationships, and/or based on the delegation of responsibilities among operators. The categorization information can be used to automatically define the customized service model for each ISP, with special nodes in the service model representing a collection of nodes pertaining to the same category.

Second Phase Discovery

By treating the ISP system as a "black box," the first phase of discovery detects most of the execution, component and organizational dependencies of the ISP. Additionally, some of the inter-service dependencies are discovered. The second phase of the discovery process is focused solely on detecting inter-service dependencies, particularly those that are not discovered by taking an external viewpoint. For example, the relationship between a mail server and an NFS server is not discoverable from an external viewpoint.

There are two basic approaches for conducting the second phase discovery. One approach uses network probes, while the other approach uses special-purpose discovery agents. Regarding the first approach, software probes installed at strategic locations on the network can snoop on packet transmissions. Since most TCP/IP communication is based on source/destiny port numbers, by processing the headers of packets that are captured by the probe, a software probe can deduce many of the relationships that exist among services. For example, a UDP packet transmitted from a mail server to the NFS port of an NFS server indicates that the mail server depends on the NFS server for its content storage.

An advantage of the approach that utilizes network probes is that the approach enables discovery of inter-service dependencies independently of the specific types of application servers residing on the host. Moreover, since it relies on just the ability to capture packets on the wire, this approach handles UDP and TCP-based services equally well.

The key difference between the approach of using network probes and the approach of using special-purpose discovery agents is that unlike the network probes, the discovery agents do not snoop on packet transmissions. Instead, the discovery agents use a number of operating systems and application-specific mechanisms to discover inter-service dependencies. These mechanisms include (1) processing service configuration information and (2) application-dependent monitoring tools. Referring first to the processing service configuration information, application servers determine their dependencies on other services from one or more configuration files. By processing the content of the configuration files, discovery agents can discover inter-service dependencies. An example of this is the processing of the web server's configuration file to discover whether it has a dependency on an NFS service. While processing the web server's configuration file, the discovery agent can also determine if the same application is being used to host multiple "virtual" servers (which is commonly used by ISPs to host web sites on behalf of their business customers). Typically, web server configuration files are specific to the type of server executed on the web server in use. The server type determination performed during the first phase of discovery is used for deciding the location and format of the configuration files.

While many Unix operating systems use configuration files that are defined in an application-specific manner, Windows NT-based systems store all application configuration information in the registry. In the Windows NT systems,

while the registry can be processed in an application-independent manner, the specific configuration attributes have to be interpreted in an application-specific manner.

Thus far, only forward-looking discovery agents have been identified. These are agents that discover dependencies of a service on other services by querying configuration files of the application providing the service. Sometimes it is easier to implement backward-looking discovery agents to discover the dependencies on a service (i.e., discover which other services are using the service). For example, the configuration file of the mail authentication server may indicate which of the mail servers are depending on the authentication server. One of the ways of implementing backward-looking discovery agents is by processing application server configuration files.

Turning now to the second mechanism of using application-independent monitoring tools, this approach is particularly attractive for services that communicate using TCP. The netstat utility can be used to determine the TCP connections that exist on an ISP host. A discovery agent that executes this tool can periodically discover information about the source and destination ports and the host locations for TCP connections, which can then be used to deduce inter-service dependencies.

This second approach of using application-independent monitoring tools exploits the fact that most TCP implementations enforce a three-minute delay for connections in the TIME_WAIT state of TCP, so that a connection persists for about three minutes even after it is no longer in use. Consequently, whenever the discovery agent is executed, it is likely to detect all the TCP connections that may have been established in the three minutes prior its execution. This same approach does not work for UDP-based services, since UDP is connectionless, and there is no state that is maintained at either the source or the destination.

The approach of monitoring TCP connections can be used to discover dependencies such as those that exist between mail servers and mail authentication servers, between servers and back-end databases, between Radius/TACACS authentication servers and terminal servers, and between similar relationships. Again, discovery agents can be forward-looking or backward-looking.

Advantages of using discovery agents, as compared to network probes, include the reduction of overhead on the ISP hosts, the relaxation of security concerns, and the fact that all of the discovery agents do not need to be deployed at the same time. Instead, the deployment of discovery agents can occur at the discretion of the ISP. As and when new discovery agents are installed on the ISP hosts, additional information is discovered about the ISP system.

FIG. 11 is a process flow of steps relevant to the second phase of the discovery process. In steps 136 and 137, the information that is obtained in the first phase of discovery is used to generate an incomplete service model instance. As previously noted, the first phase of discovery provides the necessary information for identifying component dependencies, organizational dependencies, execution dependencies and some of the inter-service dependencies. A first instance generator matches the service model template with the auto-discovered information from the first phase to generate the incomplete service model instance. However, other inter-service dependencies are not discoverable using the techniques of the first phase (e.g., the relationship between a mail server and an NFS server).

In steps 138 and 139, the holes in the incomplete service model instance are identified and information obtained in the first phase is used to determine appropriate discovery actions. The incomplete service model instance is used to determine the types of relationships that must be examined. In the mail service example, if a host is discovered in the first phase to be running a POP3 server, the second phase may be used to discover the name file service and authentication service used by the POP3 server on the particular host.

In step **140**, the network probes and/or the special-purpose discovery agents are deployed in a manner determined during execution of the step **139**. For example, application-specific knowledge may be used to parse configuration files or log files, or may be used to search a configuration registry for a particular server instance executing on a particular host. The network probes and/or discovery agents generate service dependency outputs in step **141**. The outputs are used in a second instance generation to complete the service model instance in step **143**.

Discovery of Web Hosting Environments

An ISP system that hosts web sites for business customers poses several challenges for discovery. Typically, each web site of a business customer of the ISP has a unique name (e.g., www.customer-domain.com). The ISP is typically authoritative for the customer domain, i.e, one or more of the ISP's name servers advertise the customer's domain to the global Internet. There are three different models for web hosting in an ISP system: (1) dedicated hosts; (2) virtual hosts; and (3) virtual servers. In the dedicated hosts model, the web site of the customer may be supported on one or more dedicated hosts at the site of the ISP, in which case, there are one or more IP addresses associated with the customer's web site. On the other hand, the virtual hosts model is an approach in which multiple customer web sites are supported using the same host machine in the ISP system. In this case, there is a unique IP address associated with each customer's web site. Using capabilities built into the newer operating systems, the ISP can set up multiple virtual interfaces that map to one of the physical interfaces on the host machine. Each virtual interface is associated with an IP address, which in turn maps to one of the virtual hosted web sites. The web application server configuration file defines the root directory corresponding to each customer's web site. When it receives an HTTP request, the web server processes the IP address of the server, which is specified in the HTTP request header, to determine which root directory is used for servicing the request.

With regard to the virtual servers model, such servers are found when all of the customer web sites supported using a single host machine have an IP address that is common to the host machine. To map an incoming request to a virtual web site, the web server application executing on the host exploits recent modifications made to the HTTP protocol in version 1.1. Web browsers that are compatible with HTTP/1.1 specify the web site being accessed as part of the HTTP request. Web servers that are compatible with HTTP/1.1 process the web site name and the request to determine which of the various virtual servers the request is destined for and, therefore, which of the many configurations (root directory, access list, etc.) must be used to service the request. To support this approach, the ISP associates the virtual servers with the IP address of the host using canonical name (CNAME) records in the DNS database.

There are two approaches for discovering the customer domains for which an ISP hosts web sites. In a first approach, the naming patterns of hosts in the ISP domain are exploited. As previously noted, some ISPs have host names in their domain representing the business customer web sites. For example, for each customer web site (e.g., www.customer-domain.com), the ISP may have an alternative name listed in the domain of the ISP (e.g., www.customer-domain.com.fakeisp.net). In this example, by scanning the list of all hosts in the ISP domain (i.e., fakeisp.net) and searching for host names that match the naming pattern *.com.fakeisp.net, the discovery process determines all the customer domains for which fakeisp.net hosts web sites.

In the second approach for discovering the customer domains, a discovery agent is used on each of the authoritative name servers of the ISP. In the event that the customer web sites are not listed in the ISP's domain, this alternative discovery approach becomes necessary. For a majority of sites that they host, the name servers of the ISP are also authoritative for the corresponding customer domains. Unfortunately, the DNS system does not support queries that permit an external discovery agent to query a DNS server for all domains for which it is authoritative—almost all forms of DNS queries assume that a customer is aware of the domain name of interest. Hence, to discover all of the customers whose web sites have been hosted by the ISP, a discovery agent on one or more of the main name servers of the ISP is used. For each domain name that it supports, there is a unique database that the DNS server maintains. The discovery agent on the DNS server accesses this information and reports back to the management station **108** of FIG. **9**.

Once the customer domains that are supported by the ISP are determined, the discovery process executes the first and second phase discovery methodologies to discover the hosts and services in the different customer domains. In order to enable service models to be created for web hosting services, it is essential to discover the virtual hosts and the virtual servers. There are two possible approaches to executing the discovery of the virtual hosts. In a first approach, first phase discovery is implemented by interpreting application server responses. A key observation guiding this approach is that in an ISP system, only web servers support virtual hosting. That is, the email (POP3, SMTP), News, and FTP application servers typically do not support virtual hosting. When the email, News, and FTP application servers are targeted with active tasks during the first phase discovery process, they return the name of the host machine from which they are executed as part of the response. Since the email, News, and FTP application servers are not aware of the existence of the virtual hosts, when the servers execute on a host that supports other virtual hosts, the servers return the name of the host machine (not the names of the virtual hosts) as part of their response. To discover the virtual hosts within this first approach, the discovery process determines all the host names that exist in the ISP system. The discovery process then targets each of the host names, attempting to connect to the email, News, or FTP application servers. In the event that a connection succeeds, the discovery process logs the name or names returned by the application servers as part of their response. The host name corresponds to a virtual host if its host name in the DNS database does not match the name returned by the email, News, or FTP application servers in response to active tests. For a virtual host, the name returned by the email, News, or FTP application servers represents the identity of the host machine that supports the virtual host.

In the second approach to discovering virtual hosts, second phase discovery uses discovery agents executing on the ISP hosts. In this implementation, a potentially more reliable method for discovering virtual hosts is accessed by using discovery agents installed on different hosts of the ISP system. By checking the host application server configuration files or by checking the configuration of network interfaces on the host machine, a discovery agent can determine whether a host supports virtual hosting or does not. Since virtual hosts are relevant mainly in the context of web sites, the discovery agents may be installed only on hosts that have web servers executing on them (as discovered during the first phase discovery process).

The virtual servers must also be discovered. All IP addresses that have multiple host names associated with them in the DNS database are candidates for hosting virtual servers. However, this is not a sufficient condition for identifying virtual servers, since many times multiple host names are associated with the same host for naming convenience or for other administrative purposes. A more reliable method of identifying virtual servers and hosts that support them is to use discovery agents that can process the web application server configuration files.

Extensible Discovery Architecture

Since new network services and service elements are being deployed at a rapid pace, it is important that the discovery methodologies be implemented in an extensible manner, allowing new discovery capabilities to be incrementally added to the management system. FIG. 12 depicts the extensible architecture for discovery components previously described with reference to FIG. 4. The discovery modules **50**, **52** and **54** represent the logic used for discovery of different services and service elements. The discovery template **48** is the key to the extensibility of the auto-discovery architecture in the sense that it drives how discovery is performed. The template defines the different services and service elements that need to be discovered, and the specific discovery modules that can be used to discover these elements. The template also establishes the format of the outputs from the modules.

The discovery engine **58** drives the auto-discovery process. The discovery engine interprets the discovery template **48** and for each of the service or service element types specified in the template, the engine invokes the corresponding discovery module **50**, **52** and **54** specified in the template. All of the discovery modules report the results of their execution back to the discovery engine. The discovery template contains instructions for the discovery engine to process the output of the discovery modules and to record them as a discovered instance **36**.

Some discovery modules may rely on the discovery results of other discovery modules. For example, a DNS round-robin service group discovery module for web services relies on identifying which hosts support web services, which is an output of the web service discovery module **50**. To accommodate these relationships, as part of the interface that the discovery engine **58** exposes to the discovery modules, the engine provides ways for accessing and searching the instances discovered by other discovery modules.

In contrast to the discovery modules **50**, **52** and **54**, the discovery engine **58** is designed to be independent of the services that need to be discovered. Consequently, to discover new services or service elements, a user merely has to provide a discovery template specification or one or more discovery modules for each new element. By providing an alternate discovery module for a service that is already supported, a user can also enhance capabilities of an existing discovery system.

In practice, there are two significantly different approaches to designing the discovery engine **58** and the discovery modules **50**–**54**. A first approach is to enable the discovery engine to control the discovery process. The discovery engine accesses the discovery template and determines the order in which sections of the template are processed. On the other hand, in the second approach, the discovery modules drive the discovery process. In effect, this is an event-driven approach, since the results obtained from one module will trigger subsequent activities by other modules.

Regarding the first approach in which the discovery process is driven by the discovery engine **58**, FIG. **13** illustrates the logical building blocks of the discovery engine. The discovery engine executes periodically and each time it starts, the engine processes the discovery template.

By considering the values of the dependencies variable for each of the sections in the discovery template, the discovery engine determines the order in which the sections must be processed. Thus, the discovery engine includes a template parser **142**. Sections of the template which have no dependencies are processed first. A module loader **144** directs the relevant information to the appropriate discovery module **146** for processing a particular section in which no dependencies are identified. After all such sections are processed, the discovery engine iterates through the list of template sections, choosing sections which have not been processed and which have their dependencies determined by earlier processing. This process is repeated periodically to discover new instances as and when they are added to the system being discovered. In one application, the discovery engine uses the exec system call to invoke the discovery modules at separate processes. By doing so, the discovery engine is able to handle discovery modules written in a variety programming environments.

A query processor **148** of the discovery engine **58** performs two functions. First, when a module **146** is activated, the processor **148** queries the discovery engine to obtain configuration information that guides the discovery modules. In FIG. **4**, the configuration information is generated from the configuration interface **60** that is manipulable by a user. Table 1 was previously included to depict a typical configuration file. Each line in the file represents an instruction for one of the discovery modules. The first column of the line identifies the discovery module to which the instruction pertains. There are three types of instructions that are specified in the configuration file. All of these instructions specify regular expression patterns that must be applied against the IP address or host name of the service or service element. The instructions in the configuration file are (1) criteria that instruct the discovery modules to include or exclude specific services or service elements, (2) criteria that instruct the discovery modules to associate specific services or service elements with certain categories, and (3) criteria for discovering terminal servers and for extracting POP site-to-terminal server mapping from the terminal server names.

The second function of the query processor **148** is to provide the discovery modules **146** with access to previously discovered instances. Based on configuration and discovered instance information obtained from the query processor, the discovery modules perform tests on the ISP system and report their discovery output to the discovery engine. A discovery instance generator module **150** of the discovery engine processes the results of the discovery modules and outputs the discovery instance in an appropriate format. An example of such a format was previously set forth in Table 3. The formats of the discovery template and the discovery instance are thereby hidden from the discovery modules.

As previously noted, the second approach to designing the discovery engine 58 and the discovery modules 50–54 of FIG. 12 is to establish an arrangement in which the discovery process is driven by the modules. In this alternative embodiment, the discovery engine processes the template once, invoking the discovery modules simultaneously. From this point, the discovery modules determine when different elements in the ISP system are discovered. The discovery modules execute periodically, looking for new instances. Some discovery modules are independent in the sense that they are not reliant on other modules for discovery. These independent modules begin executing immediately.

As and when a discovery module 50–54 discovers a new instance, the discovery module forwards its results to the discovery engine 58. Based on the dependencies on a discovery module, as specified in the discovery template 48, the engine 58 forwards the results to other discovery modules for which the results are relevant. The availability of new results (e.g., the discovery of a new host) may trigger discovery by other modules (e.g., the web server module checks to determine if a web server is executing on the new host), and this process continues. A key advantage to this approach, as compared to the engine-driven discovery approach, is that multiple discovery modules may be executing in parallel, discovering the ISP's services. In this approach, the discovery engine 58 mainly functions as a facilitator of communication among the discovery modules. A variant of this approach may not even involve the discovery engine, with the discovery modules registering interest in other discovery modules and information concerning newly discovered instances being directly communicated among the discovery modules, without involving a discovery engine.

Integrating Discovery with Service Models

In the scenario in which the management system uses service models for management of Internet services, there are two ways in which discovery can be integrated with service models. In a looser integration, the output of discovery (the discovered instance) is integrated with a service model template that outlines the structure of a service, and the integration automatically generates a service model instance that is customized for the ISP system being managed. However, the preferred integration is one that provides a tighter integration, and involves driving auto-discovery and service model instantiation from a common template. In this preferred approach, for each node in the service model, corresponding discovery template specifications are provided. The discovery and service model-specific components of the template can either be processed in a single application or can be processed separately. This approach towards tighter integration of discovery and service model templates is attractive for several reasons. Firstly, the service model template can serve to constrain the discovery process, since only services and service elements that are specified in the service model template need to be discovered. Secondly, depending upon its design, the service model template could end up using some of the outputs of the discovery process. Using a common template permits tighter syntax checking across the discovery and service model components of a template. Thirdly, the two-phase approach to discovery described above fits in well with the service model concept. The inter-service dependencies that need to be discovered in the second phase (internal discovery) can be determined based on the service model template. Finally, the discovery process itself can be determined based on the service model template specification. The discovery process may attempt

to traverse the service model template tree from a root node down. At each level, it attempts to discover all services or service elements of the types specified by the node, providing all of the children of a node that have been discovered. If this is not the case, the discovery process proceeds to first discover all instances of the children nodes. Continuing the tree traversal recursively, the discovery process discovers all instances that are necessary to build the service model for the ISP system being managed.

What is claimed is:

1. A method of identifying elements, services and dependencies among said elements and services of a network comprising steps of:
   - executing a first phase of discovery such that a plurality of services and service elements that are cooperative in performing said services within said network are detected, including discovering a first set of dependencies among said services and service elements, where said services are functionalities offered by said network to perform specific tasks;
   - executing a second phase of discovery using discovery results of said first phase such that inter-service dependencies among said services detected in said first phase are identified, each said identified inter-service dependency being related to a reliance of one of said services upon at least one other of said services; and
   - forming a network model that is specific to at least one said specified service detected in said first phase such that said network model maps said first set of dependencies and said inter-service dependencies that are relevant to said at least one specified service.

2. The method of claim 1 wherein said step of executing said second phase to identify said inter-service dependencies is an automated process that is based on said detection of said services and service elements in said first phase of discovery.

3. The method of claim 1 wherein said step of executing said second phase includes deploying discovery agents implemented in computer software, including enabling said discovery agents to access content of configuration files of applications that are detected in said first phase of discovery, such that accessing said content is specific to determining said inter-service dependencies.

4. The method of claim 1 wherein said step of executing said second phase includes deploying discovery agents implemented in computer software, including enabling said discovery agents to monitor connections completed via specified service elements detected in said first phase of discovery, such that said inter-service dependencies are identified.

5. The method of claim 4 wherein said step that includes deploying said discovery agents includes enabling said discovery agents to identify Transmission Control Protocol (TCP) connections of at least one host that is detected in said first phase.

6. The method of claim 1 wherein said step of executing said second phase includes deploying network probes to access information embedded within data packets transmitted between said service elements detected in said first phase, said second phase further including utilizing said accessed information of said data packets to detect said inter-service dependencies.

7. The method of claim 1 wherein said step of executing said first phase includes accessing information of a domain name service (DNS) of said network, including identifying at least two of (1) internal and external name servers, (2) round-robin service groups of said network, and (3) virtual servers and virtual hosts of said network.

8. The method of claim 1 wherein said step of executing said first phase includes recognizing naming conventions within said network, including recognizing and utilizing naming conventions relating to terminal servers of said network, said step of executing said first phase further including identifying execution dependencies relating directly to an application server being executed on a host machine and including identifying component dependencies that ensure redundancy of said services.

9. The method of claim 8 wherein said step of recognizing said naming conventions includes recognizing and utilizing patterns of host names to identify World Wide Web (WWW) sites that are stored on a common host machine of said network.

10. The method of claim 1 further comprising a step of selecting a particular core service of said network, said steps of executing said first and second phases and forming said network model being implemented in a manner specific to modeling said core service, said step of forming said network model thereby providing a representation of nodes and node-to-node connections which link all of said services, service elements and dependencies that are relevant to said core service.

11. A method of identifying elements, services and dependencies among said elements and services comprising steps of:

accessing information of a domain name service (DNS) of a network; and

utilizing said information of said DNS as a basis for determining a plurality of:

(a) a group of service elements that are generally equivalent with respect to executing a particular service within said network;

(b) a host supporting virtual hosting;

(c) a host supporting virtual servers; and

(d) name servers that are authoritative for a domain.

12. The method of claim 11 further comprising a step of selecting a core service of said network, said step of utilizing said information of said DNS being executed to model said core service, including modeling said core service such that said network components that are used to perform said core service are represented as nodes and network dependencies among said network components are represented as edges among said nodes.

13. The method of claim 12 wherein said step of selecting said core service includes identifying a service of an Internet Service Provider (ISP) and said step of modeling is executed to represent the cooperation within said ISP to perform said core service.

14. The method of claim 13 wherein said step of utilizing said information further includes determining SMTP servers that correspond to hosts which run POP3 servers.

15. The method of claim 14 wherein said step of utilizing said information further includes determining external mail gateways for the ISP.

16. A system for identifying service elements, services and dependencies among said service elements and services of a network comprising:

a discovery engine means for driving first and second phases of discovering said service elements, services and dependencies, where said services are functionalities offered by said network to perform specific tasks and where said service elements are cooperative in performing said services;

first discovery tools, responsive to said first phase of said discovery engine means, for accessing first information indicative of said service elements, services and a first set of dependencies among said service elements and services, including first information indicative of applications and first information indicative of dependencies among said service elements;

second discovery tools, responsive to said second phase of said discovery engine means and based on said first information, for accessing second information indicative of a second set of dependencies among said service elements and services, said second discovery tools including discovery agents executed in computer software that is configured to detect inter-service dependencies among said services; and

means for generating a discovered instance of at least a preselected portion of said network based on said first and second information from said first and second discovery tools thereby generating a network model which maps said first and second information as interconnected nodes in said discovered instance of said preselected portion.

17. The system of claim 16 wherein said discovery agents are configured to access configuration files of said applications and detect said inter-service dependencies based on said configuration files.

18. The system of claim 16 wherein said discovery agents are configured to monitor connections completed via specified service elements detected by said first discovery tools, said connections including TCP connections completed via a specified host machine.

19. The system of claim 16 wherein said first discovery tools include software configured to access a DNS of said network and to retrieve information indicative of at least two of (1) name servers, (2) round-robin service groups, and (3) virtual servers and virtual hosts.

20. The system of claim 16 wherein said first discovery tools include means for recognizing naming conventions of said service elements of said network, thereby enabling classification of said service elements at least partially based on type and geographic location.

* * * * *

US006178505B1

(12) **United States Patent**
Schneider et al.

(10) **Patent No.: US 6,178,505 B1**
(45) **Date of Patent: Jan. 23, 2001**

(54) **SECURE DELIVERY OF INFORMATION IN A NETWORK**

(75) Inventors: **David S. Schneider**, Woodland Hills; **Laurence R. Lipstone**, Calabasas; **Daniel Jensen**, Van Nuys, all of CA (US); **Michael B. Ribet**, Oak Brook, IL (US)

(73) Assignee: **Internet Dynamics, Inc.**, Westlake Village, CA (US)

(*) Notice: Under 35 U.S.C. 154(b), the term of this patent shall be extended for 0 days.

(21) Appl. No.: **09/034,576**

(22) Filed: **Mar. 4, 1998**

Related U.S. Application Data

(60) Provisional application No. 60/039,542, filed on Mar. 10, 1997, and provisional application No. 60/040,262, filed on Mar. 10, 1997.

(51) **Int. Cl.**^{7} **H04L 9/32**
(52) **U.S. Cl.** **713/168**
(58) **Field of Search** 713/167, 154, 713/155, 166, 168, 164

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,276,735	* 1/1994	Boebert et al.	....	713/167
5,864,683	* 1/1999	Boebert et al.	....	709/249

FOREIGN PATENT DOCUMENTS

WO 96/05549 2/1996 (WO) G06F/1/00

OTHER PUBLICATIONS

CheckPoint FireWall-1™ White Paper, Version 2.0—Jun. 1995. http://www.integralis.co.uk/checkpnt/firewall/white
Checkpoint FireWall-1, http://www.metadigm.co.uk/fw1/. 1996 *Metadigm Ltd.*
Commercial FireWalls and related FW Products, http://hp735c.csc.cuhk.hk/firewall.html. Mar. 23, 1996.
Five Domains of Network Security, *Technical Overview of the Eagle*, http://www.raptor.com/. . .T22NZ.Z56DAM.-BF3AQD.F2.
Firewakks and Security Related Information, http://www.nacisa.nato.int/FWVendor.HTM.
Che-fun Yu, Access control and authorization plan for customer control of network services, in: *IEEE Global Telecommunications Conference and exhibition, Conference Record*, vol. 2, pp. 862–869.
PCT/US98/04522, Partial international search, with indications of relevance of the references cited above. (PCT/US98/04522 has the same Specification as the application in which this IDS is being filed).

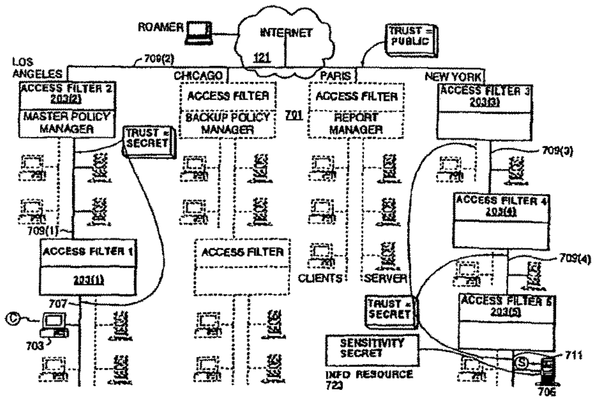
* cited by examiner

Primary Examiner—Tod R. Swann
Assistant Examiner—Matthew Smithers
(74) *Attorney, Agent, or Firm*—Gordon E. Nelson

(57) **ABSTRACT**

A scalable access filter that is used together with others like it in a virtual private network to control access by users at clients in the network to information resources provided by servers in the network. Each access filter use a local copy of an access control data base to determine whether an access request made by a user. Changes made by administrators in the local copies are propagated to all of the other local copies. Each user belongs to one or more user groups and each information resource belongs to one or more information sets. Access is permitted or denied according to of access policies which define access in terms of the user groups and information sets. The rights of administrators are similarly determined by administrative policies. Access is further permitted only if the trust levels of a mode of identification of the user and of the path in the network by which the access is made are sufficient for the sensitivity level of the information resource. If necessary, the access filter automatically encrypts the request with an encryption method whose trust level is sufficient. The first access filter in the path performs the access check and encrypts and authenticates the request; the other access filters in the path do not repeat the access check.

26 Claims, 31 Drawing Sheets



US006016318A

United States Patent [19]

Tomoike

[11] **Patent Number: 6,016,318**

[45] **Date of Patent: *Jan. 18, 2000**

[54] **VIRTUAL PRIVATE NETWORK SYSTEM OVER PUBLIC MOBILE DATA NETWORK AND VIRTUAL LAN**

[75] Inventor: **Hiroyuki Tomoike**, Tokyo, Japan

[73] Assignee: **NEC Corporation**, Tokyo, Japan

[*] Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

[21] Appl. No.: **08/892,280**

[22] Filed: **Jul. 14, 1997**

[30] **Foreign Application Priority Data**

Jul. 12, 1996 [JP] Japan 8-203015

[51] **Int. Cl.**^{7} **H04J 3/24**

[52] **U.S. Cl.** **370/401**; 370/338; 370/402; 709/249

[58] **Field of Search** 370/338, 329, 370/331, 401, 402; 455/517; 709/249

[56] **References Cited**

U.S. PATENT DOCUMENTS

H1641	4/1997	Sharman	370/338
4,823,338	4/1989	Chan et al.	370/522
5,159,592	10/1992	Perkins	370/338
5,442,633	8/1995	Perkins et al.	370/331
5,533,029	7/1996	Gardner	370/329
5,600,644	2/1997	Chang et al.	370/404
5,636,216	6/1997	Fox et al.	370/402
5,737,525	4/1998	Picazo, Jr. et al.	370/351
5,771,459	6/1998	Demery et al.	455/517
5,812,552	9/1998	Arora et al.	370/401

FOREIGN PATENT DOCUMENTS

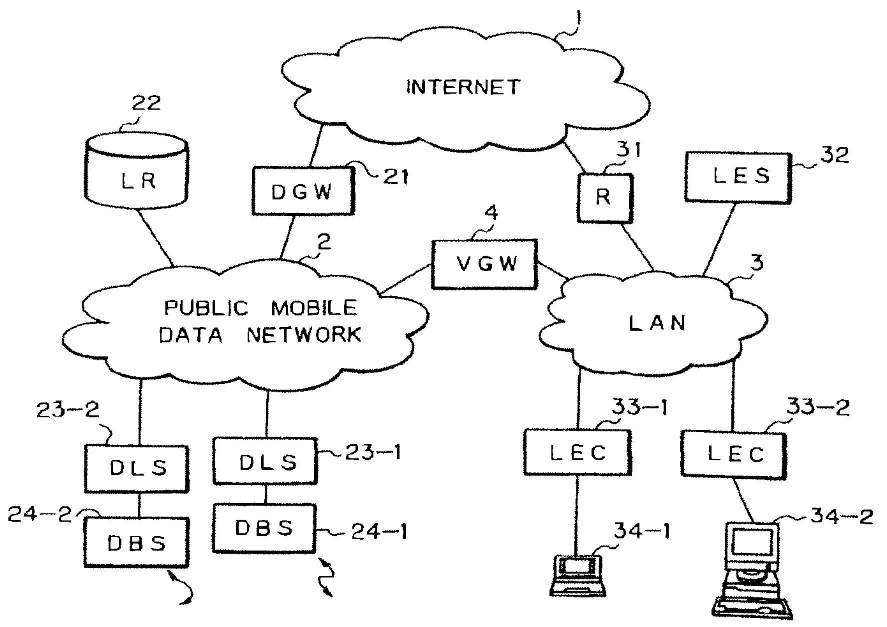
5-327720	12/1993	Japan .
6-337824	12/1994	Japan .
7-170286	7/1995	Japan .
7-212364	8/1995	Japan .
9-130405	5/1997	Japan .
9-261265	10/1997	Japan .

Primary Examiner—Zarni Maung
Assistant Examiner—Philip B. Tran
Attorney, Agent, or Firm—Foley & Lardner

[57] **ABSTRACT**

In a virtual private network system accessed by an internet, a virtual local area network (LAN) is connected to a LAN emulation server and IAN emulation clients, and a router is connected between the internet and the virtual LAN. Also, a public mobile data network is connected to a location register and mobile data subscriber processing units, and a data gateway is connected between the internet and the public mobile data networks Further, a virtual private network gateway is connected between the virtual LAN and the public mobile data network. A mobile data terminal having one IP address and one public network address and can be connected to either one of the LAN emulation clients or one of the mobile data subscriber processing units.

12 Claims, 10 Drawing Sheets



US006353614B1

(12) **United States Patent**
Borella et al.

(10) **Patent No.:** **US 6,353,614 B1**
(45) **Date of Patent:** **Mar. 5, 2002**

(54) **METHOD AND PROTOCOL FOR DISTRIBUTED NETWORK ADDRESS TRANSLATION**

(75) Inventors: **Michael S. Borella**, Naperville; **David Grabelsky**, Skokie; **Ikhlaq Sidhu**, Vernon Hills, all of IL (US); **Brian D. Petry**, San Diego, CA (US)

(73) Assignee: **3Com Corporation**, Santa Clara, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/035,600**

(22) Filed: **Mar. 5, 1998**

(51) **Int. Cl.**^{7} **H04L 12/28**; H04L 12/52

(52) **U.S. Cl.** **370/389**; 370/401; 370/474; 370/475; 709/238

(58) **Field of Search** 370/351, 352, 370/353, 354, 355, 356, 389, 392, 474, 475, 476, 400, 401; 709/217, 218, 219, 238, 245, 223

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,159,592 A		10/1992	Perkins	
5,227,778 A		7/1993	Vacon et al.	
5,526,489 A	*	6/1996	Nilakantan et al.	709/228
5,550,984 A		8/1996	Gelb	
5,636,216 A		6/1997	Fox et al.	
5,708,655 A		1/1998	Toth et al.	
5,793,763 A		8/1998	Mayes et al.	
5,812,819 A		9/1998	Rodwin et al.	
5,867,660 A		2/1999	Schmidt et al.	
5,872,847 A		2/1999	Boyle et al.	
6,011,782 A	*	1/2000	DeSimone et al.	370/260
6,055,236 A		4/2000	Nessett et al.	
6,157,950 A	*	12/2000	Krishnan	709/223

OTHER PUBLICATIONS

Tsirtsis, George, O'Neill, Alan, Internet Engineering Task Force, Internet Draft, "NAT Bypass for End 2 End 'sensitive'applications", <draft-tsirtsis-nat-bypass-00.txt>, Jan. 1998, pp. 1 to 6.

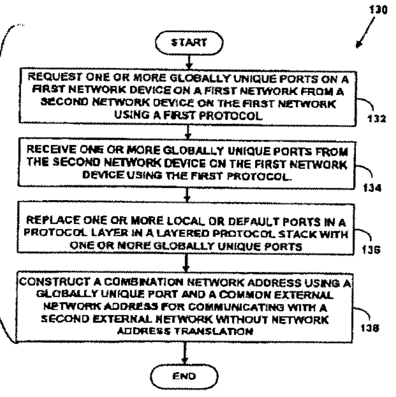
(List continued on next page.)

Primary Examiner—Ajit Patel
Assistant Examiner—Bob A. Phunkulh
(74) *Attorney, Agent, or Firm*—McDonnell Boehnen Hulbert & Berghoff; Steven Lesavich

(57) **ABSTRACT**

A method and protocol for Distributed Network Address Translation ("DNAT") is provided. DNAT is used to overcome the limited address 32-bit address space used for versions of the Internet Protocol ("IP"). DNAT is used with small office or home office networks or other legacy local network that have multiple network devices using a common external network address to communicate with an external network. The protocol includes a port allocation protocol to allocate globally unique ports to network devices on a local computer network. The globally unique ports are used in a combination network address with a common external network address such as an IP address, to identify multiple network devices on a local network to an external network such as the Internet, an intranet, or a public switched telephone network. The method includes requesting one or more globally unique ports from network devices on a local network, receiving the ports, and replacing local ports with the globally unique ports. The network devices on the local network use the combination network address with the common external network address and the globally unique port to uniquely identify themselves during communications with an external network. DNAT overcomes the large computation burdens encountered when network address translation is done by a router for multiple network devices on a local network using a common external network address and simplifies routers since a router in a DNAT system does not have to support multiple individual protocols. DNAT helps extend the life of versions of IP using 32-bit addressing, allows a local network to efficiently switch between external network service providers and allows a local network to purchase a smaller block of external network addresses.

42 Claims, 10 Drawing Sheets


130
START
REQUEST ONE OR MORE GLOBALLY UNIQUE PORTS ON A FIRST NETWORK DEVICE ON A FIRST NETWORK FROM A SECOND NETWORK DEVICE ON THE FIRST NETWORK USING A FIRST PROTOCOL
132
RECEIVE ONE OR MORE GLOBALLY UNIQUE PORTS FROM THE SECOND NETWORK DEVICE ON THE FIRST NETWORK DEVICE USING THE FIRST PROTOCOL
134
REPLACE ONE OR MORE LOCAL OR DEFAULT PORTS IN A PROTOCOL LAYER IN A LAYERED PROTOCOL STACK WITH ONE OR MORE GLOBALLY UNIQUE PORTS
136
CONSTRUCT A COMBINATION NETWORK ADDRESS USING A GLOBALLY UNIQUE PORT AND A COMMON EXTERNAL NETWORK ADDRESS FOR COMMUNICATING WITH A SECOND EXTERNAL NETWORK WITHOUT NETWORK ADDRESS TRANSLATION
138
END

OIPE JC
FEB 2 2 2002
PATENT & TRADEMARK OFFICE

PATENT

#9/C
3-18-02

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

RECEIVED
MAR 0 1 2002
Technology Center 2100

In re Application of:	
Edmund Colby MUNGER *et al.*	
Application No. 09/504,783	Group Art Unit: 2153
Filed: February 15, 2000	Examiner: K. Lim
For: IMPROVEMENTS TO AN AGILE NETWORK PROTOCOL FOR SECURE COMMUINCATIONS WITH ASSURED SYSTEM AVAILABILITY	Atty Docket: 00479.85672

SECOND PRELIMINARY AMENDMENT

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Applicants submit the following amendment and request its entry prior to examination of the claims. The Office is authorized to charge any required fees for consideration of this paper to our Deposit Account No. 19-0733.

IN THE CLAIMS:

Please add the following new claims:

C1

82. A data processing device, comprising memory storing a domain name server (DNS) proxy module that intercepts DNS requests sent by a client and, for each intercepted DNS request, performs the steps of:

03/18/2002 EWARREN 00000001 190733 09504783
01 FC:102 252.00 CH
02 FC:103 180.00 CH

02/27/2002 JBALINAN 00000096 190733 09504783
01 FC:102 84.00 CH

~~(i) determining whether the intercepted DNS request corresponds to a secure server;~~

~~(ii) when the intercepted DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a nonsecure computer; and~~

~~(iii) when the intercepted DNS request corresponds to a secure server, automatically initiating an encrypted channel between the client and the secure server.~~

C1

~~83. The data processing device of claim 82, wherein step (iii) comprises the steps of:~~

~~(a) determining whether the client is authorized to access the secure server; and~~

~~(b) when the client is authorized to access the secure server, sending a request to the secure server to establish an encrypted channel between the secure server and the client.~~

~~84. The data processing device of claim 83, wherein step (iii) further comprises the step of:~~

~~(c) when the client is not authorized to access the secure server, returning a host unknown error message to the client.~~

~~85. The data processing device of claim 84, wherein the client comprises a web browser into which a user enters a URL resulting in the DNS request.~~

86. A data processing device, comprising memory storing a domain name server (DNS) proxy module that intercepts DNS requests sent by a client and, for each intercepted DNS request, when the intercepted DNS request corresponds to a secure server, determines whether the client is authorized to access the secure server and, if so, automatically initiates an encrypted channel between the client and the secure server.

87. A computer readable medium storing a domain name server (DNS) proxy module comprised of computer readable instructions that, when executed, cause a data processing device to perform the steps of:

C1

(i) intercepting a DNS request sent by a client;

(ii) determining whether the intercepted DNS request corresponds to a secure server;

(iii) when the intercepted DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a nonsecure computer; and

(iv) when the intercepted DNS request corresponds to a secure server, automatically initiating an encrypted channel between the client and the secure server.

88. The computer readable medium of claim 87, wherein step (iii) comprises the steps of:

(a) determining whether the client is authorized to access the secure server; and

(b) when the client is authorized to access the secure server, sending a request to the secure server to establish an encrypted channel between the secure server and the client.

89. The computer readable medium of claim 88, wherein step (iii) further comprises the step of:

(c) when the client is not authorized to access the secure server, returning a host unknown error message to the client.

C

90. The computer readable medium of claim 89, wherein the client comprises a web browser into which a user enters a URL resulting in the DNS request.

91. A computer readable medium comprising computer readable instructions that, when executed, cause a domain name server (DNS) proxy module to intercept DNS requests sent by a client and, for each intercepted DNS request, when the intercepted DNS request corresponds to a secure server, determines whether the client is authorized to access the secure server and, if so, automatically initiates an encrypted channel between the client and the secure server.

Remarks

Applicants have added new claims 82 - 91 to more completely claim the disclosed invention. Support for the new claims may be found at least on pages 59-60 and in FIG. 26.

If the Examiner has any questions or wishes to discuss this amendment, the Examiner is invited to telephone the undersigned representative at the number set forth below.

Respectfully submitted,

BANNER & WITCOFF, LTD.

U.S.P.T.O.
Reg. No. 49,024

Date: 2/22/02 By: [illegible]

Bradley C. Wright
Registration No. 38,061

11th Floor
1001 G Street N.W.
Washington, D.C. 20001
(202) 508-9100

2153 ~~#~~

PTO/SB/21 (08-00)
Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Please type a plus sign (+) inside this box → [+]

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

OIPE JC186 FEB 22 2002 PATENT & TRADEMARK OFFICE

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/504,783
Filing Date	February 15, 2000
First Named Inventor	Edmund Colby Munger
Group Art Unit	2153
Examiner Name	Krisna Lim
Attorney Docket Number	00479.85672
Total Number of Pages in This Submission	

RECEIVED MAR 01 2002 Technology Center 2100

ENCLOSURES *(check all that apply)*

- ☒ Fee Transmittal Form
  - ☐ Fee Attached
- ☐ Amendment / Response
  - ☐ After Final
  - ☐ Affidavits/declaration(s)
- ☐ Extension of Time Request
- ☐ Express Abandonment Request
- ☒ Information Disclosure Statement
- ☐ Certified Copy of Priority Document(s)
- ☐ Response to Missing Parts/ Incomplete Application
  - ☐ Response to Missing Parts under 37 CFR 1.52 or 1.53
- ☐ Assignment Papers *(for an Application)*
- ☐ Drawing(s)
- ☐ Licensing-related Papers
- ☐ Petition
- ☐ Petition to Convert to a Provisional Application
- ☐ Power of Attorney, Revocation Change of Correspondence Address
- ☐ Terminal Disclaimer
- ☐ Request for Refund
- ☐ CD, Number of CD(s) ______
- ☐ After Allowance Communication to Group
- ☐ Appeal Communication to Board of Appeals and Interferences
- ☐ Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*
- ☐ Proprietary Information
- ☐ Status Letter
- ☒ Other Enclosure(s) *(please identify below)*:
  Second Preliminary Amendment

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm or Individual name	Bradley C. Wright, Reg. No. 38,061 **U.S.P.T.O. Reg. No. 49,024**
Signature	Ross D[illegible]
Date	February 22, 2002

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on this date:

Typed or printed name			
Signature		Date	

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be send to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

PTO/SB/17 (10-01)
Approved for use through 10/31/2002. OMB 0651-0032
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 19 persons are required to respond to a collection of information unless it displays a valid OMB control number.

OIPE
FEB 2 2 2002
PATENT & TRADEMARK OFFICE

FEE TRANSMITTAL for FY 2002

Patent fees are subject to annual revision.

TOTAL AMOUNT OF PAYMENT ($) 84

Complete if Known	
Application Number	09/504,783
Filing Date	February 15, 2000
First Named Inventor	Edmund Colby Munger
Examiner Name	K. Lim
Group / Art Unit	2153
Attorney Docket No.	000479.85672

RECEIVED
MAR 0 1 2002
Technology Center 2100

METHOD OF PAYMENT (check all that apply)

☐ Check ☐ Credit card ☐ Money Order ☐ Other ☐ None
☒ Deposit Account:

Deposit Account Number	19-0733
Deposit Account Name	Banner & Witcoff, Ltd.

The Commissioner is authorized to: *(check all that apply)*
☒ Charge fee(s) indicated below ☒ Credit any overpayments
☐ Charge any additional fee(s) during the pendency of this application
☐ Charge fee(s) indicated below, **except for the filing fee**
to the above-identified deposit account.

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
101	740	201	370	Utility filing fee	
106	330	206	165	Design filing fee	
107	510	207	255	Plant filing fee	
108	740	208	370	Reissue filing fee	
114	160	214	80	Provisional filling fee	

SUBTOTAL (1) ($) 0

2. EXTRA CLAIM FEES

				Extra Claims		Fee from below		Fee Paid
Total Claims	37	- ~~21~~ 27**	=	~~0~~ 10	X	18	=	~~$~~ 180
Independent Claims	~~10~~ 8	- ~~3~~ 4**	=	14 [illegible]	X	84	=	~~84~~ 336
Multiple Dependent					X		=	$516

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
103	18	203	9	Claims in excess of 20
102	84	202	42	Independent claims in excess of 3
104	280	204	140	Multiple dependent claim, if not paid
109	84	209	42	** Reissue independent claims over original patent
110	18	210	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($) 84

**or number previously paid, if greater; For Reissues, see above

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
105	130	205	65	Surcharge - late filing fee or oath	
127	50	227	25	Surcharge - late provisional filing fee or cover sheet.	
139	130	139	130	Non-English specification	
147	2,520	147	2,520	For filing a request for reexamination	
112	920*	112	920*	Requesting publication of SIR prior to Examiner action	
113	1,840*	113	1,840*	Requesting publication of SIR after Examiner action	
115	110	215	55	Extension for reply within first month	
116	400	216	200	Extension for reply within second month	
117	920	217	460	Extension for reply within third month	
118	1,440	218	720	Extension for reply within fourth month	
128	1,960	228	980	Extension for reply within fifth month	
119	320	219	160	Notice of Appeal	
120	320	220	160	Filing a brief in support of an appeal	
121	280	221	140	Request for oral hearing	
138	1,510	138	1,510	Petition to institute a public use proceeding	
140	110	240	55	Petition to revive – unavoidable	
141	1,280	241	640	Petition to revive – unintentional	
142	1,280	242	640	Utility issue fee (or reissue)	
143	460	243	230	Design issue fee	
144	620	244	310	Plant issue fee	
122	130	122	130	Petitions to the Commissioner	
123	50	123	50	Processing fee under 37 CFR 1.17 (q)	
126	180	126	180	Submission of Information Disclosure Stmt	
581	40	581	40	Recording each patent assignment per property (times number of properties)	
146	740	246	370	Filing a submission after final rejection (37 CFR § 1.129(a))	
149	740	249	370	For each additional invention to be examined (37 CFR § 1.129(b))	
179	740	279	370	Request for Continued Examination (RCE)	
169	900	169	900	Request for expedited examination of a design application	

Other fee (specify) _____

*Reduced by Basic Filing Fee Paid

SUBTOTAL (3) ($) 0

SUBMITTED BY				Complete *(if applicable)*	
Name (Print/Type)	Bradley C. Wright	*Registration No. Attorney/Agent*	38,061 U.S.P.T.O.	*Telephone*	(202) 508-9160
Signature	[illegible]		Reg No 49,024	*Date*	February 22, 2002

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

O I P E JC186
FEB 21 2002
PATENT & TRADEMARK OFFICE

PATENT APPLICATION

#10

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re Application Of

Edmond Colby Munger et al.

Serial No.: 09/504,783

Filed: February 15, 2000

For: IMPROVEMENTS TO AN AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS WITH ASSURED SYSTEM AVAILABILITY

Group Art Unit: 2153

Examiner: K. Lim

Atty. Dkt. No. 00479.85672

RECEIVED
MAR 01 2002
Technology Center 2100

SUPPLEMENTAL INFORMATION DISCLOSURE STATEMENT

Assistant Commissioner for Patents and Trademarks
Washington, D.C. 20231

Sir:

In accordance with 37 C.F.R. § 1.97(c) and 1.98, enclosed is a PTO Form-1449 listing art for consideration by the Examiner and a copy of each of the identified documents. It is believed no fee is required to make this a complete and timely filing. However, if a fee is required, please charge our Deposit Account No. 19-0733.

Consideration of this information is respectfully requested.

The submission of the listed document is not intended as an admission that any such document constitutes prior art against the claims of the present application. Applicant does not waive any right to take any action that would be appropriate to antedate or otherwise remove any listed document as a competent reference against the claims of the present application.

Respectfully submitted,

BANNER & WITCOFF, LTD.

U.S.P.T.O. Reg. No. 49,024

By: [illegible]
for Bradley C. Wright
Registration No. 38,061

1001 G. Street, N.W.
Washington, D.C. 20001-4597
(202) 508-9100
Dated: 2/22/02

BCW/RAD/mmd

O.I.P.E. JC186
FEB 22 2002
PATENT & TRADEMARK OFFICE

USPTO Form 1449 U.S. Department of Commerce Patent and Trademark Office INFORMATION DISCLOSURE CITATION Sheet 1 of 1	Attorney Docket No. 00479.85672	Serial No. 09/504,783
	Applicant(s): Edmund Colby Munger et al.	
	Filing Date: February 15, 2000	Group: 2153

U.S. PATENT DOCUMENTS

Examiner Initial	Patent No.	Date	Name	Class	Subclass	Filing Date (if appropriate)
KL	6,243,749	6/5/01	Sitaraman et al.			
KL	6,119,171	9/12/00	Alkhatib			
KL	6,052,788	4/18/00	Wesinger, Jr. et al.			
KL	6,006,259	12/21/99	Adelman et al.			
KL	5,905,859	5/18/99	Holloway et al.			
KL	5,898,830	4/27/99	Wesinger, Jr. et al.			
KL	5,892,903	4/6/99	Klaus			
KL	5,805,801	9/8/98	Holloway et al.			
KL	5,796,942	8/18/98	Esbensen			

FOREIGN PATENT DOCUMENTS

Examiner Initial	Document No.	Date	Country	Class	Subclass	Translation YES	Translation NO
KL	WO 00/70458	11/23/00	PCT				

OTHER DOCUMENTS (including Author, Title, Date, Pertinent Pages, etc.)

KL	Linux FreeS/WAN Index File, printed from http://liberty.freeswan.org/freeswan_trees/freeswan-1.3/doc/ on February 21, 2002, 3 Pages
KL	J. Gilmore, "Swan: Securing the Internet against Wiretapping", printed from http://liberty.freeswan.org/freeswan_trees/freeswan-1.3/doc/rationale.html on February 21, 2002, 4 pages
KL	Glossary for the Linux FreeS/WAN project, printed from http://liberty.freeswan.org/freeswan_trees/freeswan-1.3/doc/glossary.html on February 21, 2002, 25 pages
KL	Alan O. Frier et al., "The SSL Protocol Version 3.0", November 18, 1996, printed from http://www.netscape.com/eng/ss13/draft302.txt on February 4, 2002, 56 pages

EXAMINER KRISNA LIM	DATE CONSIDERED 6/28/02

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to Applicant.

US006243749B1

(12) **United States Patent**
Sitaraman et al.

(10) **Patent No.: US 6,243,749 B1**
(45) **Date of Patent: Jun. 5, 2001**

(54) DYNAMIC NETWORK ADDRESS UPDATING

(75) Inventors: **Aravind Sitaraman**, Santa Clara; **Jane Jiaying Jin**, San Jose; **Maria Alice Dos Santos**, Redwood City; **Sampath Kumar Sthothra Bhasham**, Santa Clara, all of CA (US)

(73) Assignee: **Cisco Technology, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/169,182**

(22) Filed: **Oct. 8, 1998**

(51) **Int. Cl.**^{7} **G06F 13/00**

(52) **U.S. Cl.** **709/223**; 709/220; 709/238; 709/245

(58) **Field of Search** 709/217, 220, 709/223, 225, 227, 228, 229, 230, 238, 245

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,769,810	9/1988	Eckberg, Jr. et al.	370/60
4,769,811	9/1988	Eckberg, Jr. et al.	370/60
4,933,937	6/1990	Konishi	370/85.13
5,014,265	5/1991	Hahne et al.	370/60
5,224,099	6/1993	Corbalis et al.	370/94.2
5,367,517	11/1994	Cidon et al.	370/54
5,423,002	6/1995	Hart	395/200
5,430,715	7/1995	Corbalis et al.	370/54
5,509,006	4/1996	Wilford et al.	370/60
5,570,360	10/1996	Klausmeier et al.	370/60
5,592,470	1/1997	Rudrapatna et al.	370/320
5,668,857	9/1997	McHale	379/93.07
5,678,006	10/1997	Valizadeh et al.	395/200.02
5,699,521	12/1997	Iizuka et al.	395/200.15
5,734,654	3/1998	Shirai et al.	370/396
5,805,595	9/1998	Sharper et al.	370/442
5,835,720	11/1998	Nelson et al.	395/200.54

(List continued on next page.)

OTHER PUBLICATIONS

Active Software, Inc., "Active Software's Integration System", printed from http://www.activesw.com/products/products.html, on Jul. 24, 1998.

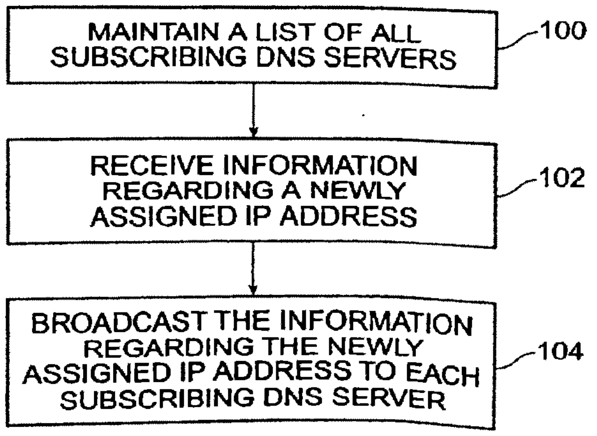
(List continued on next page.)

Primary Examiner—Viet D. Vu
(74) *Attorney, Agent, or Firm*—Marc S. Hanish; Thelen Reid & Priest LLP

(57) **ABSTRACT**

An information broker is provided which receives information regarding the updating of IP addresses and distributes the information to subscribing Domain Name Service (DNS) or Dynamic Host Configuration Protocol (DHCP) servers. A list of subscribing servers is maintained by the broker. The broker may broadcast information regarding the allocation of a IP addresses to subscribing DNS servers, which then may be added to the DNS databases or the database may be updated with the new information. The broker may also broadcast information regarding the revocation of IP addresses to subscribing DNS servers, which may then be used to clear DNS entries in the database. Revocation of dynamically allocated IP addresses in networks with multiple DHCP servers may also be simplified by using the broker, where the broker broadcasts information regarding the revocation of IP addresses to subscribing DHCP servers. Utilization of the broker within a segment of the Internet allows a user to determine the dynamically allocated IP address of a user within the segment simply by making a standard DNS query.

98 Claims, 2 Drawing Sheets


MAINTAIN A LIST OF ALL SUBSCRIBING DNS SERVERS
100
RECEIVE INFORMATION REGARDING A NEWLY ASSIGNED IP ADDRESS
102
BROADCAST THE INFORMATION REGARDING THE NEWLY ASSIGNED IP ADDRESS TO EACH SUBSCRIBING DNS SERVER
104

U.S. PATENT DOCUMENTS

5,835,725		11/1998	Chiang et al.	395/200.58
5,835,727		11/1998	Wong et al.	395/200.68
5,838,683		11/1998	Corley et al.	370/408
5,845,070		12/1998	Ikudome	395/187.01
5,852,607		12/1998	Chin	370/401
5,854,901		12/1998	Cole et al.	395/200.75
5,898,780		4/1999	Liu et al.	380/25
5,918,016		6/1999	Brewer et al.	395/200.5
5,926,458		7/1999	Yin	370/230
5,974,453		10/1999	Andersen et al.	709/220
6,009,103	*	12/1999	Woundy	370/401
6,055,236		4/2000	Nessett et al.	370/389
6,091,951	*	7/2000	Sturniolo et al.	455/432
6,092,178	*	7/2000	Jindal et al.	712/27

OTHER PUBLICATIONS

Ascend Communications, Inc., "Access Control Product Information", 4 pages, unknown date.
Ascend Communications, Inc., "Remote Access Network Security", printed from http://www.ascend.com/1103.html, on Jul. 24, 1998, pp. 1–8.
Droms, R., "Dynamic Host Configuration Protocol," Network Working Group, RFC 1531, Oct. 1993.
NAT and Networks, printed from http://www.csn.tu-chemnitz.de/~mha/linux-ip-nat/diplom/node4.html, on Sep. 19, 1998.
"NAT–PC Webopaedia Definition and Links", 1998, Mecklermedia Corporation, printed from http://webopedia.internet.com/TERM/N/NAT.html, on Sep. 19, 1998, 1 page.

"Network Address Translation Information", printed from http://www.uq.edu.au/~gadmacka/content/natinformation.htm, on Sep. 19, 1998.

Network Registrar, "Regain Confidence and Control Over Your IP Address Infrastructure", American Internet Corporation, Bedford, MA, 1998.

Network Registrar, "Hot Products & Solutions", American Internet Corporation, printed from http://www.american.com/networkregistrar, html, on Jul. 24, 1998.

Network Registrar, "Hot Products & Solutions—IP Address Management: A White Paper", American Internet Corporation, Bedford, MA, printed from http://www.american.com/ip-mgmt.html, on Jul. 24, 1998.

Network Registrar, "Hot Products & Solutions—Deploying Class of Service Using Network Registrar", American Internet Corporation, Bedford, MA, printed from http://american.com/applicationCOS-network.html, on Jul. 24, 1998.

Rigney, et al., "Remote Authentication Dial In User Service (RADIUS)", Network Working Group, RFC 2138, Apr. 1997, pp. 1–57.

"Three Ways to Manage IP Addresses", PC Magazine: IP Address Management, printed from http://www.zdnet.com/pcmag/features/ipmanage/ip-s2.htm, on Sep. 10, 1998.

* cited by examiner

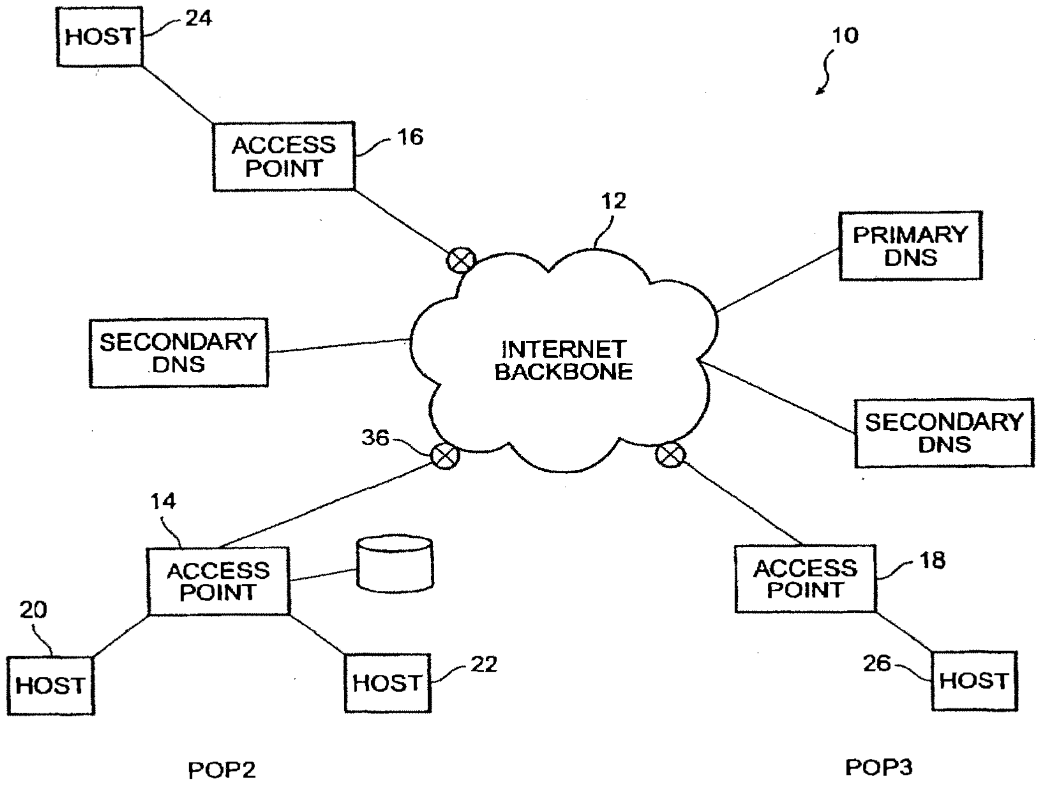


FIG. 1
(PRIOR ART)

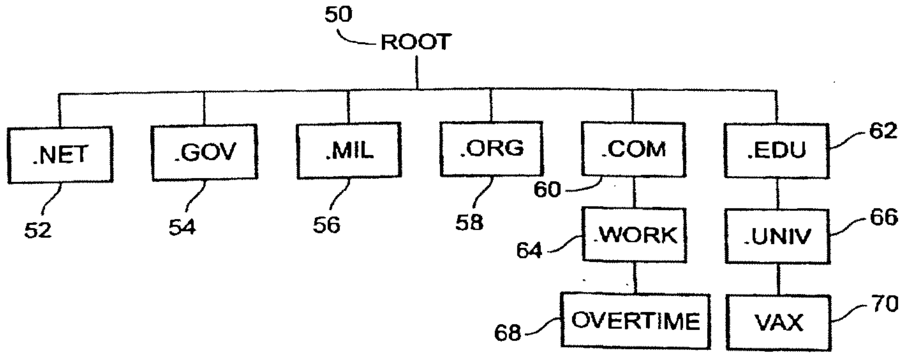


FIG. 2
(PRIOR ART)

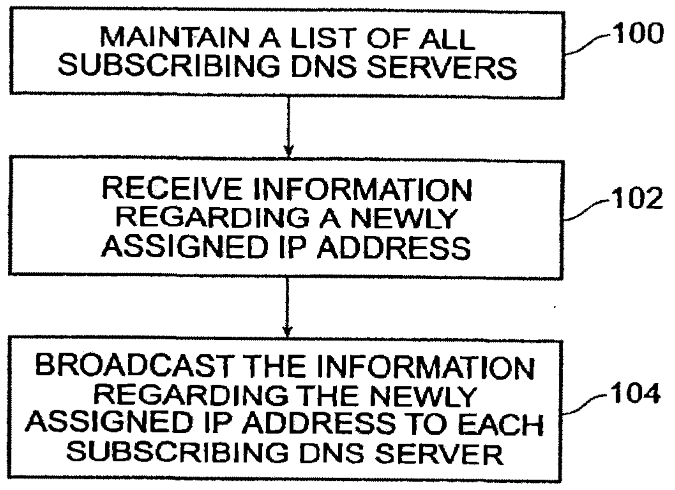

MAINTAIN A LIST OF ALL SUBSCRIBING DNS SERVERS
100
RECEIVE INFORMATION REGARDING A NEWLY ASSIGNED IP ADDRESS
102
BROADCAST THE INFORMATION REGARDING THE NEWLY ASSIGNED IP ADDRESS TO EACH SUBSCRIBING DNS SERVER
104

FIG. 3

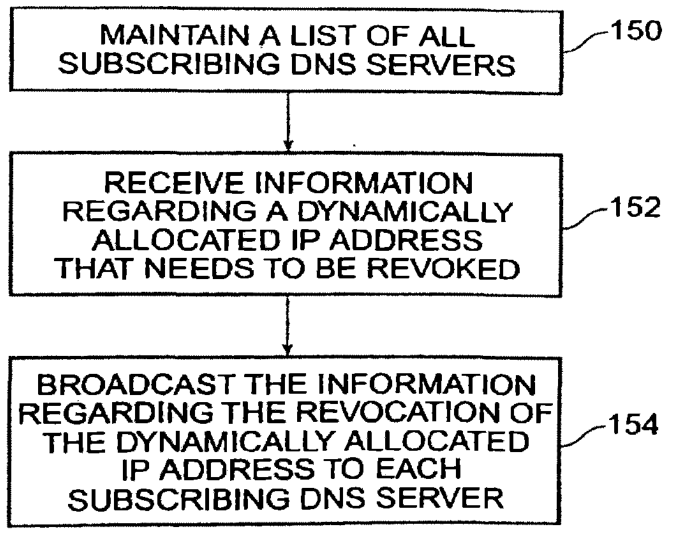

MAINTAIN A LIST OF ALL SUBSCRIBING DNS SERVERS
150
RECEIVE INFORMATION REGARDING A DYNAMICALLY ALLOCATED IP ADDRESS THAT NEEDS TO BE REVOKED
152
BROADCAST THE INFORMATION REGARDING THE REVOCATION OF THE DYNAMICALLY ALLOCATED IP ADDRESS TO EACH SUBSCRIBING DNS SERVER
154

FIG. 4

DYNAMIC NETWORK ADDRESS UPDATING

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to managing host addresses. More particularly, the present invention relates to managing dynamically allocated host addresses to allow subscribers to reliably locate other subscribers who have been dynamically allocated host addresses.

2. The Background

As shown in FIG. 1, the Internet, or any large computer network, **10** may be described as a group of interconnected computing networks (not shown) that are tied together through a backbone **12**. The computing networks, in turn, provide access points, such as access points **14**, **16** and **18**, through which users may connect to the Internet via a station (a computer having a connection to a network) or host, such as hosts **20**, **22**, **24**, and **26**. An access point is essentially a location on the Internet that permits access to the Internet. An access point may include a modem pool (not shown) maintained by an ISP (Internet Services Provider) which enables its subscribers to obtain Internet access through a host having a dial-up connection. Those of ordinary skill in the art will recognize that other types of access methods may be provided by an ISP such as frame relay, leased lines, ATM (asynchronous transfer mode), ADSL, and the like.

Regardless of the access method used, each device (e.g., a host or router) that receives, sends and/or routes information between or among other devices on Internet **10** is configured to communicate with other devices using a communication protocol that may be understood by the other devices. The current communication protocol used by these devices on the Internet is TCP/IP (transmission control protocol/internet protocol). In addition, each device that can send or receive information (e.g., a host device) must also have a unique host address. The type of host address used for the Internet, or an equivalent switched network that uses TCP/IP, is commonly referred to as an IP address. A standard TCP/IP address is 4 bytes (32 bits) in length, providing a total of 2^{32} possible IP addresses. Those of ordinary skill in the art will readily recognize that not all of these possible IP addresses are available due to administrative expediencies, such as reserving blocks of IP addresses for future use.

Sending or receiving information using the TCP/IP protocol requires encapsulating information into packets. Each packet includes a header and a payload. The header contains information related to the handling of the payload by a receiving host or routing device, while the payload contains part or all of the user information. The information in the header includes the sender's and the recipient's addresses and is used to route the packet through the Internet until the packet is received by a host having an IP address that matches the packet's destination address (when referring to the source address and destination address of a packet, the source address and destination address are commonly referred to as "SA" and "DA", respectively). This enables users to accurately send and receive information with each other through their respective host computers.

By implementing a protocol common to all devices using Internet **10**, users may send and receive information with other users on the Internet in a seamless manner regardless of geographic location or the type of host and/or interconnected network used. While IP addresses themselves are in numerical form, in order to make navigating the sea of addresses simpler, the Domain Name Service (DNS) was formed. DNS enables the central managing of host names to

IP addresses. It is actually a distributed database which allows for the dissemination of new host information as needed. There are a great many DNS servers distributed throughout the Internet, and most large ISPs maintain their own DNS server.

FIG. 2 is a diagram illustrating the DNS hierarchy, which is similar to that of a computer file system, At the top of the hierarchy is the root domain **50**, which consists of a group of root servers to service the top-level domains. The top level domains are separated into organizational and geographical domains. Many countries have their own top-level domains, such as .uk for the United Kingdom, .de for Germany, and jp for Japan (not shown). The United States has no country-specific top-level domain, but is the main user of the six organizational top-level domains, which are net for network support organizations **52**, .gov for government agencies **54**, mil for military users **56**, org for not for profit organizations **58**, .com for commercial enterprises **60**, and .edu for educational facilities **62**. There are also a near infinite number of lower level domains. Each level of domain names may have another level of domain names below it. For example, a lower level domain .work **64** may be located under the .com domain **60**, and the lower level domain .univ **66** may be located under the .edu domain **62**. At the lowest level are the hosts. For example, the host labeled overtime **68** may be located under the .work sub-domain under the .com domain while the host labeled vax **70** may be located under the .univ sub-domain under the .edu domain. The proper way to read these two DNS host names would then be overtime.work.com and vax.univ.edu.

The steps of locating an IP address from a host, sub-domain, and domain name proceeds as in the following example. If a user in the vax.univ.edu domain wishes to contact a user with the user name sun in the work.com domain, the first step is to contact its own DNS server. Therefore, if the vax.univ.edu host is configured with a DNS server at the IP address 133.3.1.3, the user sends a DNS request to that IP address. The DNS server then searches for the entry in its database. Generally, DNS servers only maintain a database of host addresses (or sub-domain names) within its own subnet. Therefore, the DNS server would look for an IP address corresponding to the domain/sub-domain combination .univ.edu. It may or may not have information that precise. It may only have information regarding the IP address of the .com domain and not the .work.com domain. If it has information about the IP address of the DNS server of the .work.com domain, it passes this information to the user, which then contacts the .work.com DNS server and requests the IP address of the precise user it wishes to contact in the .work.com domain. If however, the DNS server associated with the vax.univ.edu host only has information about the address of the DNS server of the .com domain, it returns only that address, and the user must recursively navigate down the branches of DNS servers in the com domain until locating the address it needs (in the present example, it only searches down one level, but in more complicated hierarchies it may need to search through many levels of DNS servers).

It is also possible that a higher level DNS server will simply forward the request packet down the hierarchy and wait to inform the user of the host address until it hears back from the lower level DNS server, thus avoiding having to contact the user at each step in the hierarchy. However, this still presents the problem of recursing, which increases the complexity of a search.

The dramatic increase in popularity of the Internet in recent years has created a concern about the number of available IP addresses. ISPs and domains are generally allocated a finite number of IP addresses. The ISPs and domains, therefore, are constantly looking for ways to limit the number of IP addresses they use while still providing access to the greatest number of users.

One solution for mitigating the effect of the number of users requiring host addresses is to dynamically allocate host addresses for users who do not have dedicated connections to the Internet, such as users who use dial-up access methods to connect to an ISP. Dynamic allocation of IP addresses entails having a pool of IP addresses, such as IP address pool, from which an ISP can draw from each time a valid subscriber (who does not use a dedicated connection or a connection that does not have a framed IP address, i.e., a static IP address) seeks to access the Internet. Once the subscriber logs on to an ISP and is properly authenticated, the ISP allocates an IP address for use by the user. This task is normally performed by a Dynamic Host Configuration Protocol (DHCP) server existing on the ISP (or other local segment of the Internet).

Upon log-off, the DHCP server releases the assigned/allocated IP address, rendering that IP address available for subsequent use by another user. In this way, a set of IP addresses can be used to provide access to a number of users that exceed the number of IP address comprising the IP address pool, assuming that at any given time the number of users seeking to log-on and obtain dynamic IP addresses is less than or equal to the number of IP address available in the IP address pool.

Recently, software advances have allowed users to begin to merge existing technologies, like telephone service, into their Internet service. One example of this phenomenon is a utility known as Internet Phone. With the Internet Phone utility installed on his computer, a user may "dial" a friend's computer and speak (either through a microphone connected to the computer or through an integrated telephone) with his friend, who has a similar system. Communication is accomplished over the Internet utilizing a protocol called Voice over IP (VoIP). VoIP utilizes IP packets to carry digital audio transmissions. Through data compression techniques (which includes filtering out much o the silences that accompany most conversations), it is possible to conduct real-time conversations through the Internet.

Another example of the technology-merging phenomenon is in Internet Chat. Internet Chat is similar to e-mail in that users type messages to one another on their screens. However, and Internet Chat session takes place in real-time. Therefore, when a user types a sentence on his screen and presses <enter>, the message is transmitted instantaneously to the recipient, who then may respond to the message. The recipient may then respond in a similar fashion, creating a real-time, typed "conversation".

A problem arises in using these technologies when a user wishes to initiate a conversation or chat session with a dial-up user. There is currently no way for a system to resolve a dynamically assigned destination address. Therefore, programs like Internet Chat or Internet Phone are virtually useless when used in conjunction with dial-up users. The one solution is to determine the users actual dynamic IP address. This, however, requires efforts on both parties to the conversation.

Additionally, it has become more and more common to have multiple DHCP servers, rather than a single DHCP server, for a single ISP or local segment of the Internet. These multiple DHCP servers are distributed throughout the ISP or local segment of the Internet and may contain different information.

Multiple DHCP servers may tend to create a problem with regards to revocation of dynamically allocated IP addresses. When an ISP determines it should revoke a dynamically allocated IP addresses (such as when a dial-up user disconnects from the ISP), it must then search each of the DHCP servers to make sure the address is removed from all DHCP servers which have stored the dynamically allocated address.

What is needed is a solution which overcomes the drawbacks of the prior art.

SUMMARY OF THE INVENTION

An information broker is provided which receives information regarding the updating of IP addresses and distributes the information to subscribing Domain Name Service (DNS) or Dynamic Host Configuration Protocol (DHCP) servers. A list of subscribing servers is maintained by the broker. The broker may broadcast information regarding the allocation of a IP addresses to subscribing DNS servers, which then may be added to the DNS databases or the database may be updated with the new information. The broker may also broadcast information regarding the revocation of IP addresses to subscribing DNS servers, which may then be used to clear DNS entries in the database. Revocation of dynamically allocated IP addresses in networks with multiple DHCP servers may also be simplified by using the broker, where the broker broadcasts information regarding the revocation of IP addresses to subscribing DHCP servers. Utilization of the broker within a segment of the Internet allows a user to determine the dynamically allocated IP address of a user within the segment simply by making a standard DNS query.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a prior art block diagram of the Internet.

FIG. 2 is a prior art block diagram illustrating the hierarchy of some top level domain names.

FIG. 3 is a flow diagram illustrating a method for managing IP addresses in a network including one or more Domain Name Service (DNS) servers in accordance with a presently preferred embodiment of the invention.

FIG. 4 is a flow diagram illustrating a method for managing IP addresses in a network including one or more Dynamic Host Configuration Protocol (DHCP) servers in accordance with a presently preferred embodiment of the invention.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

Those of ordinary skill in the art will realize that the following description of the present invention is illustrative only and not in any way limiting. Other embodiments of the invention will readily suggest themselves to such skilled persons.

The present invention utilizes a broker to publish information regarding dynamically allocated addresses to DNS and DHCP servers. The broker may also be used to update DNS servers regarding newly assigned static IP addresses.

FIG. 3 is a flow diagram illustrating a method for updating one or more DNS servers in accordance with a preferred embodiment of the present invention. At step **100**, the broker maintains a list of all subscribing DNS servers. A subscribing DNS server is any DNS server that wishes to be updated by the broker. Therefore, generally every DNS server maintained by an ISP will be a subscribing DNS server with regards to a broker maintained by the ISP. It is also possible for DNS servers outside the ISP to subscribe as well. When a new DNS server wishes to subscribe or an existing DNS server wishes to unsubscribe, it need simply send a message so indicating to the broker. At step **102**, the broker receives information regarding a newly assigned IP address. This information will most likely have been sent from a DHCP server in the case of dynamically allocated IP addresses and from a DNS server or ISP coordinator in the case of newly assigned static IP addresses.

At step **104**, the broker broadcasts the information regarding the newly assigned IP address to each subscribing DNS server. Each DNS server may then add the newly assigned IP address to its database. In the case of dynamically allocated IP address, it is common for the newly assigned IP address to be replacing an existing IP address (which may have been assigned to a different user before), in which case the DNS servers may update their servers to indicate this change. The broadcasting may take the form of sending an allocation event message throughout the network containing the appropriate information.

A user may then find out the dynamically allocated or newly allocated address of another user by simply making a standard DNS query, allowing for utilities such as Internet Chat or Internet Phone to operate at full capability.

FIG. 4 is a flow diagram illustrating a method for updating one or more DNS servers in accordance with a another embodiment of the present invention. At step **150**, the broker maintains a list of all subscribing DHCP and DNS servers. A subscribing DHCP or DNS server is any DHCP or DNS server that wishes to be maintained by the broker. Therefore, generally every DHCP or DNS server maintained by an ISP will be a subscribing DHCP or DNS server with regards to a broker maintained by the ISP. It is also possible for DHCP or DNS servers outside the ISP to subscribe as well. When a new DHCP or DNS server wishes to subscribe or an existing DHCP or DNS server wishes to unsubscribe, it need simply send a message so indicating to the broker. At step **152**, the broker receives information regarding a dynamically allocated IP address that needs to be revoked. This information will most likely have been sent from a DHCP server which had been alerted as to a user disconnecting from the service, or directly from software tracking when users disconnect from the service.

At step **154**, the broker broadcasts the information regarding the revocation of the dynamically allocated IP address to each subscribing DHCP or DNS server. Each subscribing DHCP or DNS server may then update their databases to reflect this change. Subscribing DNS servers will simply clear the corresponding record from their databases, while subscribing DHCP servers may clear the record and return the dynamic IP address to a pool. This allows for the effective management of revoking dynamically allocated IP addresses. The broadcasting may take the form of sending a revocation event message through the network containing the appropriate information. In most systems currently being used, there is only a single DHCP server, or there are multiple DHCP servers but they do not share common information (i.e. not distributed). In these systems, there is no need to maintain a list of subscribing DHCP servers as there is no need for the broker to update the DHCP servers.

The broker itself may be executed in either a software or a hardware application. The broker may be designed to utilize the Common Object Request Broker Architecture (CORBA), which handles the communication of messages to and from objects in a distributed, multi-platform environment. CORBA provides a standard way of executing program modules in a distributed environment. The broker, therefore, may be incorporated into an Object Request Broker (ORB) within a CORBA compliant network.

To make a request of an ORB, a client may use a dynamic invocation interface (which is a standard interface which is independent of the target object's interface) or an Object Management Group Interface Definition Language (OMG IDL) stub (the specific stub depending on the interface of the target object). For some functions, the client may also directly interact with the ORB. The object is then invoked. When an invocation occurs, the ORB core arranges so a call is made to the appropriate method of the implementation. A parameter to that method specifies the object being invoked, which the method can use to locate the data for the object. When the method is complete, it returns, causing output parameters or exception results to be transmitted back to the client.

The broker may also be applied to any type of network address, rather than simply IP addresses. The use of the Internet as an example in this application is not intended to limit the scope of the invention to use on the Internet, as it may be used in a wide variety of networks. Likewise, the use of the terms DNS server and DHCP server is illustrative only and should be read to include any type of servers that may perform tasks that handle network addressing.

While embodiments and applications of this invention have been shown and described, it would be apparent to those skilled in the art that many more modifications than mentioned above are possible without departing from the inventive concepts herein. The invention, therefore, is not to be restricted except in the spirit of the appended claims.

What is claimed is:

1. A method for assigning network addresses in a network including one or more Domain Name Service (DNS) servers, said method including:

allocating a network address;

sending information regarding said allocated network address to a broker; and

broadcasting said information regarding said allocated network address to the one or more DNS servers using said broker.

2. The method of claim 1, wherein said allocating includes dynamically allocating a network address.

3. The method of claim 1, wherein said allocating includes allocating a static network address.

4. The method of claim 1, further including:

receiving said broadcast information regarding said allocated network address in each of said one or more DNS servers; and

updating each of said one or more DNS servers with said broadcast information regarding said allocated network address.

5. The method of claim 1, wherein the one or more DNS servers are only those DNS servers which have subscribed to said broker, and the method further includes maintaining a list of those DNS servers which have subscribed to said broker.

6. The method of claim 5, wherein said broadcasting includes sending information regarding said dynamically allocated network address to only those DNS servers which have subscribed to said broker.

7. A method for revoking network addresses in a network including one or more DNS servers, said method including:

revoking a dynamically allocated network address;

sending information regarding said revoked dynamically allocated network address to a broker; and broadcasting said information regarding said revoked dynamically allocated network address to the one or more DNS servers using said broker.

8. The method of claim 7, further including:

receiving said broadcast information regarding said revoked dynamically allocated network address in each of said one or more DNS servers; and

updating each of said one or more DNS servers with said broadcast information regarding said revoked dynamically allocated network address.

9. The method of claim 7, wherein the one or more DNS servers are only those DNS servers which have subscribed to said broker, and the method further includes maintaining a list of those DNS servers which have subscribed to said broker.

10. The method of claim 9, wherein said broadcasting includes sending information regarding said revoked dynamically allocated network address to only those DNS servers which have subscribed to said broker.

11. The method of claim 7, wherein the network further includes one or more Dynamic Host Configuration Protocol (DHCP) servers, and said broadcasting includes broadcasting said information regarding revoked dynamically allocated networked address to the one or more DHCP servers using said broker.

12. The method of claim 11, further including:

receiving said broadcast information regarding said allocated network address in each of said one or more DNS servers;

updating each of said one or more DNS servers with said broadcast information regarding said allocated network address;

receiving said broadcast information regarding said revoked dynamically allocated network address in each of said one or more DHCP and one or more DNS servers; and

updating each of said one or more DHCP servers and one or more DNS servers with said broadcast information regarding said revoked dynamically allocated network address.

13. The method of claim 11, wherein the one or more DNS servers are only those DNS servers which have subscribed to said broker, the one or more DHCP servers are only those DHCP servers which have subscribed to said broker, and the method further includes maintaining a list of those DNS servers and DHCP servers which have subscribed to said broker.

14. The method of claim 13, wherein said broadcasting information regarding said allocated network address includes sending information regarding said allocated network address to only those DNS servers which have subscribed to said broker, and said broadcasting information regarding said revoked dynamically allocated network address includes sending information regarding said revoked dynamically allocated network address to only those DNS servers and DHCP servers which have subscribed to said broker.

15. A method for managing network addresses in a network including one or more Domain Name Service (DNS) servers, said method including:

allocating a network address;

sending information regarding said allocated network address to a broker;

broadcasting said information regarding said allocated network address to the one or more DNS servers using said broker;

revoking a dynamically allocated network address;

sending information regarding said dynamically allocated network address to said broker; and

broadcasting said information regarding said dynamically allocated network address to the one or more DNS servers using said broker.

16. The method of claim 15, wherein said allocating includes dynamically allocating a network address.

17. The method of claim 15, wherein said allocating includes allocating a static network address.

18. The method of claim 15, further including:

receiving said broadcast information regarding said allocated network address in each of said one or more DNS servers;

updating each of said one or more DNS servers with said broadcast information regarding said allocated network address;

receiving said broadcast information regarding said revoked dynamically allocated network address in each of said one or more DNS servers; and

updating each of said one or more DNS servers with said broadcast information regarding said revoked dynamically allocated network address.

19. The method of claim 15, wherein the one or more DNS servers are only those DNS servers which have subscribed to said broker, and the method further includes maintaining a list of those DNS servers which have subscribed to said broker.

20. The method of claim 19, wherein said broadcasting information regarding said allocated network address includes sending information regarding said allocated network address to only those DNS servers which have subscribed to said broker, and said broadcasting information regarding said revoked dynamically allocated network address includes sending information regarding said revoked dynamically allocated network address to only those DNS servers which have subscribed to said broker.

21. The method of claim 15, wherein the network further includes one or more Dynamic Host Configuration Protocol (DHCP) servers, and said broadcasting includes broadcasting said information regarding revoked dynamically allocated networked address to the one or more DHCP servers using said broker.

22. The method of claim 21, further including:

receiving said broadcast information regarding said allocated network address in each of said one or more DNS servers;

updating each of said one or more DNS servers with said broadcast information regarding said allocated network address;

receiving said broadcast information regarding said revoked dynamically allocated network address in each of said one or more DHCP and one or more DNS servers; and

updating each of said one or more DHCP servers and one or more DNS servers with said broadcast information regarding said revoked dynamically allocated network address.

23. The method of claim 21, wherein the one or more DNS servers are only those DNS servers which have subscribed to said broker, the one or more DHCP servers are only those DHCP servers which have subscribed to said broker, and the method further includes maintaining a list of those DNS servers and DHCP servers which have subscribed to said broker.

24. The method of claim 23, wherein said broadcasting information regarding said allocated network address includes sending information regarding said allocated network address to only those DNS servers which have subscribed to said broker, and said broadcasting information regarding said revoked dynamically allocated network address includes sending information regarding said revoked dynamically allocated network address to only those DNS servers and DHCP servers which have subscribed to said broker.

25. A method for managing IP addresses in a network segment of the Internet, the network segment including one or more DNS servers, the method including:

receiving information regarding an assigned IP address;

broadcasting said information regarding an assigned IP address to the one or more DNS servers using a broker, for eventual update of the one or more DNS servers.

26. The method of claim 25, wherein said receiving information regarding an assigned IP address includes receiving information regarding an assigned IP address from a DHCP server.

27. The method of claim 25, wherein said information regarding an assigned IP address is information regarding a dynamically allocated IP address.

28. The method of claim 25, wherein said information regarding an assigned IP address is information regarding a static IP address.

29. The method of claim 25, further including maintaining a list of subscribing DNS servers.

30. The method of claim 29, wherein said broadcasting said information regarding an assigned IP address includes broadcasting said information regarding an assigned IP address only to subscribing DNS servers.

31. A method for managing IP addresses in a network segment of the Internet, the network segment including one or more DNS servers, the method including:

receiving information regarding a revoked dynamically allocated IP address; and

broadcasting said information regarding a revoked dynamically allocated IP address to the one or more DNS servers using a broker, for eventual update of the one or more DNS servers.

32. The method of claim 31, wherein said receiving information regarding a revoked dynamically allocated IP address includes receiving information regarding a revoked dynamically allocated IP address from a DHCP server.

33. The method of claim 25, further including maintaining a list of subscribing DNS servers.

34. The method of claim 33, wherein said broadcasting said information regarding a revoked dynamically allocated IP address includes broadcasting said information regarding a revoked dynamically allocated IP address only to subscribing DNS servers.

35. The method of claim 31, wherein the network segment of the Internet further includes one or more DHCP servers and said broadcasting said information regarding a revoked dynamically allocated IP address includes broadcasting said information regarding a revoked dynamically allocated IP address to the one or more DHCP servers and the one or more DNS servers.

36. The method of claim 35, further including maintaining a list of subscribing DNS servers and DHCP servers.

37. The method of claim 36, wherein said broadcasting said information regarding a revoked dynamically allocated IP address includes broadcasting said information regarding a revoked dynamically allocated IP address only to subscribing DNS servers and DHCP servers.

38. A method for managing IP addresses in a network segment of the Internet, the network segment including one or more DNS servers, the method including:

receiving information regarding an assigned IP address;

broadcasting said information regarding an assigned IP address to the one or more DNS servers using a broker, for eventual update of the one or more DNS servers;

receiving information regarding a revoked dynamically allocated IP address; and

broadcasting said information regarding a revoked dynamically allocated IP address to the one or more DNS servers using said broker, for eventual update of the one or more DNS servers.

39. The method of claim 38, wherein said receiving information regarding an assigned IP address includes receiving information regarding an assigned IP address from a DHCP server.

40. The method of claim 38, wherein said receiving information regarding a revoked dynamically allocated IP address includes receiving information regarding a revoked dynamically allocated IP address from a DHCP server.

41. The method of claim 38, wherein said information regarding an assigned IP address is information regarding a dynamically allocated IP address.

42. The method of claim 38, wherein said information regarding an assigned IP address is information regarding a static IP address.

43. The method of claim 38, further including maintaining a list of subscribing DNS servers.

44. The method of claim 43, wherein said broadcasting said information regarding an assigned IP address includes broadcasting said information regarding an assigned IP address only to subscribing DNS servers, and said broadcasting said information regarding a revoked dynamically allocated IP address includes broadcasting said information regarding a revoked dynamically allocated IP address only to subscribing DNS servers.

45. The method of claim 38, wherein the network segment of the Internet further includes one or more DHCP servers and said broadcasting said information regarding a revoked dynamically allocated IP address includes broadcasting said information regarding a revoked dynamically allocated IP address to the one or more DHCP servers and the one or more DNS servers.

46. The method of claim 45, further including maintaining a list of subscribing DNS servers and DHCP servers.

47. The method of claim 46, wherein said broadcasting said information regarding an assigned IP address includes broadcasting said information regarding an assigned IP address only to subscribing DNS servers, and said broadcasting said information regarding a revoked dynamically allocated IP address includes broadcasting said information regarding a revoked dynamically allocated IP address only to subscribing DNS servers and DHCP servers.

48. A method for dynamically allocating a network address to a subscriber in a communications network, the communications network having one or more DNS servers, the method comprising:

assigning a host address to the subscriber by selecting an address from a pool of available network addresses, said assigning step performed in response to the subscriber attempting to log-on to the communications network;

sending information regarding said subscriber as well as said host address to a broker;

utilizing said broker to broadcast said information regarding said subscriber as well as said host address to the one or more DNS servers; and updating the one or more DNS servers with said information regarding said subscriber as well as said host address.

49. The method of claim 48, wherein the one or more DNS servers are only those DNS servers which have subscribed to said broker, and the method further includes maintaining a list of those DNS servers which have subscribed to said broker.

50. A method for revoking a dynamically allocated network address assigned by a DHCP server in a communications network, the communications network having one or more DNS servers, the method comprising:

removing the dynamically allocated network address from the DHCP server which assigned the address;

returning the dynamically allocated network address to a pool of available addresses associated with said DHCP server which assigned the address;

sending information regarding the dynamically allocated network address to a broker;

utilizing said broker to broadcast said information regarding the dynamically allocated network address to the one or more DNS servers; and

updating the one or more DNS servers with said information regarding said dynamically allocated network address.

51. The method of claim 50, wherein the one or more DNS servers are only those DNS servers which have subscribed to said broker, and the method further includes maintaining a list of those DNS servers which have subscribed to said broker.

52. The method of claim 50, wherein the communications network further has a plurality of DHCP servers, and said utilizing further includes utilizing said broker to broadcast information regarding the dynamically allocated network address to the plurality of DHCP servers.

53. The method of claim 52, wherein the plurality of DHCP servers are only those DHCP servers which have subscribed to said broker, and the method further includes maintaining a list of those DNS servers which have subscribed to said broker.

54. A communications network including:

one or more DNS servers, which maintain a list of assigned host addresses within the communications network;

an address allocator, which allocates a network address to a host on the communications network;

a transmitter, which sends information regarding said allocated network address to a broker; and

said broker having a broadcaster, which broadcasts said information regarding said allocated network address to said one or more DNS servers.

55. The communications network of claim 54, wherein said address allocator is a DHCP server.

56. The communications network of claim 54, wherein said network address is a status network address.

57. The communications network of claim 54, wherein said broker further includes a list of subscribing DNS servers and broadcasts said information regarding said allocated network address only to the subscribing DNS servers.

58. The communications network of claim 54, wherein said one or more DNS servers receive said broadcast information regarding said allocated network address and update themselves with said information.

59. A communications network including:

one or more DNS servers;

an address revoker, which revokes a dynamically allocated network address from a host on the communications network;

a transmitter, which sends information regarding said revoked dynamically allocated network address to a broker; and

said broker having a broadcaster, which broadcasts said information regarding said revoked dynamically allocated network address to said one or more DNS servers.

60. The communications network of claim 59, wherein said broker further includes a list of subscribing DNS servers and broadcasts said information regarding said revoked dynamically allocated network address only to the subscribing DNS servers.

61. The communications network of claim 59, wherein said one or more DNS servers receives said broadcast information regarding said revoked dynamically allocated network address and update themselves with said information.

62. A communications network including:

one or more DNS servers, which maintain a list of assigned host addresses within the communications network;

an address allocator, which allocates a network address to a first host on the communications network;

a transmitter, which sends information regarding said allocated network address to a broker;

said broker having a broadcaster, which broadcasts said information regarding said allocated network address to said one or more DNS servers; and

an address revoker, which revokes a dynamically allocated network address from a second host on the communications network,

wherein said transmitter further sends information regarding said revoked dynamically allocated network address to said broker, and

wherein said broadcaster further broadcasts said information regarding said revoked dynamically allocated network address to said one or more DNS servers.

63. The communications network of claim 62, wherein said broker further includes a list of subscribing DNS servers, and wherein said broadcaster broadcasts said information regarding said allocated network address only to the subscribing DNS servers and broadcasts said information regarding said revoked dynamically allocated network address only to the subscribing DNS servers.

64. The communications network of claim 62, wherein said one or more DNS servers receive said broadcast information regarding said allocated network address and update themselves with said information, and said one or more DNS servers receive said broadcast information regarding said revoked dynamically allocated network address and update themselves with said information.

65. The communications network of claim 62, further including one or more DHCP servers, wherein said broadcaster further broadcasts said information regarding said revoked dynamically allocated network address to said one or more DHCP servers.

66. The communications network of claim 65, wherein said broker further includes a list of subscribing DNS servers and DHCP servers, and said broadcaster broadcasts said information regarding said allocated network address only to the subscribing DNS servers and broadcasts said

information regarding said revoked dynamically allocated network address only to the subscribing DNS servers and DHCP servers.

67. The communications network of claim 65, wherein said one or more DNS servers receive said broadcast information regarding said allocated network address and update themselves with said information, and wherein said one or more DHCP servers receive said broadcast information regarding said revoked dynamically allocated network address, update themselves with said information, and return said revoked dynamically allocated network address to a pool of available addresses.

68. A broker for managing host addresses assigned to subscribers in a communications network, including:

- a receiver, which receives information regarding a subscriber and an assigned host address, said assigned host address assigned to the subscriber by selecting an address from a pool of available network addresses in response to the subscriber attempting to log on to the communications network; and
- a broadcaster which broadcasts said information regarding said subscriber as well as said assigned host address to one or more DNS servers for eventual update.

69. The broker of claim 68, further including a database manager, which maintains a list of subscribing DNS servers, wherein said broadcaster broadcasts said information and said assigned host address only to the subscribing DNS servers.

70. A broker for managing host addresses assigned to subscribers in a communications network, said broker including:

- a receiver, which receives information regarding a revoked dynamically assigned host address, which was revoked in response to a subscriber attempting to log off the communications network; and
- a broadcaster which broadcasts said information regarding said subscriber as well as said revoked dynamically assigned host address to one or more DNS servers for eventual update and release of said dynamically assigned host address into one or more pools of available addresses.

71. The broker of claim 70, further including a database manager, which maintains a list of subscribing DNS servers, wherein said broadcaster broadcasts said information and said assigned host address only to the subscribing DNS servers.

72. The broker of claim 70, wherein said broadcaster further broadcasts said information regarding said subscriber as well as said revoked dynamically assigned host address to one or more DHCP servers for eventual update and release of said dynamically assigned host address into one or more pools of available addresses.

73. The broker of claim 72, further including a database manager, which maintains a list of subscribing DHS and DHCP servers, wherein said broadcaster broadcasts said information and said assigned host address only to the subscribing DNS servers and DHCP servers.

74. A broker for managing host addresses assigned to subscribers in a communications network, said broker including:

- a receiver, which receives information regarding a subscriber and an assigned host address, said assigned host address assigned to said subscriber by selecting an address from a pool of available network addresses in response to said subscriber attempting to log on to a communications network, and which receives information regarding a revoked dynamically assigned host address, which was revoked in response to a subscriber attempting to log off the communications network; and
- a broadcaster which broadcasts said information regarding said subscriber as well as said assigned host address to one or more DNS servers for eventual update, and which broadcasts said information regarding said subscriber as well as said revoked dynamically assigned host address to said one or more DNS servers for eventual update.

75. The broker of claim 74, further including a database manager, which maintains a list of subscribing DNS servers, wherein said broadcaster broadcasts said information and said assigned host address only to the subscribing DNS servers, and broadcasts said information regarding said revoked dynamically allocated host address only to the subscribing DNS servers.

76. A broker for managing host addresses assigned to subscribers in a communications network, said broker including:

- a receiver, which receives information regarding a subscriber and an assigned host address, said assigned host address assigned to said subscriber by selecting an address from a pool of available network addresses in response to said subscriber attempting to log on to a communications network, and which receives information regarding a revoked dynamically assigned host address, which was revoked in response to a subscriber attempting to log off the communications network; and
- a broadcaster which broadcasts said information regarding said subscriber as well as said assigned host address to one or more DNS servers for eventual update, and which broadcasts said information regarding said subscriber as well as said revoked dynamically assigned host address to one or more DNS or DHCP servers for eventual update and release of said dynamically assigned host address into one or more pools of available addresses.

77. The broker of claim 76, further including a database manager, which maintains a list of subscribing DNS servers and DHCP servers, wherein said broadcaster broadcasts said information and said assigned host address only to the subscribing DNS servers and broadcasts said information regarding said revoked dynamically allocated host address only to the subscribing DNS servers and DHCP servers.

78. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform a method for managing network addresses in a network including one or more Domain Name Service (DNS) servers, said method including:

- receiving information regarding an allocated network address; and
- broadcasting said information regarding said allocated network address to the one or more DNS servers using a broker, for the eventual update of the one or more DNS servers.

79. The program storage device of claim 78, wherein said allocated network address is a dynamically allocated network address.

80. The program storage device of claim 78, wherein said allocated network address is a static network address.

81. The program storage device of claim 78, wherein the one or more DNS servers are only those DNS servers which have subscribed to said broker, and the method further includes maintaining a list of those DNS servers which have subscribed to said broker.

82. The program storage device of claim 81, wherein said broadcasting further includes sending information regarding said allocated network address to only those DNS servers which have subscribed to said broker.

83. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform a method for managing network addresses in a network including one or more DNS servers, said method including:

receiving information regarding a revoked dynamically allocated network address; and

broadcasting said information regarding said revoked dynamically allocated network address to the one or more DNS servers using a broker, for eventual update of the one or more DNS servers.

84. The program storage device of claim 83, wherein the one or more DNS servers are only those DNS servers which have subscribed to said broker and the method further includes the step of maintaining a list of those DNS servers which have subscribed to said broker.

85. The program storage device of claim 84, wherein said broadcaster sends information regarding said revoked dynamically allocated network address to only those DNS servers which have subscribed to said broker.

86. The program storage device of claim 83, wherein the network further includes one or more DHCP servers, and said broadcasting further includes broadcasting said information regarding said revoked dynamically allocated network address to the one or more DHCP servers using said broker, for eventual update of the one or more DHCP servers.

87. The program storage device of claim 86, wherein the one or more DHCP servers are only those DHCP servers which have subscribed to said broker, and the method further includes of maintaining a list of those DNS servers and DHCP servers which have subscribed to said broker.

88. The program storage device of claim 87, wherein said broadcasting further includes broadcasting said information regarding said revoked dynamically allocated network address to only those DNS servers and DHCP server which have subscribed to said broker.

89. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform a method for managing network addresses in a network including one or more DNS servers, the method including:

receiving information regarding an assigned network address;

broadcasting said information regarding an assigned network address to the one or more DNS servers using a broker, for eventual update of the one or more DNS servers;

receiving information regarding a revoked dynamically allocated network address; and

broadcasting said information regarding a revoked dynamically allocated network address to the one or more DNS servers using said broker, for eventual update of the one or more DNS servers.

90. The program storage device of claim 89, wherein said receiving information regarding an assigned network address includes receiving information regarding an assigned network address from a DHCP server.

91. The program storage device of claim 89 wherein said receiving information regarding a revoked dynamically allocated network address includes receiving information regarding a revoked dynamically allocated network address from a DHCP server.

92. The program storage device of claim 89, wherein said information regarding an assigned network address is information regarding a dynamically allocated network address.

93. The program storage device of claim 89, wherein said information regarding an assigned network address is information regarding a static network address.

94. The program storage device of claim 89, wherein the method further includes maintaining a list of subscribing DNS servers.

95. The program storage device of claim 94, wherein said broadcasting said information regarding an assigned network address includes broadcasting said information regarding an assigned network address only to subscribing DNS servers, and said broadcasting said information regarding a revoked dynamically allocated network address includes broadcasting said information regarding a revoked dynamically allocated network address only to subscribing DNS servers.

96. The program storage device of claim 89, wherein the network further includes one or more DHCP servers, wherein said broadcasting said information regarding an assigned network address further includes broadcasting said information regarding said assigned network address to the one or more DHCP servers using said broker, for eventual update of the one or more DHCP servers, and wherein said broadcasting said information regarding a revoked dynamically allocated network address further includes broadcasting said information regarding said revoked dynamically assigned network address to the one or more DHCP servers using said broker, for eventual update of the one or more DHCP servers.

97. The program storage device of claim 96, wherein the one or more DHCP servers are only those DHCP servers which have subscribed to said broker, and the method further includes maintaining a list of those DNS servers and DHCP servers which have subscribed to said broker.

98. The program storage device of claim 97, wherein said broadcasting said information regarding said assigned network address further includes broadcasting only to those DNS servers and DHCP servers which have subscribed to said broker, and said broadcasting said information regarding a revoked dynamically allocated network address further includes broadcasting said information regarding said revoked dynamically assigned network address to only those DNS servers and DHCP servers which have subscribed to said broker.

* * * * *

US006119171A

United States Patent [19]

Alkhatib

[11] **Patent Number: 6,119,171**

[45] **Date of Patent: Sep. 12, 2000**

[54] **DOMAIN NAME ROUTING**

[75] Inventor: **Hasan S. Alkhatib**, Saratoga, Calif.

[73] Assignee: **IP Dynamics, Inc.**, Santa Clara, Calif.

[21] Appl. No.: **09/015,840**

[22] Filed: **Jan. 29, 1998**

[51] **Int. Cl.**⁷ **G06F 13/14**; H04J 3/26; H04L 12/66

[52] **U.S. Cl.** **709/245**; 709/202; 709/238; 709/249; 709/250; 370/390; 370/392; 370/397; 370/409

[58] **Field of Search** 709/201, 220–224, 709/235, 238, 200–203, 205, 244–250; 711/200, 202, 205–207; 370/389–390, 392–393, 396–397, 400–401, 405–409

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,361,256	11/1994	Doeringer et al.	370/390
5,623,605	4/1997	Keshav et al.	709/236
5,717,686	2/1998	Minot et al.	370/396
5,777,989	7/1998	Mc Garvey	709/249
5,790,548	8/1998	Sistanizadeh et al.	370/401
5,805,818	9/1998	Perlman et al.	709/224
5,856,974	1/1999	Gervais et al.	370/392
5,867,667	2/1999	Butman et al.	709/249
5,884,246	3/1999	Boucher et al.	709/206
5,889,953	3/1999	Thebaut et al.	709/221
5,937,162	4/1999	Funk et al.	709/206
5,937,163	4/1999	Lee et al.	707/102

OTHER PUBLICATIONS

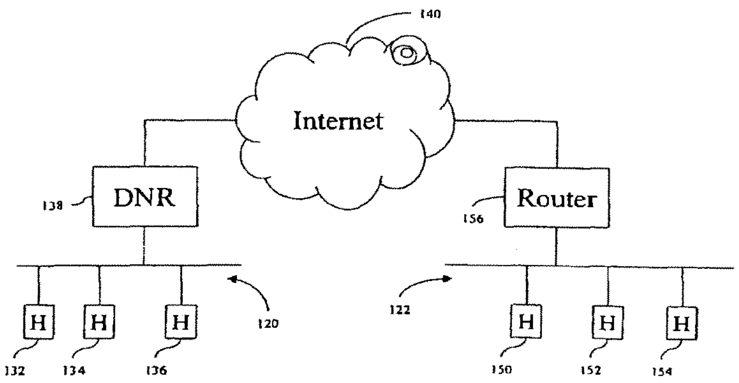
Inside AppleTalk®, Second Edition, by G. Sidhu, R. Andrews, A. Oppenheimer, 1990.

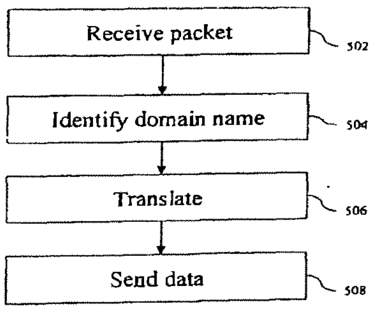
Primary Examiner—Zarni Maung
Assistant Examiner—Bharat Barot
Attorney, Agent, or Firm—Fliesler, Dubb, Meyer & Lovejoy LLP

[57] **ABSTRACT**

The present invention provides for a Domain Name Router (DNR) that uses domain names to route data sent to a destination on a network (e.g., a stub network). Each corporate entity or stub network can be assigned one or a small number of global addresses. Each of the hosts on the stub network can be assigned a global address. When a source entity sends data to a destination entity with a local address, the data is sent to the DNR using a global address. The source entity embeds the destination's domain name and its own domain name inside the data. The DNR extracts the destination's domain name from the data, translates that domain name to a local address and sends the data to the destination.

48 Claims, 11 Drawing Sheets





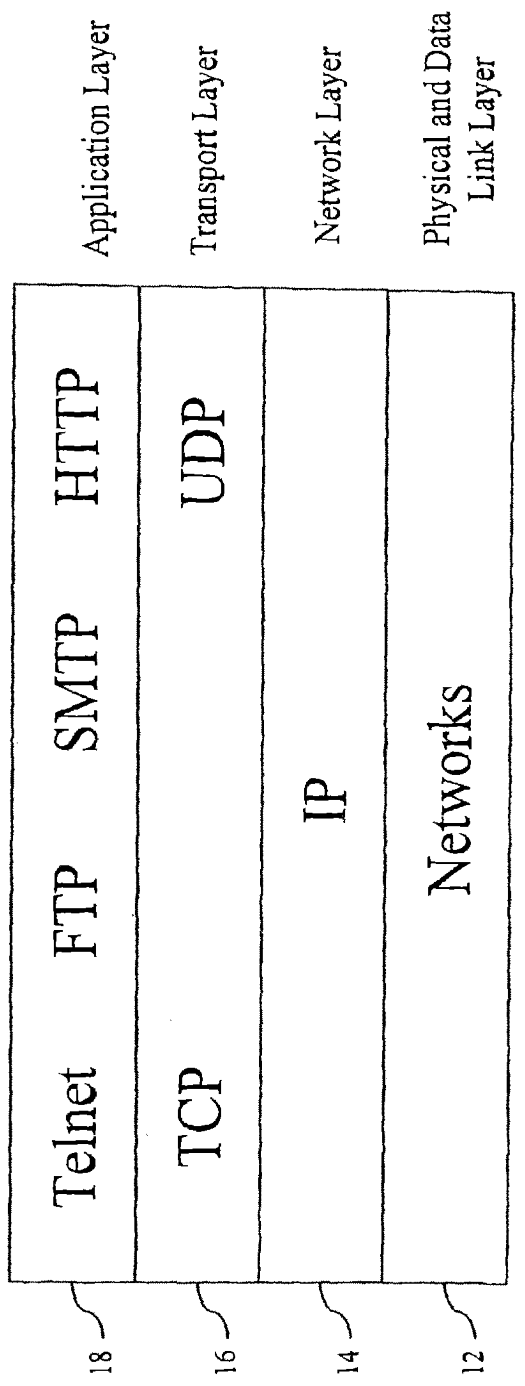

Application Layer
Transport Layer
Network Layer
Physical and Data Link Layer
Telnet
FTP
SMTP
HTTP
TCP
UDP
IP
Networks
18
16
14
12

Fig 1

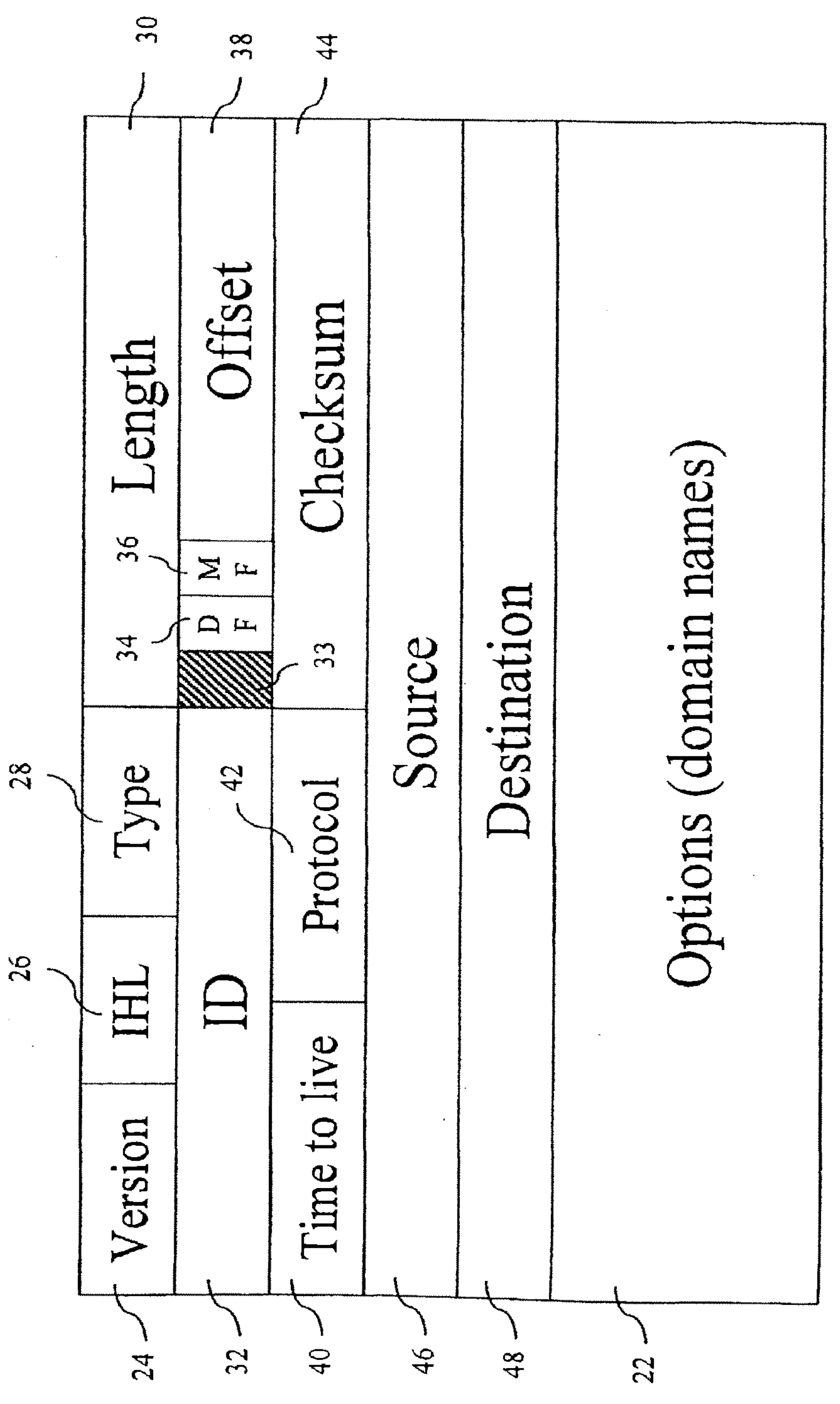

Version
IHL
Type
Length
ID
D F
M F
Offset
Time to live
Protocol
Checksum
Source
Destination
Options (domain names)
24
26
28
30
32
33
34
36
38
40
42
44
46
48
22

Fig 2

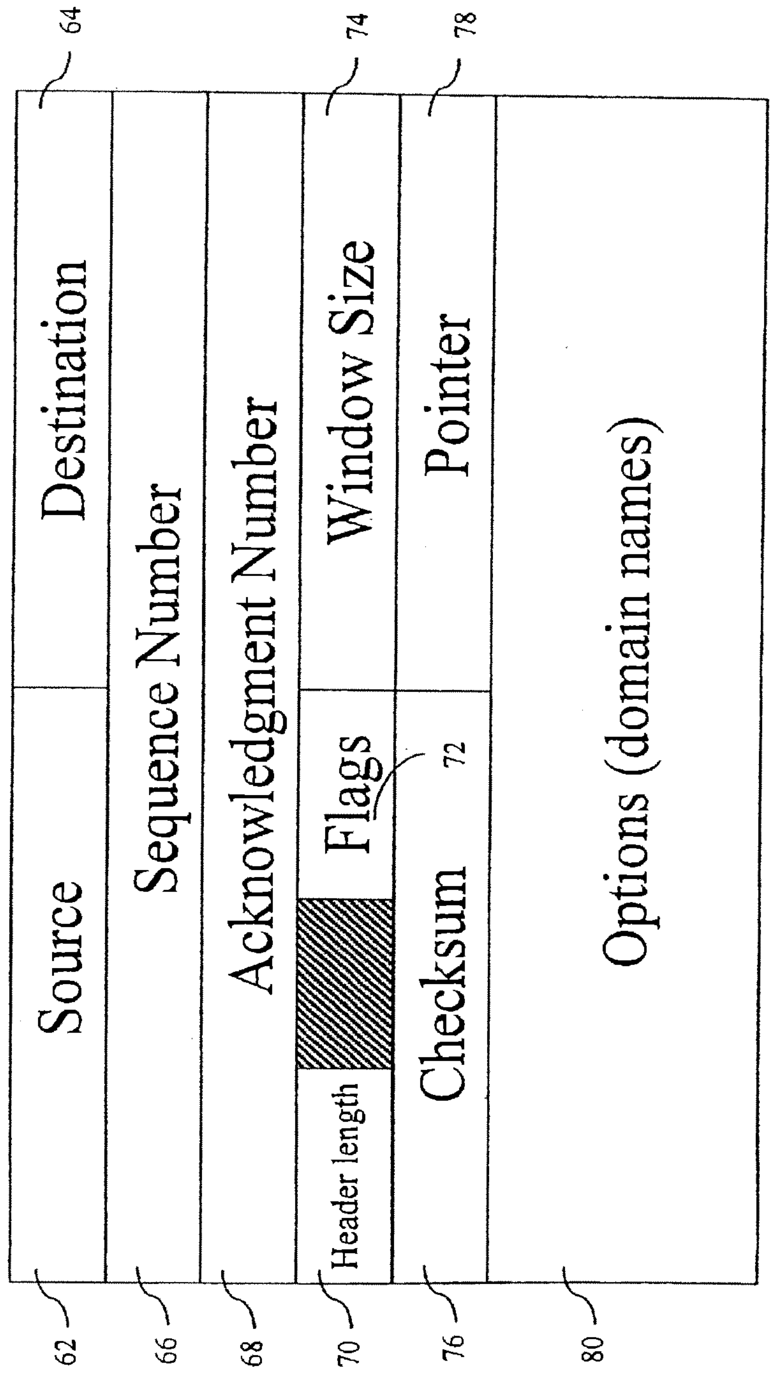

Source
Destination
Sequence Number
Acknowledgment Number
Header length
Flags
Window Size
Checksum
Pointer
Options (domain names)
62
64
66
68
70
72
74
76
78
80

Fig 3

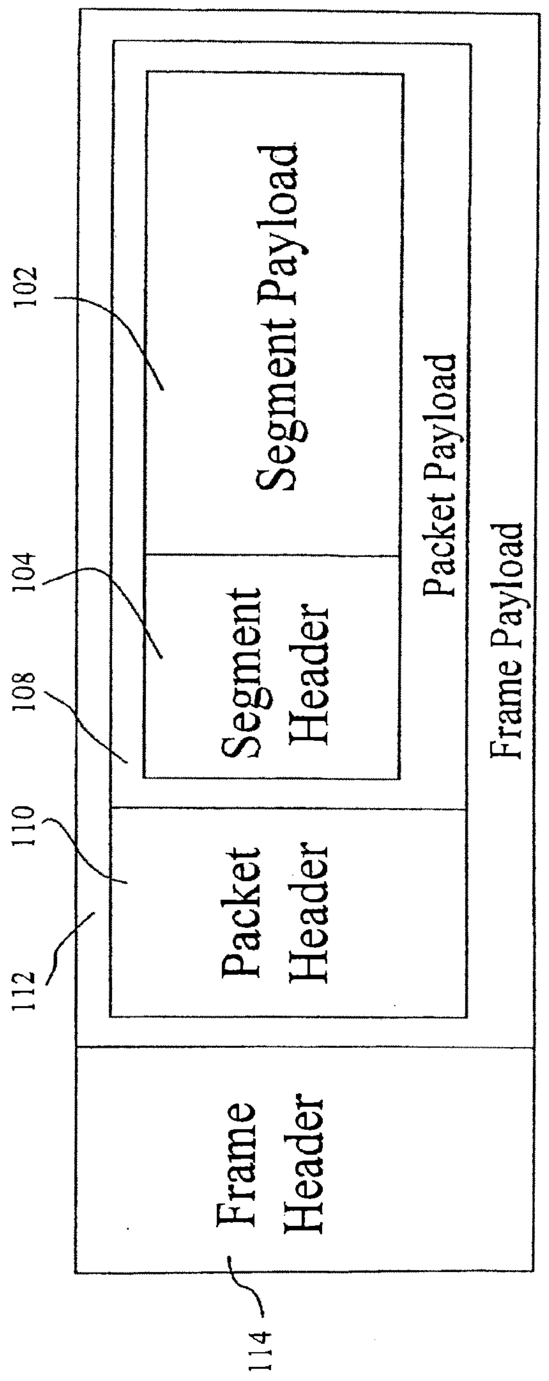

102
104
108
110
112
114
Segment Payload
Segment Header
Packet Payload
Packet Header
Frame Payload
Frame Header

Fig 4

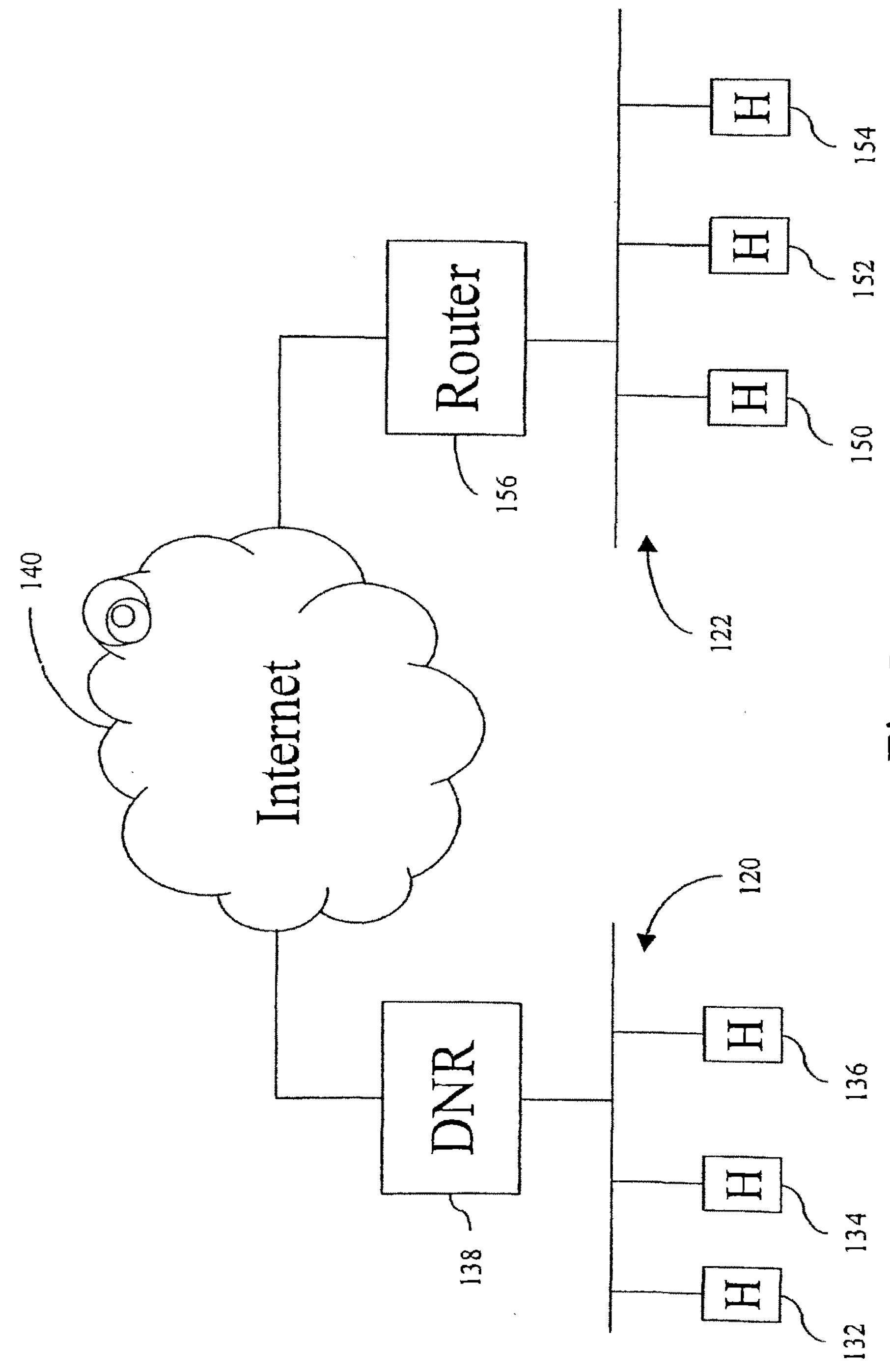

Internet
140
Router
156
H
H
H
150
152
154
122
DNR
138
H
H
H
132
134
136
120

Fig 5

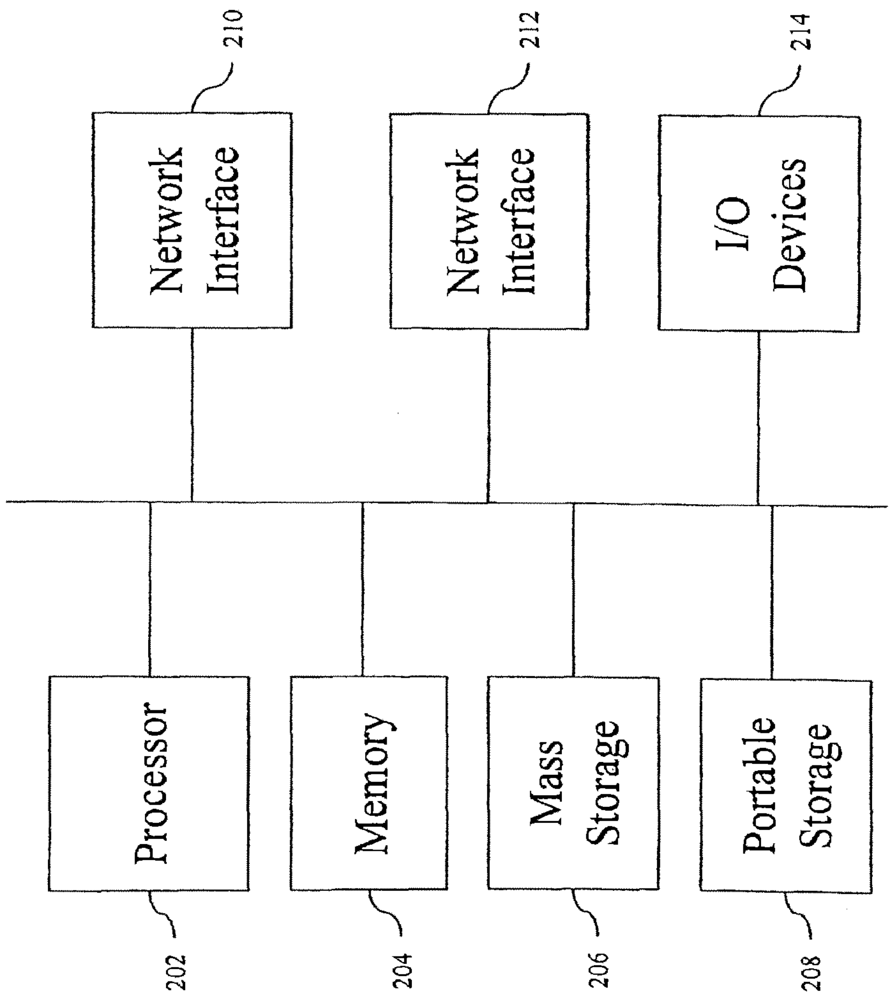

Network Interface
210
Network Interface
212
I/O Devices
214
Processor
202
Memory
204
Mass Storage
206
Portable Storage
208

Fig 6

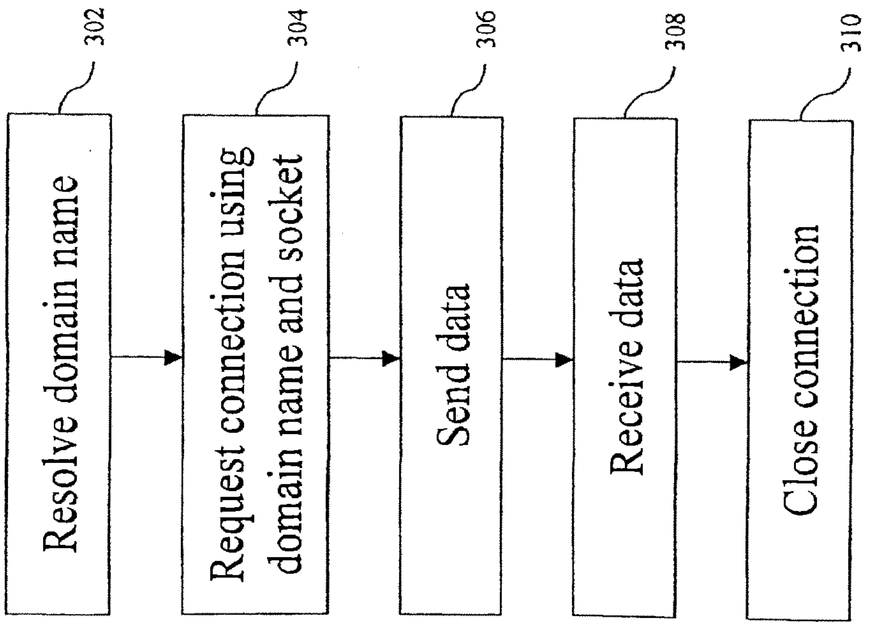

Resolve domain name
302
Request connection using domain name and socket
304
Send data
306
Receive data
308
Close connection
310

Fig 7

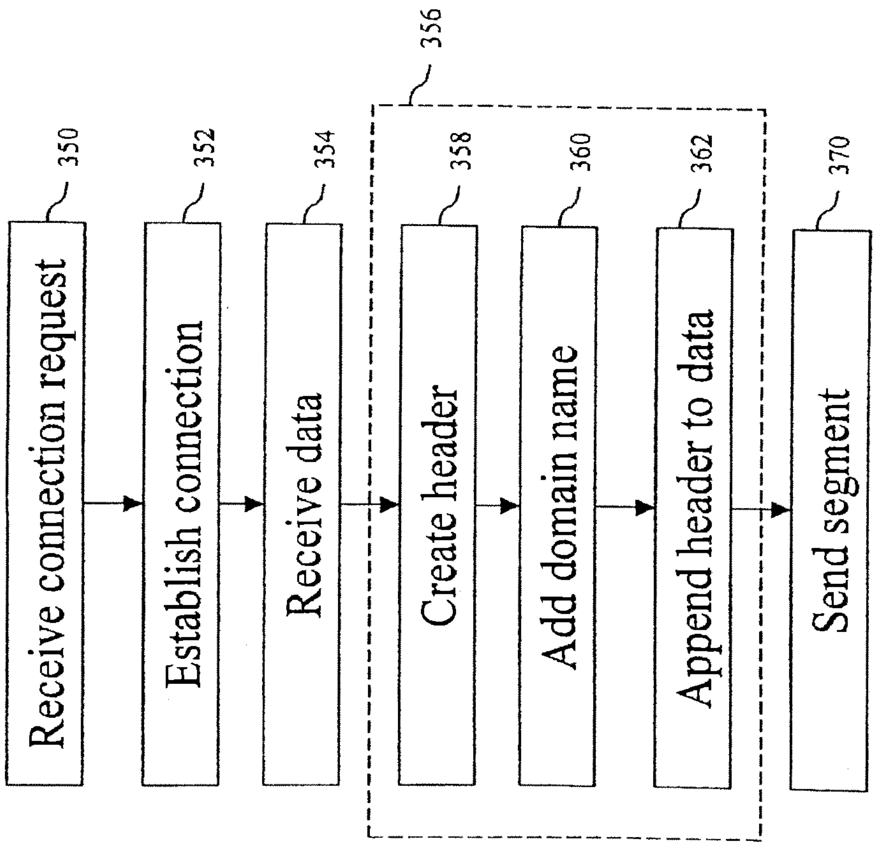

Receive connection request
350
Establish connection
352
Receive data
354
356
Create header
358
Add domain name
360
Append header to data
362
Send segment
370

Fig 8

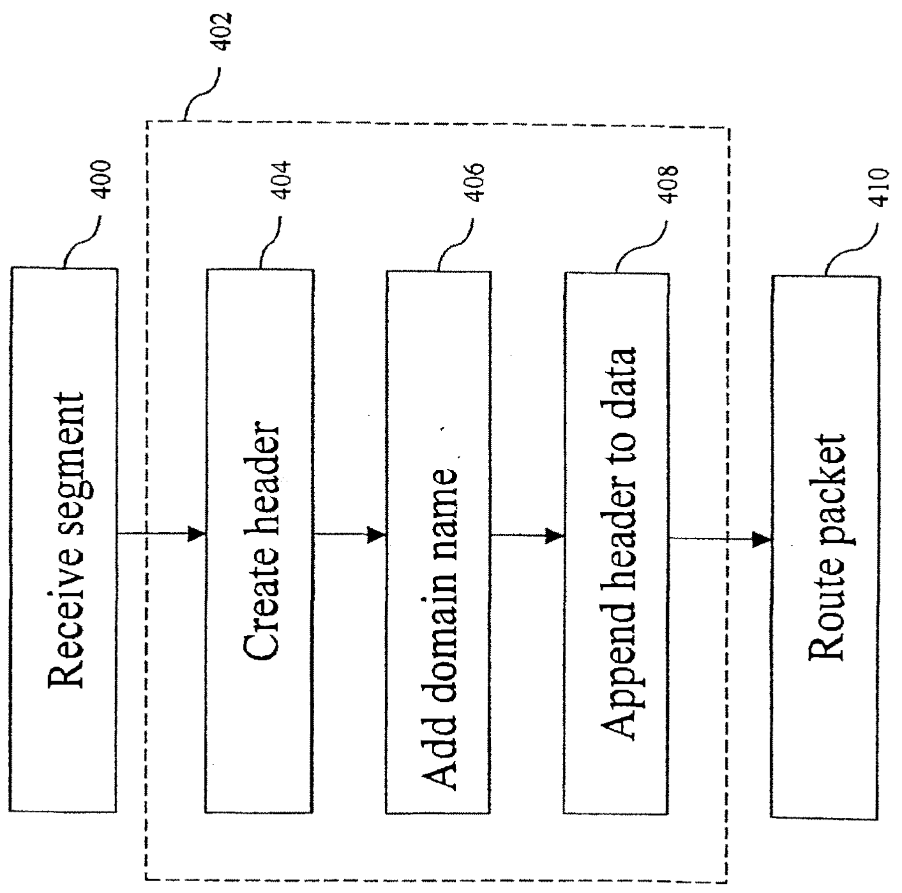

400
Receive segment
402
404
Create header
406
Add domain name
408
Append header to data
410
Route packet

Fig 9

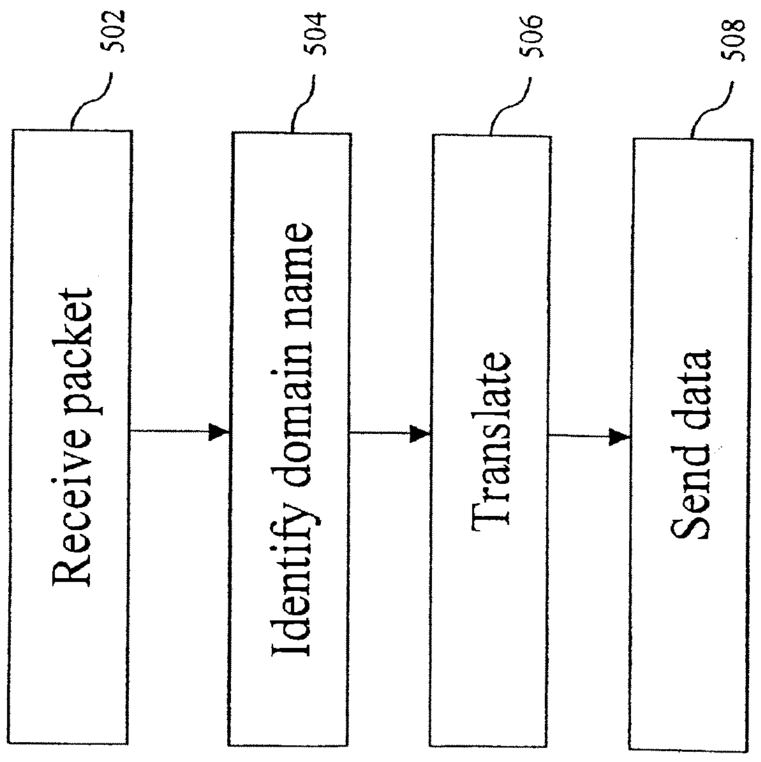

502
504
506
508
Receive packet
Identify domain name
Translate
Send data

Fig 10

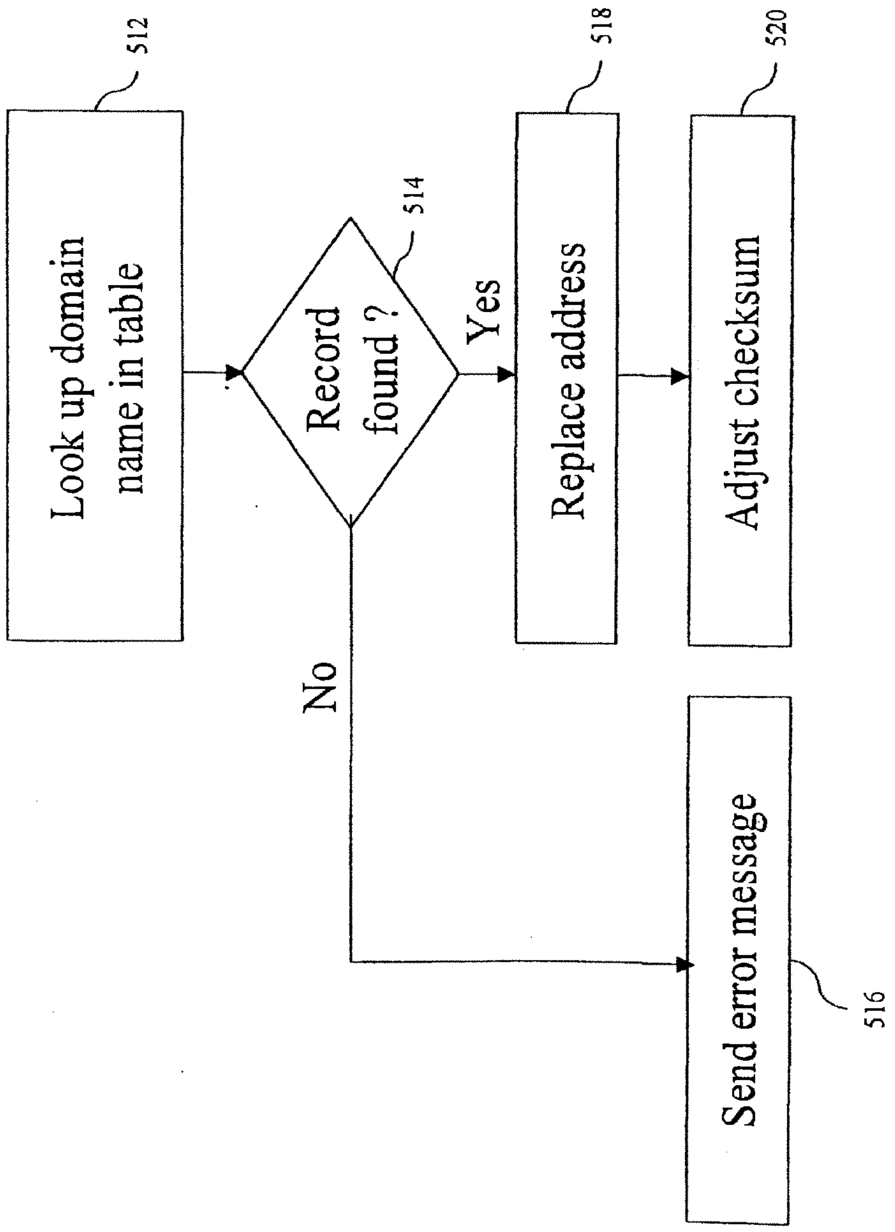

512
Look up domain name in table
514
Record found ?
Yes
No
518
Replace address
520
Adjust checksum
Send error message
516

Fig 11

DOMAIN NAME ROUTING

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention is directed to a system for using Internet domain names to route data sent to a destination on a network.

2. Description of the Related Art

Most machines on the Internet use TCP/IP (Transmission Control Protocol/Internet Protocol) to send data to other machines on the Internet. To transmit data from a source to a destination, the Internet Protocol (IP) uses an IP address. An IP address is four bytes long, which consists of a network number and a host number.

There are at least three different classes of networks currently in use: Class A, Class B and Class C. Each class has a different format for the combination of the network number and the host number in the IP addresses. A Class A address includes one byte to specify the network and three bytes to specify the host. The first bit of a Class A address is a 0 to indicate Class A. A Class B address uses two bytes for the network address and two bytes for the host address. The first two bits of the Class B address are 10 to indicate Class B. The Class C address includes three bytes to specify the network and one byte for the host address. The first three bits of the Class C network address are 110 to indicate Class C. The formats described above allow for 126 Class A networks with 16 million hosts each; 16,382 Class B networks with up to 64K hosts each; and 4 million Class C networks with up to 256 hosts each.

When written out, IP addresses are specified as four numbers separated by dots (e.g. 198.68.70.1). Users and software applications rarely refer to hosts, mailboxes or other resources by their numerical IP address. Instead of using numbers, they use ASCII strings called domain names. A domain name is usually in the form of prefix.name_of_organization.top_level_domain. There are two types of top level domains: generic and countries. The generic domains are com (commercial), edu (educational institutions), gov (the U.S. Federal Government), int (international organizations), mil (the U.S. Armed Forces), net (network providers), and org (non-profit organizations). The country domains include one entry for each country. An example of a domain name is saturn.ttc.com. The term "saturn" is the prefix and may refer to a particular host in the network. The phrase "ttc" is the name of the organization and can be used to identify one or more networks to the outside world. The phrase "com" signifies that this address is in the commercial domain. The Internet uses a Domain Name System to convert the domain name to an IP address.

The Internet Protocol has been in use for over two decades. It has worked extremely well, as demonstrated by the exponential growth of the Internet. Unfortunately, the Internet is rapidly becoming a victim of its own popularity: it is running out of addresses. Over 4 billion addresses exist, but the practice of organizing the address space into classes wastes millions of addresses. In particular, the problem is the Class B network. For most organizations, a Class A network, with 16 million addresses is too big, and a Class C network with 256 addresses is too small. A Class B network appears to be the right solution for most companies. In reality, however, a Class B address is far too large for most organizations. Many Class B networks have fewer than 50 hosts. A Class C network would have done the job, but many organizations that ask for Class B networks thought that one day they would outgrow the 8 bit host field.

One proposed solution to the depleting address problem is Classless Inter Domain Routing (CIDR). The basic idea behind CIDR is to allocate the remaining Class C networks in varied sized blocks. If a site needs 2,000 addresses, it is given a block of contiguous Class C networks, and not a full Class B network address. In addition to using blocks of contiguous Class C networks as units, the allocation rules for Class C addresses are also changed by partitioning the world into four zones. Each zone includes a predefined number of Class C networks. Although CIDR may buy a few more years time, IP addresses will still run out in the foreseeable future.

Another proposed solution is Network Address Translation (NAT). This concept includes predefining a number of Class C network addresses to be or local addresses (also called private addresses). The remainder of the addresses are considered global addresses. Global addresses are unique addresses. That is, no two entities on the Internet will have the same global address. Local addresses are not unique and can be used by more than one organization or network. However, a local address cannot be used on the Internet. Local addresses can only be used within a private network. NAT assumes that less all of the machines on a private network will not need to access the Internet at all times. Therefore, there is no need for each machine to have a global address. A company can function with a small number of global addresses assigned to one or more gateway computers. The remainder of the machines on the private network will be assigned local addresses. When a particular machine on the private network using a local address attempts to initiate a communication to a machine outside of the private network (e.g. via the Internet), the gateway machine will intercept the communication, change the source machine's local address to a global address and set up a table for translation between global addresses and local addresses. The table can contain the destination address, port numbers, sequencing information, byte counts and internal flags for each connection associated with a host address. Inbound packets are compared against entries in the table and permitted through the gateway only if an appropriate connection exists to validate their passage. One problem with the NAT approach is that it only works for communication initiated by a host within the network to a host on the Internet which has a global IP address. The NAT approach specifically will not work if the communication is initiated by a host outside of the private network and is directed to a host with a local address on the private network.

Another solution that has been proposed is a new version of the Internet Protocol called IPv6 (Internet Protocol version 6, also known as IPng). IPv6 is not compatible with the existing Internet Protocol (IPv4). For example, IPv6 has a longer address than IPv4. Additionally, the IPv6 header is different than the IPv4 header. Because IPv6 is not compatible with IPv4, almost all routing equipment on the Internet must be replaced with updated equipment that is compatible with IPv6. Such widespread replacement of legacy equipment is enormously expensive.

As can be seen, the current proposals to solve the diminishing IP addresses problem are inadequate and/or unduly expensive. Therefore, a system is needed that can effectively alleviate the diminishing IP addresses problem without unreasonable costs.

SUMMARY OF THE INVENTION

The present invention, roughly described, provides for a system for using domain names to route data sent to a destination on a network. One example includes routing data to a destination on a stub network. A stub network is a network owned by an organization that it is connected to the Internet through one or more gateways. Nodes in the stub network may be made visible to other nodes on the Internet or to other nodes in other stub networks interconnected through the Internet. Rather than use an entire set of global addresses for a Class A, B or C network, each corporate entity or stub network can be assigned one or a small number of global addresses. Each of the hosts can be assigned a local address. The same local addresses can be used by many different organizations. When a source entity sends data to a destination entity in a stub network with a local address, the data is sent to a global address for the destination's network. The global address is assigned to a Domain Name Router in communication with the destination's network. The Domain Name Router serves as a gateway between the Internet and the stub network. The Domain Name Router routes IP traffic between nodes on the Internet (identified by their globally unique IP addresses) and nodes in its stub network. The source entity embeds the destination's domain name and its own domain name somewhere inside the data. The Domain Name Router receives the data, extracts the destination's domain name from the data, translates that domain name to a local address in its stub network and sends the data to the destination. Note that the source entity could have either a local address or a global address and still be able to utilize the present invention.

One method for practicing the present invention includes packaging at least a subset of data to be communicated to an entity on a network into a data unit. That data unit is sent to a Domain Name Router or other similar entity. Information representing the domain name of the destination is extracted from the data unit and used to determine a local address for the destination. Once a local address is determined, the data unit is sent to that local address.

The data unit can be formed by receiving a first set of data and a domain name. A field (or other subset) is created, which includes a first set of information representing the domain name. The field is appended to the first set of data to create the data unit. The data unit is sent to the Domain Name Router. The data unit could be an IP packet, a TCP segment, or any other data unit suitable for use with the present invention as long as the domain name can be reliably extracted from the data. In one embodiment, the information used to represent the domain name could include an encrypted version of the domain name, an encoded version of the domain name, a compressed version of the domain name, etc.

In one embodiment, the data unit sent to the Domain Name Router includes a global IP address for the Domain Name Router. After translating the domain name to a local address, the Domain Name Router will replace the global address for the Domain Name Router with the local address of the destination. The step of replacing the global address with the local address can include adjusting any appropriate checksums or any other necessary fields in the data unit.

The Domain Name Router can be implemented using software stored on a processor readable storage medium and run on a computer or a router. Alternatively, the Domain Name Router can be specific hardware designed to carry out the methods described herein.

These and other objects and advantages of the invention will appear more clearly from the following detailed description in which the preferred embodiment of the invention has been set forth in conjunction with the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a symbolic diagram showing the layers of the TCP/IP Reference Model.

FIG. 2 shows the Internet Protocol (IP) header.

FIG. 3 shows the Transmission Control Protocol (TCP) header.

FIG. 4 shows the nesting of segments, packets and frames.

FIG. 5 is a block diagram of two stub networks connected to the Internet.

FIG. 6 is a simplified block diagram of one exemplar hardware platform for implementing a Domain Name Router.

FIG. 7 is a flow chart describing the steps used by an application process to send data according to the present invention.

FIG. 8 is a flow chart describing the steps used by a transport layer process to send data according to the present invention.

FIG. 9 is a flow chart describing the steps used by a network layer process to send data according to the present invention.

FIG. 10 is a flow chart describing the steps performed by a Domain Name Router.

FIG. 11 is a flow chart describing the translation step of FIG. 10.

DETAILED DESCRIPTION

FIG. 1 shows the TCP/IP reference model for designing and building a network. The model includes four layers: Physical and Data Link Layer **12**, Network Layer **14**, Transport Layer **16**, and Application Layer **18**. The physical layer portion of Physical and Data Link Layer **12** is concerned with transmitting raw bits over a communication channel. The design issues include ensuring that when one side sends a 1 bit it is received by the other side as a 1 bit, not as a 0 bit. Typical questions addressed are how many volts should be used to represent a 1 bit, how many volts to represent a 0 bit, how many microseconds a bit lasts, whether transmissions may proceed simultaneously in both directions, how the initial connection is established, how it is torn down when both sides are finished, and how many pins the network connector has. The data link portion of Physical and Data Link Layer **12** takes the raw transmission facility and transforms it into a line that appears to be relatively free of transmission errors. It accomplishes this task by having the sender break the input data up into frames, transmit the frames and process the acknowledgment frames sent back by the receiver.

Network Layer **14** permits a host to inject packets into a network and have them travel independently to the destination. The protocol used for Network Layer **14** on the Internet is called the Internet Protocol (IP).

Transport Layer **16** is designed to allow peer entities on the source and destination to carry on a "conversation." On the Internet, two end-to-end protocols are used. The first one, the Transmission Control Protocol (TCP), is a reliable connection-oriented protocol that allows a byte stream originating on one machine to be delivered without error to another machine on the Internet. It fragments the incoming byte stream into discrete packets and passes each one to Network Layer **14**. At the destination, the receiving TCP process reassembles the received packets into the output stream. TCP also handles flow control to make sure a fast sender cannot swamp a slow receiver with more packets than it can handle. The second protocol used in Transport Layer 16 on the Internet, User Datagram Protocol (UDP), is an unreliable connectionless protocol for applications that do not want TCP sequencing or flow control. UDP is used for one-shot, client server type requests-reply queries for applications in which prompt delivery is more important than accurate delivery. Transport Layer 16 is shown as being above Network Layer 14 to indicate that Network Layer 14 provides a service to Transport Layer 16. Similarly, Transport Layer 16 is shown below Application Layer 18 to indicate that Transport Layer 16 provides a service to Application Layer 18.

Application Layer 18 contains the high level protocols, for example, Telnet, File Transfer Protocol (FTP), Electronic Mail—Simple Mail Transfer Protocol (SMTP), and HyperText Transfer Protocol (HTTP).

The following discussion describes the network and transport layers in more detail. The main function of Network Layer 14 is routing packets from a source entity to a destination entity. In most subnets, packets will require multiple hops to make the journey. The Network Layer software uses one or more routing methods for deciding which output line an incoming packet should be transmitted on. There are many routing methods that are well known in the art that can be used in a network layer. For purposes of this patent, no specific routing method is required. Any suitable routing method known in the art will suffice. Some examples of known routing methods include shortest path routing, flooding, flow based routing, distance vector routing, link state routing, hierarchical routing, routing for mobile hosts, broadcast routing and multicast routing. Within a network on the Internet, a suitable routing method may also be based on the Distance Vector Protocol or its successor the Open Shortest Path First (OSPF) protocol. Between networks on the Internet, the Border Gateway Protocol (BGP) can be used.

Communication in the Internet works as follows. Transport Layer 16 breaks up a stream of data from Application Layer 18 into a number of segments. Network Layer 14, using the Internet Protocol, transports the segments in one or more IP packets from source to destination, without regard to whether these machines or entities are on the same network. Each segment can be fragmented into small units as it is transported. When all of the fragments finally get to the destination machine, they are reassembled by Network Layer 14 into the original segment. This segment is then handed to the Transport Layer 16, which inserts it into the receiving process' (Application Layer 18) input stream.

An IP packet consists of a header and a data portion. The format of an IP header is shown in FIG. 2. FIG. 2 shows six rows making up the header. Each row is 32 bits wide. The first five rows of the header comprise a 20 byte fixed portion of the header. The last row of the header provides a variable sized Options section 22. Version field 24 keeps track of which version of the protocol the packet belongs to. The current version used on the Internet is version 4. IHL field 26 describes the length of the header in 32 bit words. Type field 28 indicates the type of service requested. Various combinations of reliability and speed are possible. Length field 30 includes the size of the packet, including both the header and the data. Identification field 32 is needed to allow the destination host to determine which segment the received fragment belongs to. All fragments of a segment contain the same identification value. Next comes three flags, which include an unused bit 33 and then two 1 bit fields 34 and 36. In one embodiment of the present invention, the unused bit 33 is used to indicate that the source of the packet uses a domain name for unique identification on the Internet instead of using a globally unique IP address. DF field 34 stands for don't fragment. It is an order to the routers not to fragment the segment because the destination is incapable of putting the pieces back together again. MF field 36 stands for more fragments. All fragments except for the last one have this bit set. Fragment offset field 38 indicates where in the current segment this fragment belongs. Time to Live field 40 is used to limit packet lifetime. It is supposed to count time in seconds, allowing a maximum life time of 255 seconds. In practice, it may count hops. The time is decremented on each hop by a router. When the time to live hits 0, the packet is discarded and a warning is sent back to the source using an Internet Control Messaging Protocol (ICMP) packet. This feature prevents packets from wandering around forever. Protocol Field 42 indicates which transport layer type is to receive the segment. TCP is one possibility, UDP is another. The present invention is not limited to any particular protocol. Checksum field 44 verifies the header. One method for implementing a checksum is to add up all 16 bit half words as they arrive and take the ones compliment of the result. Note that the checksum must be recomputed at each hop because the Time to Live field 40 changes. Source field 46 indicates the IP address for the source of the packet and destination field 48 indicates the IP address for the destination of the packet.

Options field 22 is a variable length field designed to hold other information. Currently, options used on the Internet indicate security, suggested routing path, previous routing path and time stamps, among other things. In one embodiment of the present invention, is contemplated that the source and destination's domain names are added to Options field 22. In one alternative, the actual full ACSII strings can be added directly into the options field, first listing the source's domain name and followed by the destination's domain name (or vice versa). In other alternatives, the two domain names can be encoded, compressed, encrypted or otherwise altered to provide more efficient use of storage space, security or compatibility. In embodiments where the domain name is encoded, encrypted, compressed, etc., the information stored is said to represent the domain name. That is, an entity can read that information and extract (or identify) the domain name from that information. That extraction or identification can be by unencoding, decoding, decompressing, unencrypting, etc.

In another embodiment, the domain names of the source, destination or both are added to the end of the data portion (e.g. data field 108 of FIG. 4) of a packet as a trailer. In this case, Length field 30 needs to account for the extra bytes added at the end of the data field. Legacy routers can treat this trailer as an integral part of the data field and ignore it.

Network Layer 14 is comprised of a number of processes running on the source, destination and, possibly, one or more routers. The process(es) implementing the Network Layer on the source or destination machines can be in the operating system kernel, in a separate user process, in a library package, in a network application, on a network interface card or in other suitable configurations.

The network entity, the process implementing the network layer, receives a segment from the transport layer process. The network entity appends a header to the segment to form a packet. The packet is sent to a router on a network or the Internet. Each router has a table listing IP addresses for a number of distant networks and IP addresses for hosts in the network closest to the router. When an IP packet arrives, its destination address is looked up in the routing table. If the packet is for a distant network, it is forwarded to the next router listed in the table. If the distant network is not present in the router's tables, the packet is forwarded to a default router with more extensive tables. If the packet is for a local host (e.g. on the router's Local Area Network (LAN)), it is sent directly to the destination.

Although every machine in the Internet has an IP address, these addresses alone cannot be used for sending packets because the data link layer does not understand Internet addresses. Most hosts are attached to a LAN by an interface board that only understands LAN addresses. For example, every Ethernet board comes equipped with a 48 bit Ethernet address. Manufacturers of Ethernet boards request a block of addresses from a central authority to ensure that no two boards have the same address. The boards send and receive frames based on a 48 bit Ethernet address. For one entity to transmit data to another entity on the same LAN using an Ethernet address, the entity can use the Address Resolution Protocol (ARP). This protocol includes the sender broadcasting a packet onto the Ethernet asking who owns the particular IP address in question. That packet will arrive at every machine on the Ethernet and each machine will check its IP address. The machine that owns the particular IP address will respond with its Ethernet address. The sending machine now has the Ethernet address for sending data directly to the destination on the LAN. At this point, the Data Link Layer 12 on the sender builds an Ethernet frame addressed to the destination, puts the packet into the payload field of the frame and dumps the frame onto the Ethernet. The Ethernet board on the destination receives the frame, recognizes it is a frame for itself, and extracts the IP packet from the frame.

The goal of Transport Layer 16 is to provide efficient and reliable service to its users (processes in Application Layer 18). To achieve this goal, Transport Layer 16 makes use of the services provided in Network Layer 14. The one or more processes that implement the transport layer are called the transport entity. The transport entity can be in the operating system kernel, in a separate user process, in a library package, in network applications or on the network interface card. Typically, executable software implementing a transport entity or a network entity would be stored on a processor readable storage medium (e.g. a hard disk, CD-ROM, floppy disk, tape, memory, etc.).

The transport layer improves the quality of service of the network layer. For example, if a transport entity is informed halfway through a long transmission that its network connection has been abruptly terminated, it can set up a new network connection to the remote transport entity. Using this new network connection, the transport entity can send a query to the destination asking which data arrived and which did not, and then pick up from where it left off. In essence, the existence of Transport Layer 16 makes it possible for a transport service to be more reliable than the underlying network service. Lost data can be detected and compensated for by the Transport Layer 16. Furthermore, transport service primitives can be designed to be independent of the network service primitives, which may vary considerably from network to network.

TCP was specifically designed to provide a reliable end-to-end byte stream over an unreliable internetwork. An internetwork differs from a single network because different parts may have different topologies, bandwidths, delays, packet sizes and other parameters. Each machine supporting TCP has a TCP entity. A TCP entity accepts user data streams from local processes (application layer), breaks them up into pieces and sends each piece as a separate segment to the network entity. When segments arrive at a machine they are given to the TCP entity, which reconstructs the original byte stream. The IP layer gives no guarantee that segments will be delivered properly, so it is up to the TCP entity to time out and retransmit them as need be. Segments that do arrive may do so in the wrong order. It is also up to the TCP entity to reassemble them into messages in the proper sequence. In short, TCP must furnish the reliability that most users want and that the Internet Protocol does not provide.

TCP service is obtained by having both the sender and receiver create endpoints called sockets. Each socket has a socket number (or address) consisting of the IP address of the host and a 16 bit number local to that host called a port. To obtain TCP service, a connection must be explicitly established between a socket on the sending machine and a socket on the receiving machine. Port numbers below 256 are called well known ports and are reserved for standard services. For example, any process wishing to establish a connection to a host to transfer a file using FTP can connect to the destination host port 21. Similarly, to establish a remote log-in session using Telnet, port 23 is used.

When an application wishes to set up a connection to a remote application process, the application process issues a connect primitive requesting that the transport layer set up a connection between two sockets. If the connect succeeds, the process returns a TCP reference number used to identify the connection on subsequent calls. After the connection is established, the application process can issue a send command and pass the TCP reference number with the data (or pointer to data) for sending to the destination. The present invention also requires that when the application issues its connect command, in addition to sending the two socket addresses the application also provides the transport entity with the domain name for the destination. In addition, the operating system or the application should make the domain name of the source available to the connect command. One alternative to accomplish this is to have the operating system retrieve the domain name from the DNR or from a local DNS server through a reverse DNS IP lookup. The source's domain name can be retrieved at start-up time of a node and be made available to the network layer. Another alternative is to have the application provide the domain name of the source either directly or through a reverse DNS IP lookup. These domain names will be associated with the TCP reference number. Alternatively, the domain names can be passed to the transport layer each time a request to send data is made.

When receiving a request to send data, the TCP entity builds a data unit called a segment. The TCP entity interfaces with the network entity by requesting the network entity to either send a packet or receive a packet. A request to send a packet can include up to seven parameters. The first parameter will be a connection identifier. The second parameter is a flag indicating more data is coming. The third parameter indicates the packet type. The fourth parameter is a pointer to the actual data to be transmitted (i.e. the segment). The fifth parameter indicates the number of bytes in the segment. The sixth parameter is the source's domain name. The seventh parameter is the destination's domain name.

The segment that is created by the TCP entity and passed to the IP entity includes a header section and a data section. FIG. 3 shows a layout of the TCP header. The header consists of a fixed format 20 byte header followed by a variable length Options field 80. The entire header is appended to the data to comprise a segment. In FIG. 3, each of the first five rows represent 32 bits. The option field 80 can be one or more 32 bit words. Source field 62 indicates the source's port. Destination field 64 identifies the destination's port. Sequence number field 66 and acknowledge number field 68 are used for tracking the sequence of segments exchanged between the sender and the receiver. Header length field 70 indicates the number of 32 bit words contained in the TCP header. Header length field 70 is followed by six one bit flags 72. The first flag indicates the presence of urgent data. The second flag indicates that the acknowledgment number 68 is valid. The third flag indicates that the data is PUSHed data (data that should be sent immediately). The fourth flag is used to reset a connection. The fifth flag is used to establish connections and the sixth flag is used to release a connection. Window size field 74 indicates the maximum number of bytes that can be sent without waiting for an acknowledgment. Checksum field 76 provides a checksum for the header, the data and a conceptual pseudo header. The pseudo header includes a 32 bit IP address of the source, a 32 bit IP address of the destination, the protocol number for TCP and the byte count for the TCP segment (including the header).

Option field 80 was designed to provide a way to add extra facilities not covered by the regular header. In some instances, the option field is used to allow a host to specify the maximum TCP payload it is willing to accept. In one embodiment of the present invention, the source's domain name and/or destination's domain name are stored in Options Field 80. In another embodiment, the source's and/or destination's domain name are stored in the data portion (see data portion 102 of FIG. 4) of the TCP segment.

The TCP/IP reference model also supports the connectionless transport protocol, UDP. UDP provides a way for applications to send encapsulated raw IP packets and send them without having to establish a connection. A UDP segment consists of an 8 byte header followed by the data. The head includes the source port, destination port, the length of the header and data, and a checksum.

FIG. 4 shows the relationship between segments, packets and frames. When an application issues a request to send data, TCP breaks up the data into segments. The segment includes a header 104 and a payload (data portion) 102. The segment is passed to the IP entity (network layer entity). The IP entity incorporates the segment into the data portion 108 (IP payload) and appends a header 110 to that data portion to form a packet. Thus, the payload for an IP packet includes the TCP segment. The IP packet is then given to the data link layer 12 which takes the packet and appends a header 114 to the packet to create a frame. Thus, the IP packet is the payload 112 for the frame.

The present invention provides for a Domain Name Router (DNR) that uses domain names to route data sent to a destination on a network. The IP address space is divided into global addresses and local address. Global addresses are unique addresses that should only be used by one entity having access to the Internet. Local addresses are used for entities not having direct access to the Internet. Since local addresses are not generally used on the Internet, many private networks can have entities using the same local address. To avoid collisions, no entity should use a local address on the Internet.

Rather than use the entire set of global addresses for a Class A, B or C network, each corporate entity or network can be assigned one or a small number of global address to be used by the DNR. Each of the hosts on the network can be assigned a local address. The same local addresses can be used by many different networks. When a source entity sends data to a destination entity with a local address, the

data is sent to the global address for the destination's network. The source entity embeds the destination's domain name and its own domain name somewhere inside the data. Since the DNR for the destination's network is assigned the global address for the destination's network, the DNR receives the data. The DNR extracts the destination's domain name from the data, translates that domain name to a local address and sends the data to the destination. Note that the source entity could have either a local address or a global address and still be able to utilize the present invention.

FIG. 5 shows two LANs 120 and 122 connected to Internet 140. LAN 120 includes three hosts 132, 134 and 136 connected to each other and to DNR 138. DNR 138 is also connected to Internet 140. Network 122 includes three hosts 150, 152 and 154 connected to each other and to router 156. Router 156 is also connected to Internet 140. DNR 138 is able to route IP packets received from the Internet to a local host (132, 134, 136) by using the domain name in accordance with the present invention. In one embodiment, router 156 is also a DNR; however, router 156 need not be a DNR.

FIG. 6 shows one example of a hardware architecture for a DNR. The DNR includes a processor 202, a memory 204, a mass storage device 206, a portable storage device 208, a first network interface 210, a second network interface 212 and I/O devices 214. Processor 202 can be a Pentium Processor or any other suitable processor. The choice of processor is not critical as long as a suitable processor with sufficient speed and power is chosen. Memory 204 could be any conventional computer memory. Mass storage device 206 could include a hard drive, CD-ROM or any other mass storage device. Portable storage 208 could include a floppy disk drive or other portable storage device. The DNR includes two network interfaces. In other embodiments, the DNR could include more than two network interfaces. The network interfaces can include network cards for connecting to an Ethernet or other type of LAN. In addition, one or more of the network interfaces can include or be connected to a firewall. Typically, one of the network interfaces will be connected to the Internet and the other network interface will be connected to a LAN. I/O devices 214 can include one or more of the following: keyboard, mouse, monitor, front panel, LED display, etc. Any software used to perform the routing methods and/or the methods of FIGS. 10 and 11 are likely to be stored in mass storage 206 (or any form of non-volatile memory), a portable storage media (e.g. floppy disk or tape) and, at some point, in memory 204. The above described hardware architecture is just one suitable example depicted in a generalized and simplified form. The DNR could include software running on a computer, dedicated hardware, a dedicated router with software to implement the domain name routing or other software and/or hardware architectures that are suitable.

In one embodiment, the domain name routing is done at the network layer. Thus, the domain names are inserted into Options field 22 of an IP header. Other embodiments can place the domain names in other portions of an IP packet, including the data portion (such as a trailer to the data). In other alternatives, the domain name can be stored in the options field 80 of a TCP segment, the data portion of a TCP segment, other fields of the TCP segment, data sent from the application layer, or in another data unit. If the domain names are inserted in Options field 22, it is not necessary to place them in Options field 80. Similarly, if the domain names are inserted in Options field 80, it is not necessary that they appear in Options field 22. However, in one embodiment, it may be simpler to place the domain names in multiple data units (e.g. both options fields). The point is that the domain names must be somewhere inside an IP packet, whether it is in the payload (or a trailer) or the header.

FIGS. 7–10 are flow charts which describe the process for sending data according to the present invention. It is assumed that a message is being sent from host 150 to host 132. In this example, it is assumed that host 132 has a local address and host 150 has a global address. For example purposes, it is assumed that host 150 and 132 are computers. Alternatively, host 150 and 152 can be other electronic devices that can communicate on the Internet.

FIG. 7 describes an application layer process predominantly run on host 150. In step 302, host 130 resolves the domain name. The user wants to send data to another process. The user provides the domain name of the destination. A resolver process converts the domain name to an IP address.

Every domain, whether it is a single host or a top level domain, has a set of resource records associated with it. For a single host, the most common resource record is its IP address. When a resolver process gives a domain name to the domain name system, it gets back the resource records associated with that domain name.

A resource record has five fields: domain name, time to live, class, type and value. The time to live field gives an indication of how stable the record is. Information that is highly stable is assigned a large value such as the number of seconds in a day. The third field is the class. For the Internet the class is IN. The fourth field tells the type of resource record. One domain may have many resource records. There are at least eight types of resource records that are important to this discussion: SOA, A, MX, NS, CNAME, PTR, HINFO, and TXT. The value field for an SOA record provides the name of the primary source of information about the name server zone, e-mail address of its administrator, a unique serial number and various flags and time outs in the value field. The value field for an A record holds a 32 bit IP address for the host. The value field for the MX record holds the domain name of the entity willing to accept e-mail for that particular domain name. The NS record specifies name servers. The CNAME record allows aliases to be created in the value field. A PTR record just points to another name in the value field, which allows look up of an IP address for a particular domain name. The value field of the HINFO record indicates the type of machine and operating system that the domain name corresponds to. An example of resource records for a host is found below in Table 1.

TABLE 1

Domain Name	Time to Live	Class	Type	Value
saturn.ttc.com	86400	IN	HINFO	Sun unix
saturn.ttc.com	86400	IN	A	188.68.70.1
saturn.ttc.com	86400	IN	MX	mars.ttc.com

Table 1 includes three resource records for an entity with a domain name of saturn.ttc.com. The first resource record indicates a time to live of 86,400 seconds (one day). The type of record is HINFO and the value indicates that the entity is a Sun workstation running the UNIX operating system. The second line is a resource record of type A, which indicates that the IP address for saturn.ttc.com is 198.68.70.1. The third line indicates that e-mail for saturn.ttc.com should be sent to mars.ttc.com. It is likely that there will be a DNS record, which indicates the IP address for mars.ttc.com.

The DNS name space is divided into non-overlapping zones. Each zone is some part of the Internet space and contains name servers holding the authoritative information about that zone. Normally, a zone will have one primary name server and one or more secondary name servers which get their information from the primary name server. When a resolver process has a query about a domain name, it passes the query to one of the local name servers. If the host being sought falls under the jurisdiction of that name server, then that domain name server returns the authoritative resource record. An authoritative record is one that comes from the authority that manages the record. If, however, the host is remote and no information about the requested host is available locally, the name server sends a query message to the top level name server for the host requested. The top level name server will then provide the resource records to the local name server which may cache the information and forwarded it to the original resolver process. Since the cached information in the local name server is not the authoritative record, the time to live field is used to determine how long to use that information.

In one embodiment, DNR 130 serves as the authority DNS server for the hosts on LAN 120. Thus, DNR 130 would store resource records for host 132. One of the resource records for host 132 would be a type A record correlating the global address of DNR 130 with the domain name for host 132.

Looking back at FIG. 7, after the domain name has been resolved the application process is in possession of the IP address for its desired destination. In step 304, the application process requests the transport layer (e.g. TCP) to establish a connection. A socket must have been set up in both the source and destination. The application process submits the source's socket, the destination's socket, the source's domain name and the destination's domain name to the transport layer. In step 306, the application requests that the transport layer send data. In step 308, the application process may request that the transport layer receive data (optional). In step 310, the connection between the source and destination is closed.

FIG. 8 explains how the transport layer of host 150 sends the data in conjunction with the request by the application layer in the steps of FIG. 7. In step 350, the transport layer (e.g. TCP) receives the connection request from the application layer. In step 352, the transport layer establishes a connection between the source socket and destination socket. In one embodiment, the connection request includes the domain names of the destination and the source. Alternatively, the domain names can be passed during step 354. In step 354, the transport layer receives from the application layer the data to be sent to the destination socket. Step 354 can include actually receiving data or a pointer to data. The data received can be broken up into one or more segments and each of the segments will be sent separately. In step 356, one or more segments are created. Creating the segments includes the step of creating a header (step 358), adding the source's domain name and the destination's domain name to the header or data portion (step 360), and appending the header to the data (step 362). If the domain names are to be added to the IP packet and not to the TCP segment, then step 360 is skipped. After the segment is created, the transport layer sends the segments in step 370. Sending a segment includes passing the segment to the network layer.

FIG. 9 describes the steps taken by the network layer on host 150 to send data in response to the steps of FIG. 8. In step 400, the network layer receives a segment and a request to send a packet on the Internet (or other network). As discussed above, the request to send a packet passes the source and destination domain names. Alternatively, the domain names can be embedded in the data. In step **402**, a packet is created. The step of creating the packet includes creating the header (step **404**), adding the domain names of the source and destination to the header or data portion (step **406**) and appending the header to the data (step **408**). If the domain name is to be added as part of the TCP segment and not part of the IP packet, step **406** can be skipped. After the packet is created in step **402**, the network layer routes the packet in step **410**. The packet is routed from host **150**, through router **156**, through Internet **140** and to DNR **138**. The IP packets routed include the destination IP address of DNR **138** and the source IP address of host **150**, both of which are global addresses. The IP packet also includes the domain name of hosts **132** and **150**. In one embodiment, the IP packet would not include the source's domain name. Note that the steps of FIG. **9** can be repeated for each segment.

In one embodiment, host **150** has a local address and router **156** is a DNR. When an IP packet sent from host **150** is received at router **156**, the local address of host **150** is replaced by the global address of router **156**.

FIG. **10** describes the steps performed by DNR **138** when it receives the IP packet from host **150**. In step **502**, DNR **138** receives the IP packet. In step **504**, DNR **138** identifies the destination's domain name from the packet. Identifying the domain name could include looking for the domain name in the header, data portion or other location in an IP packet, TCP segment, application data, etc. Identifying the domain name may include reading an ASCII string. Alternatively, if the domain names are compressed, encrypted, encoded, etc., then DNR **148** would need to decode, decompress, unencrypt, etc. In step **506**, DNR **138** translates the destination domain name to a local address and in step **508** the packet is routed to the destination with the local address.

FIG. **11** describes one exemplar embodiment for performing the step of translating the destination domain name to a local address (step **506** of FIG. **10**). Other suitable methods of translating a domain name can also be used. Translating a domain name can include less than all of the steps of FIG. **11**. In step **512**, DNR **138** looks up the domain name in a DNR table stored in its memory or other storage device. The DNR table includes domain names and corresponding local addresses. In one embodiment, the DNR table could also include Ethernet addresses. It is also possible that the local network includes multiple DNRs, forming a tree. Thus, the entry in the DNR table for a particular domain name could be just an address for another DNR. The packet would then be sent to another DNR, and the second DNR that would then use the domain name to find the final (or next) local address to the destination or another DNR, etc. The DNR table can be set up manually by the administrator for the network or may be set up automatically through embedded software, firmware or hardware.

In step **514**, the DNR determines whether a record for the domain name was found. If no record was found, then an error message is sent back to host **150** in step **516**. If a record is found, the global address for DNR **138** in the IP packet is replaced with the local address in the table. In step **520**, the checksum for the IP header is adjusted if necessary. Since the destination IP address has changed in the header, the checksum may need to be adjusted accordingly. If the application incorporates information used by the IP packet into its data payload, such application packets may need to be adjusted as a result of the change in destination IP address.

When a packet is received by a host, the Network Layer passes the source and destination domain names to the Transport Layer (at least once for each connection). The Transport Layer may pass the source and destination domain names to the Application Layer. Any of the layers can use the source's domain name to send a reply.

Although FIG. **5** shows DNR **138** connected to and located between the LAN and the Internet, DNR **138** could also be located inside the LAN. The present invention can be used with network paradigms other than the TCP/IP reference model and/or the Internet.

The foregoing detailed description of the invention has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed, and obviously many modifications and variations are possible in light of the above teaching. The described embodiments were chosen in order to best explain the principles of the invention and its practical application to thereby enable others skilled in the art to best utilize the invention in various embodiments and with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the claims appended hereto.

I claim:

1. A method for communicating data, comprising the steps of:

receiving a data unit, said data unit includes a destination address and a first set of information representing a first domain name, said destination address corresponds to each entity in a set of two or more entities, said domain name corresponds to a first entity in said set of entities;

translating said first domain name to a first address, said first address corresponds to said first entity and does not correspond to any other entity in said set of entities; and

sending said data unit to said first entity using said first address.

2. A method according to claim **1**, wherein:

said first set of information includes said first domain name.

3. A method according to claim **1**, wherein:

said first set of information includes said first domain name and a second domain name, said second domain name is associated with a source of said data unit.

4. A method according to claim **1**, wherein:

said first set of information includes a compressed form of said first domain name.

5. A method according to claim **1**, wherein:

said first set of information includes an encoded form of said first domain name.

6. A method according to claim **1**, wherein:

said first set of information includes an encrypted form of said first domain name.

7. A method according to claim **1**, wherein:

said data unit includes a TCP segment.

8. A method according to claim **1**, wherein:

said data unit includes an IP packet.

9. A method according to claim **8**, wherein:

said IP packet includes a header; and

said first set of information is stored in said header.

10. A method according to claim **9**, wherein:

said header includes an options field; and

said first set of information is stored in said options field.

11. A method according to claim **8**, wherein:

said IP packet includes a header; and

said header includes a flag indicating use of domain name routing.

12. A method according to claim 8, wherein:

said IP packet includes a header portion and data portion; and

said first set of information is stored in said data portion.

13. A method according to claim 1, wherein:

said step of translating includes finding a record in a table associated with said first domain name, said record in said table includes said first address; and

said steps of receiving, translating and sending are performed by a router.

14. A method according to claim 1, wherein:

said step of sending includes sending said data unit to a router.

15. A method according to claim 1, wherein:

said step of sending includes routing said data unit to said first entity.

16. A method according to claim 1, wherein:

said destination address is a global address; and

said first address is a local address.

17. A method according to claim 16, further comprising the step of:

replacing said global address in said data unit with said local address, said step of replacing being performed after said step of translating.

18. A method according to claim 17, wherein:

said data unit includes a checksum; and

said method for routing data further comprises the step of adjusting said checksum in said data unit, said step of adjusting said checksum being performed after said step of replacing said global address.

19. A method according to claim 1, further including the step of:

acting as an authority domain name server for a destination, said destination being associated with said first address.

20. A method according to claim 1, wherein:

said step of receiving is performed by a second entity said second entity, corresponds to said destination address.

21. A processor readable storage medium having processor readable code embodied on said processor readable storage medium, said processor readable code for programming a processor to perform a method comprising the steps of:

receiving a data unit, said data unit includes a destination address and a first set of information representing a first domain name, said destination address corresponds to each entity in a set of two or more entities, said domain name corresponds to a first entity in said set of entities;

translating said first domain name to a first address, said first address corresponds to said first entity and does not correspond to any other entity in said set of entities; and

routing said data unit toward said first entity using said first address.

22. A processor readable storage medium according to claim 21, wherein:

said data unit is a TCP segment;

said TCP segment includes a header; and

said first set of information is stored in said header.

23. A processor readable storage medium according to claim 21, wherein:

said data unit is an IP packet;

said IP packet includes a data portion and a header portion; and

said first set of information is stored in said data portion.

24. A processor readable storage medium according to claim 21, wherein:

said data unit is an IP packet;

said IP packet includes a header;

said header includes an options field; and

said first set of information is stored in said options field.

25. A processor readable storage medium according to claim 21, wherein:

said first set of information includes information representing a second domain name, said second domain name associated with a source of said data unit.

26. A processor readable storage medium according to claim 21, wherein:

said step of translating includes finding a record in a table associated with said first domain name, said record in said table includes said first address.

27. A processor readable storage medium according to claim 21, wherein:

said destination address is a global address; and

said first address is a local address.

28. A processor readable storage medium according to claim 27, said method further comprises the step of:

replacing said global address in said data unit with said local address, said step of replacing being performed after said step of translating.

29. A processor readable storage medium according to claim 28, wherein:

said data unit includes a checksum; and

said method further comprises the step of adjusting said checksum in said data unit, said step of adjusting said checksum being performed after said step of replacing said global address.

30. A method for communicating data, comprising the steps of:

receiving a first set of data;

receiving a domain name associated with a destination; and

creating a data unit for use with a protocol below an application layer, said step of creating a data unit includes the steps of creating a header, appending said header to said first set of data and adding a first set of information representing said domain name to said data unit, said header includes a destination address, said domain name being different than said destination address, said destination address corresponds to an intermediate entity associated with a set of two or more destination entities, said domain name corresponds to a first entity in said set of destination entities.

31. A method according to claim 30, wherein:

said data unit is an IP packet.

32. A method according to claim 30, wherein:

said header is an IP header;

said IP header includes an option field; and

said first set of information is stored in said options field.

33. A method according to claim 30, further comprising the step of:

sending said data unit to another entity.

34. A method according to claim 30, wherein:

said step of adding a first set of information adds said first set of information as a trailer to said first set of data.

35. A method according to claim 30, wherein:

said step of adding a first set of information is performed prior to said step of appending said header to said first set of data.

36. A method according to claim 30, further including the steps of:

sending said data unit to said intermediate entity using said destination address for delivery to said first entity, said destination address is a global address;

receiving said data unit at said intermediate entity;

translating said domain name to a local address, said local address corresponds to said first entity and does not correspond to any other entity in said set of entities; and

sending said data unit to said first entity using said local address.

37. A processor readable storage medium having processor readable code embodied on said processor readable storage medium, said processor readable code for programming a processor to perform a method comprising the steps of:

receiving a first set of data;

receiving a domain name associated with a destination; and

creating a data unit for use with a protocol below an application layer, said step of creating a data unit includes the steps of creating a header, appending said header to said first set of data and adding a first set of information representing said domain name to said data unit, said header includes a destination address, said domain name being different than said destination address, said destination address corresponds to an intermediate entity associated with a set of two or more destination entities, said domain name corresponds to a first entity in said set of destination entities.

38. A processor readable storage medium according to claim 37, wherein:

said data unit is an IP packet.

39. A processor readable storage medium according to claim 37, wherein:

said header is an IP header;

said IP header includes an options field; and

said first set of information is stored in said options field.

40. A processor readable storage medium according to claim 37, wherein:

said data unit is an IP packet; and

said step of adding a first set of information adds said first set of information as a trailer to said first set of data.

41. A processor readable storage medium according to claim 37, further including the step of:

sending said data unit to a router using said destination address for delivery to a destination host, said destination address is a global address, said domain name corresponds to said destination host, said destination host addressed by a local address.

42. An apparatus for communicating data, comprising:

a processor;

a first network interface in communication with said processor;

a second network interface in communication with said processor; and

a processor readable storage element in communication with said processor, said processor readable storage element storing processor readable code for programming said processor, said processor readable code comprising:

first code for receiving a data unit at said first network interface, said data unit includes a global address and a first set of information representing a first domain name, said global address corresponds to said apparatus, said domain name corresponds to a first entity in a set of entities not including said apparatus,

second code for translating said first domain name to a local address, said local address corresponds to said first entity and does not correspond to any other entity in said set of entities, and

third code for sending said data unit to said first entity using said second network interface and said local address.

43. An apparatus according to claim 42, wherein:

said second network interface is an Ethernet interface.

44. An apparatus according to claim 42, wherein:

said processor readable storage element stores a table, said table includes a set of records, each record of said set of records includes a domain name and a local address.

45. An apparatus according to claim 42, wherein:

said processor readable storage element stores a table, said table includes a set of records, each record of said set of records includes a domain name and a global address.

46. An apparatus according to claim 42, wherein:

said data unit is an IP packet;

said IP packet includes a header portion and a data portion; and

said first set of information is stored in said data portion.

47. An apparatus according to claim 42, wherein:

said second code replaces said global address in said data unit with said local address.

48. An apparatus according to claim 47, wherein:

said data unit includes a checksum; and

said second code adjusts said checksum in said data unit.

* * * * *

United States Patent [19]
Wesinger, Jr. et al.

[11] **Patent Number: 6,052,788**
[45] **Date of Patent: Apr. 18, 2000**

[54] FIREWALL PROVIDING ENHANCED NETWORK SECURITY AND USER TRANSPARENCY

[75] Inventors: **Ralph E. Wesinger, Jr.**, San Jose; **Christopher D. Coley**, Morgan Hill, both of Calif.

[73] Assignee: **Network Engineering Software, Inc.**, San Jose, Calif.

[21] Appl. No.: **09/299,941**

[22] Filed: **Apr. 26, 1999**

Related U.S. Application Data

[63] Continuation of application No. 08/733,361, Oct. 17, 1996, Pat. No. 5,898,830.

[51] **Int. Cl.**^{7} **G06F 13/00**
[52] **U.S. Cl.** 713/201; 713/200; 714/4; 714/18; 380/4; 380/25
[58] **Field of Search** 713/200, 201; 714/4, 18; 380/3, 23, 25; 340/825.34

[56] **References Cited**

U.S. PATENT DOCUMENTS

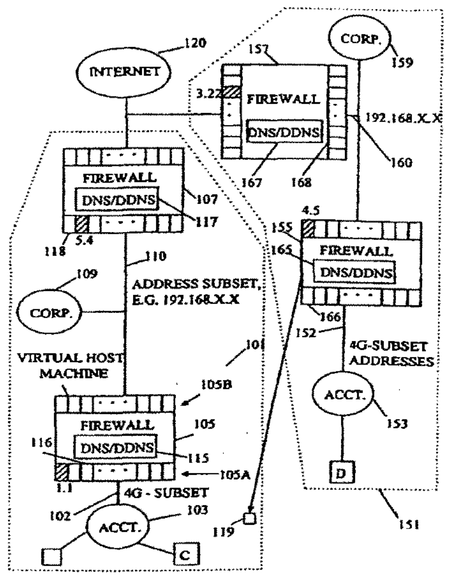
5,617,540	4/1997	Civanlar et al.	395/200.11
5,633,371	6/1997	Yu	395/500
5,671,279	9/1997	Elgamal	380/23
5,673,322	9/1997	Pepe et al.	380/49
5,734,865	3/1998	Yu	395/500
5,864,683	1/1999	Boebart et al.	395/200.79
5,870,550	2/1999	Wesinger, Jr. et al.	395/200.48
5,878,212	3/1999	Civanlar et al.	395/200.33
5,898,830	4/1999	Wesinger, Jr. et al.	395/187.01
5,935,245	8/1999	Sherer	713/200

Primary Examiner—Norman Michael Wright
Attorney, Agent, or Firm—McDonnell Boehnen Hulbert & Berghoff

[57] **ABSTRACT**

The present invention, generally speaking, provides a firewall that achieves maximum network security and maximum user convenience. The firewall employs "envoys" that exhibit the security robustness of prior-art proxies and the transparency and ease-of-use of prior-art packet filters, combining the best of both worlds. No traffic can pass through the firewall unless the firewall has established an envoy for that traffic. Both connection-oriented (e.g., TCP) and connectionless (e.g., UDP-based) services may be handled using envoys. Establishment of an envoy may be subjected to a myriad of tests to "qualify" the user, the requested communication, or both. Therefore, a high level of security may be achieved. The usual added burden of prior-art proxy systems is avoided in such a way as to achieve full transparency—the user can use standard applications and need not even know of the existence of the firewall. To achieve full transparency, the firewall is configured as two or more sets of virtual hosts. The firewall is, therefore, "multihomed," each home being independently configurable. One set of hosts responds to addresses on a first network interface of the firewall. Another set of hosts responds to addresses on a second network interface of the firewall. In one aspect, programmable transparency is achieved by establishing DNS mappings between remote hosts to be accessed through one of the network interfaces and respective virtual hosts on that interface. In another aspect, automatic transparency may be achieved using code for dynamically mapping remote hosts to virtual hosts in accordance with a technique referred to herein as dynamic DNS, or DDNS.

6 Claims, 9 Drawing Sheets



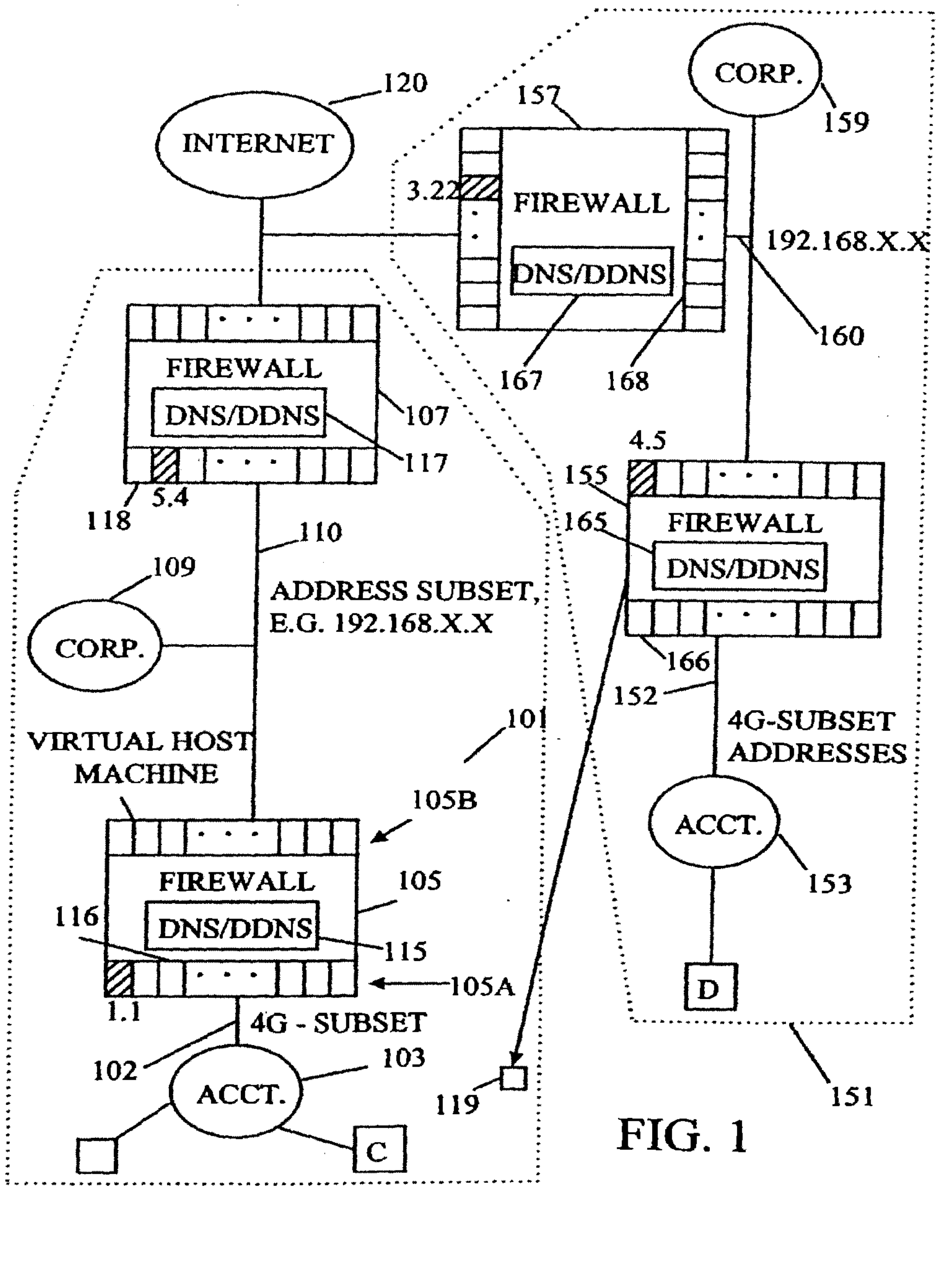

INTERNET
120
157
3.22
FIREWALL
DNS/DDNS
167
168
CORP.
159
192.168.X.X
160
FIREWALL
DNS/DDNS
107
117
118
5.4
110
109
CORP.
ADDRESS SUBSET, E.G. 192.168.X.X
4.5
155
165
FIREWALL
DNS/DDNS
166
152
4G-SUBSET ADDRESSES
ACCT.
153
D
151
VIRTUAL HOST MACHINE
101
105B
FIREWALL
105
116
DNS/DDNS
115
105A
1.1
4G - SUBSET
102
103
ACCT.
C
119

FIG. 1

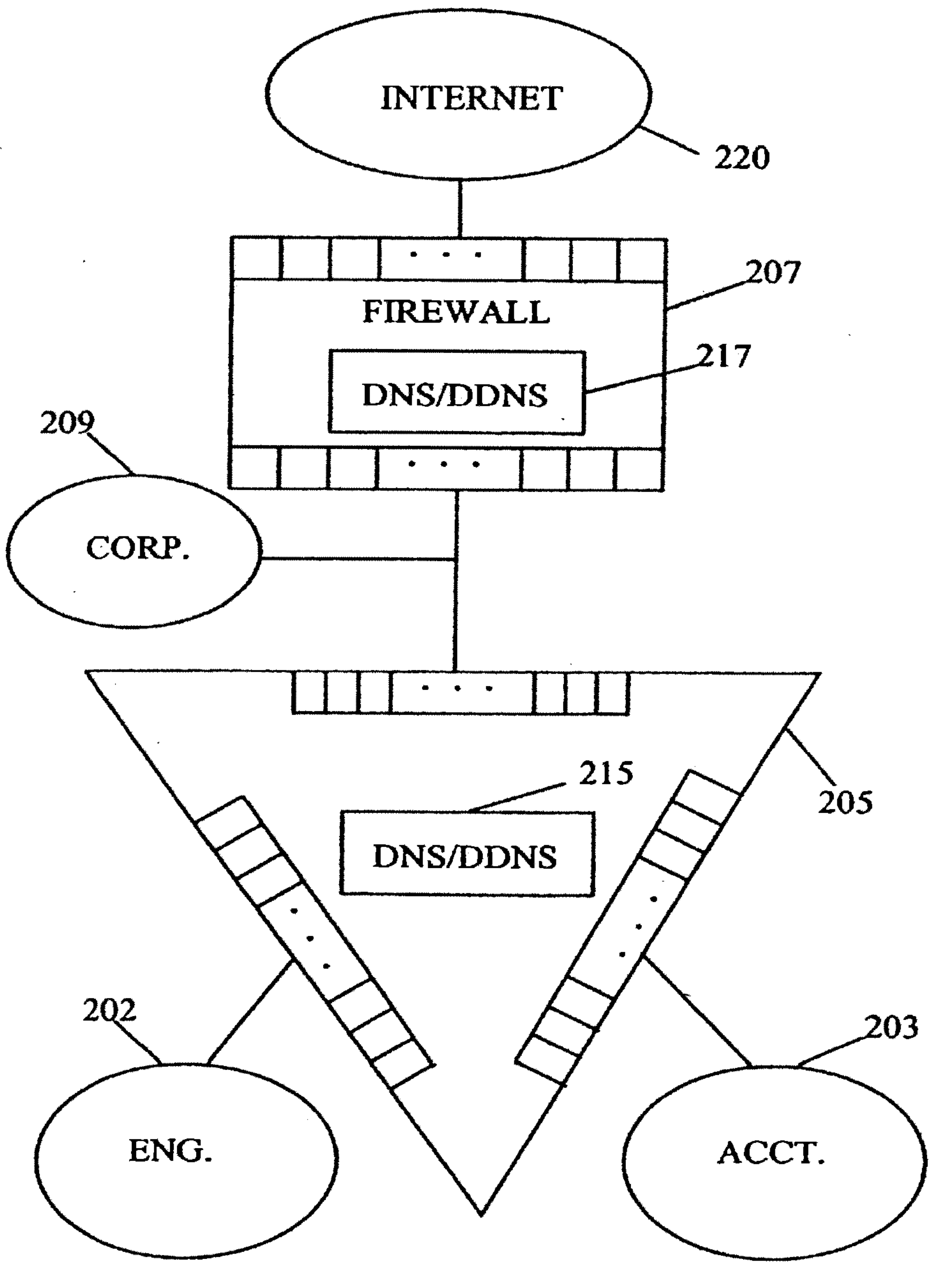

INTERNET
220
FIREWALL
207
DNS/DDNS
217
209
CORP.
215
DNS/DDNS
205
202
203
ENG.
ACCT.

FIG. 2

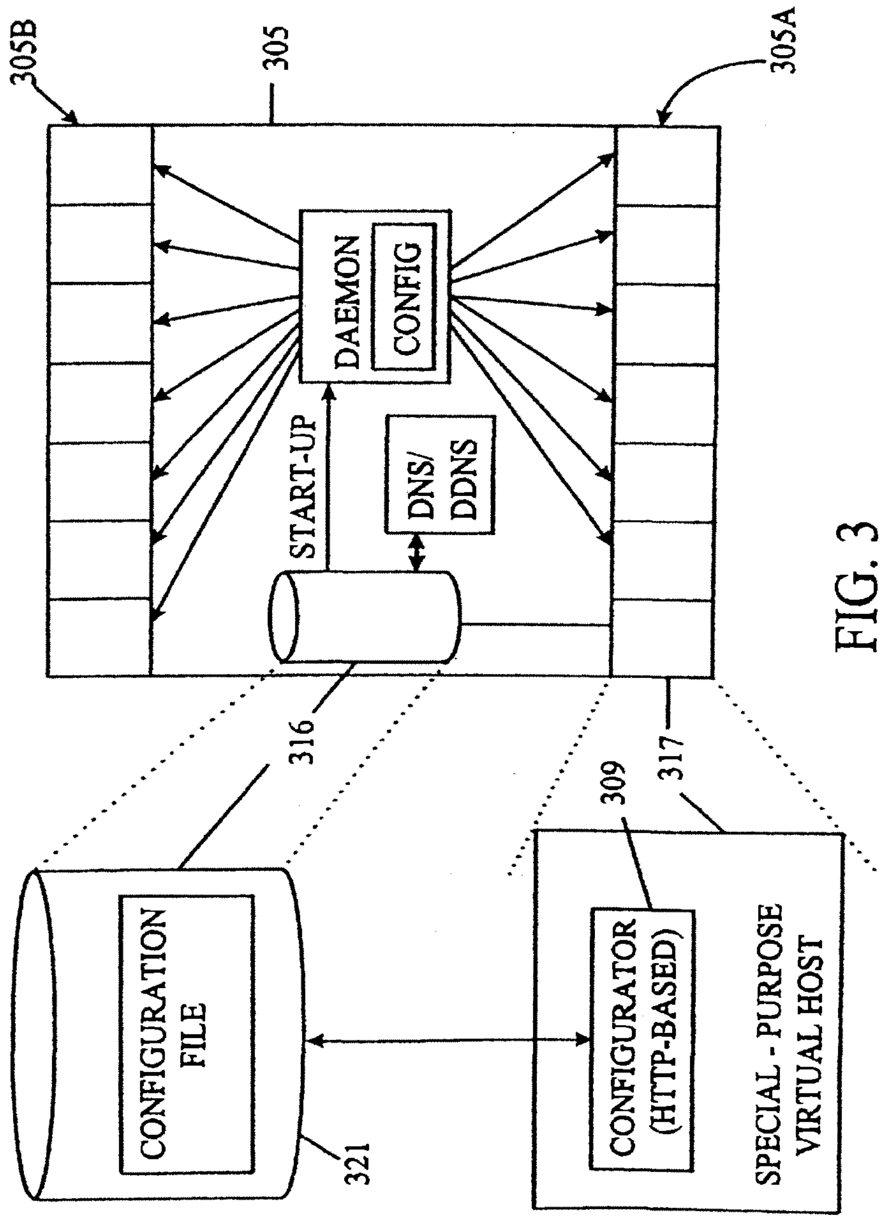

305B
305
305A
DAEMON
CONFIG
START-UP
DNS/
DDNS
316
317
309
CONFIGURATION
FILE
CONFIGURATOR
(HTTP-BASED)
SPECIAL - PURPOSE
VIRTUAL HOST
321

FIG. 3

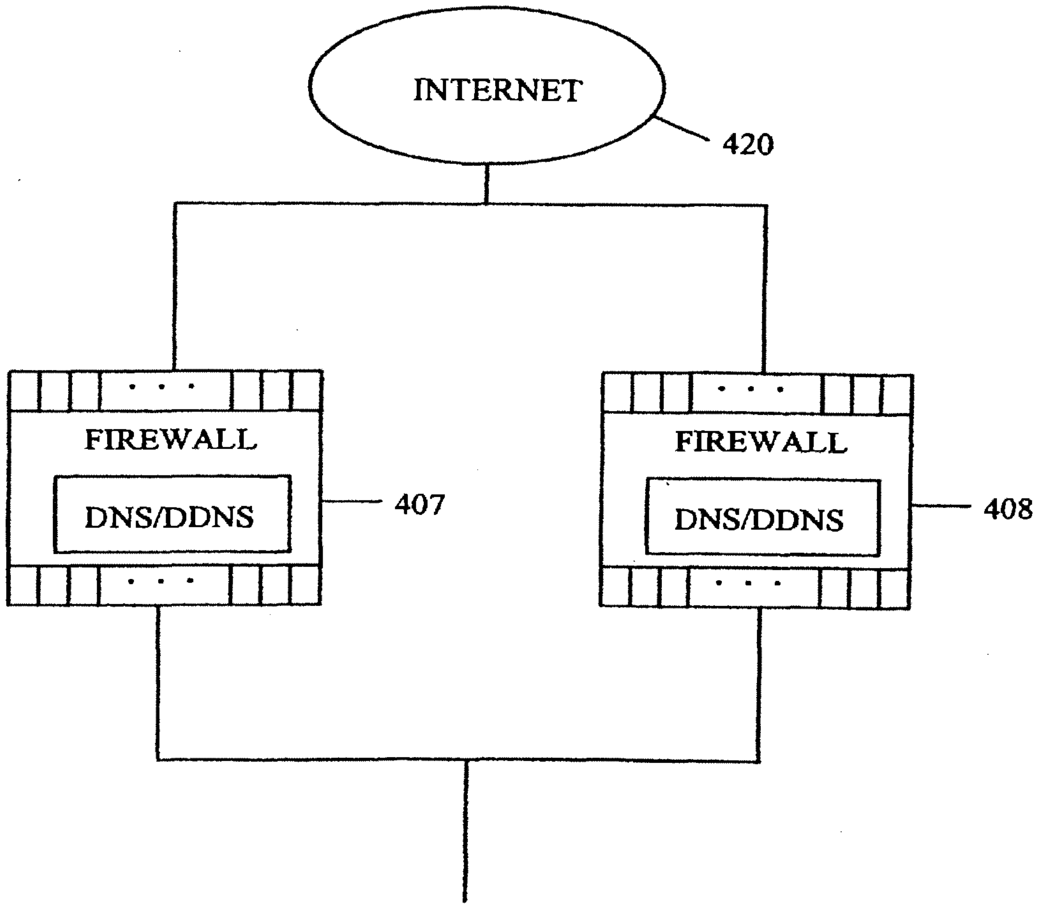


FIG. 4

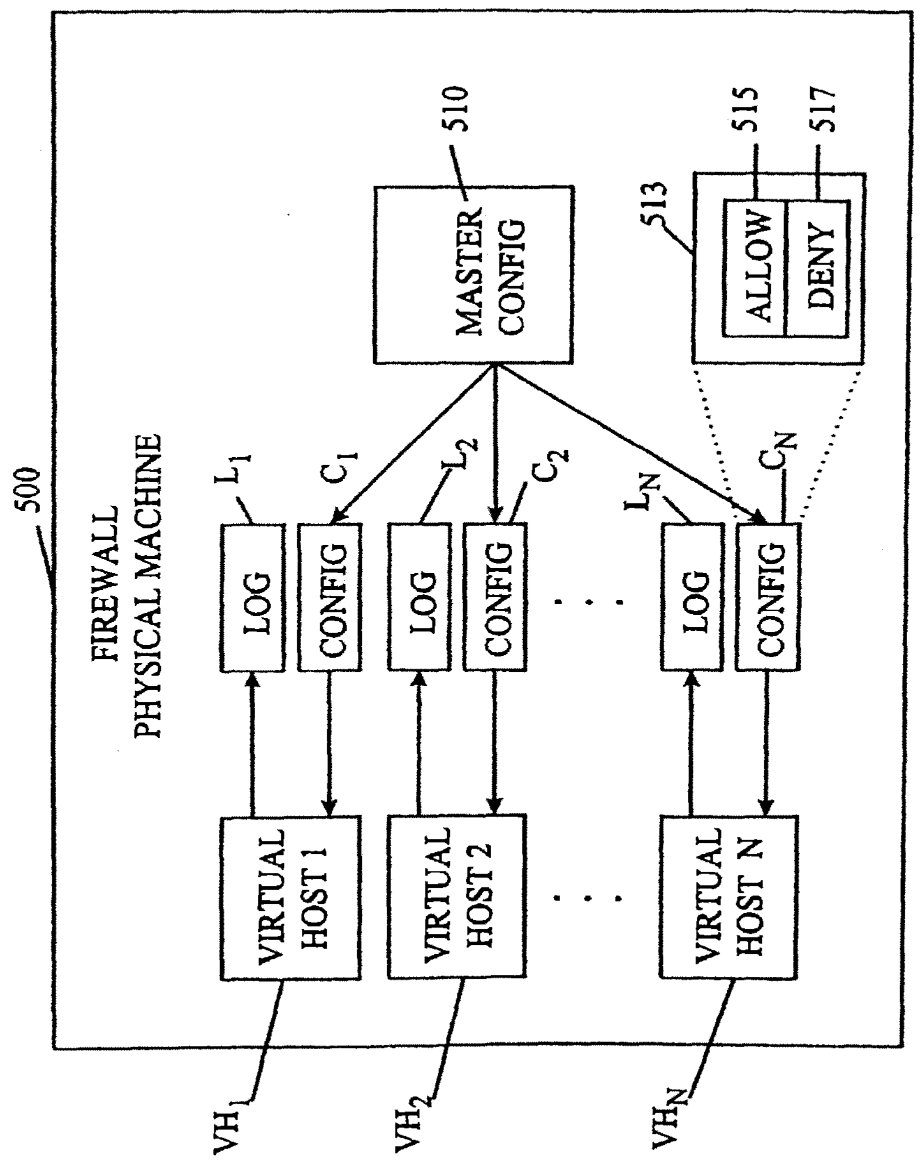

500
FIREWALL
PHYSICAL MACHINE
MASTER CONFIG
510
513
ALLOW
DENY
515
517
L_1
C_1
L_2
C_2
L_N
C_N
LOG
CONFIG
LOG
CONFIG
LOG
CONFIG
VIRTUAL HOST 1
VIRTUAL HOST 2
VIRTUAL HOST N
VH_1
VH_2
VH_N

FIG. 5

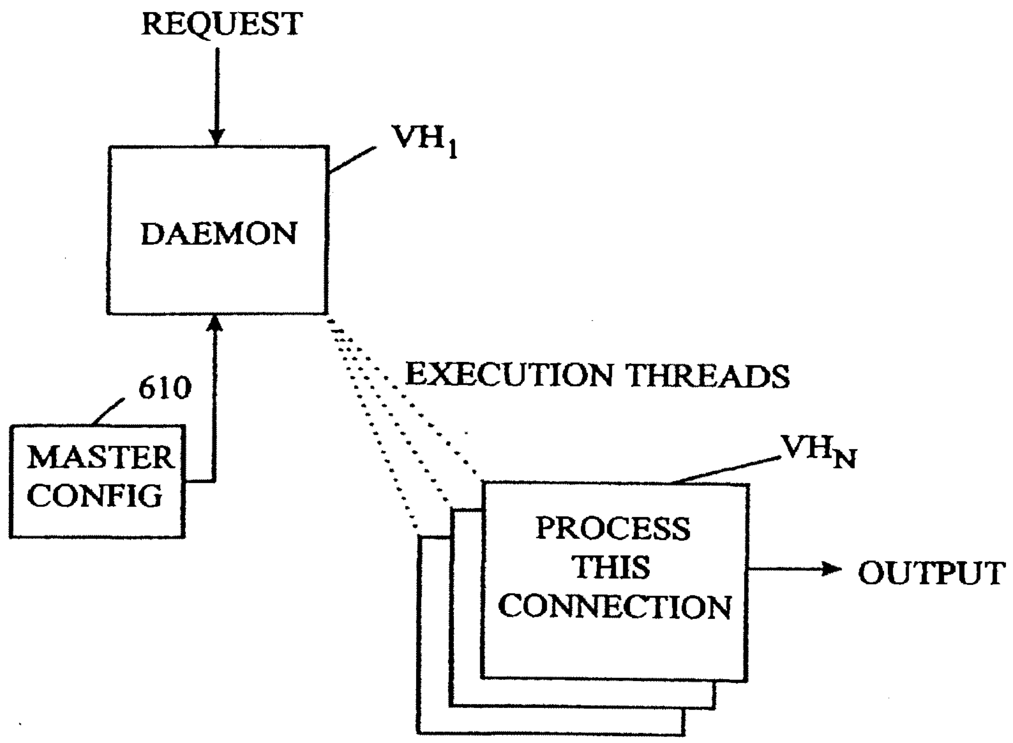

REQUEST
VH$_1$
DAEMON
610
MASTER CONFIG
EXECUTION THREADS
VH$_N$
PROCESS THIS CONNECTION
OUTPUT

FIG. 6

```
PORT             = 80
RULE1            = {
                 TIME             = "1AM-12PM"
}
WWW.SRMC.COM                      = {
                 .CGI             = "PROCESSCGI"
                 ROOT             = "/HOME/SRMC/HTML"
}
WWW.HONOLULU.NET                  = {
                 .CGI             = ""
                 ROOT             = "/HOME/HONOLULU/HTML"
}
WWW.SANJOSE.NET                   = {
                 .CGI             = "PROCESSCGI"
                 ALLOW            = {
                                  *.SRMC.COM
                                  205.138.192.*
                                  205.138.192.0/23
                 }
                 DENY             = {
                                  MISTERPAIN.COM
                 }
}
WWPROXY.SRMC.COM                  = {
                 MODE             = RT_SERVERPROXY
}
NS.SRMC.COM                       = {
                 ALLOW            = {
                                  192.168.0.*
                                  192.168.1.*    = RULE1
                                  192.168.2.*    = {
                                                 TIME          = "1AM-12PM"
                                  }
                                  192.168.3.*    = 192.168.2.*
                 }
}
MJU.SRMC.COM                      = {
                 ALLOW            = {
                                  192.168.0.0/23               = RULE1
                 }
                 DENY             = {
                                  192.168.0.*    = {
                                                 TIME          = "12PM-1AM"
                                  }
                 }
}
```

FIG. 7

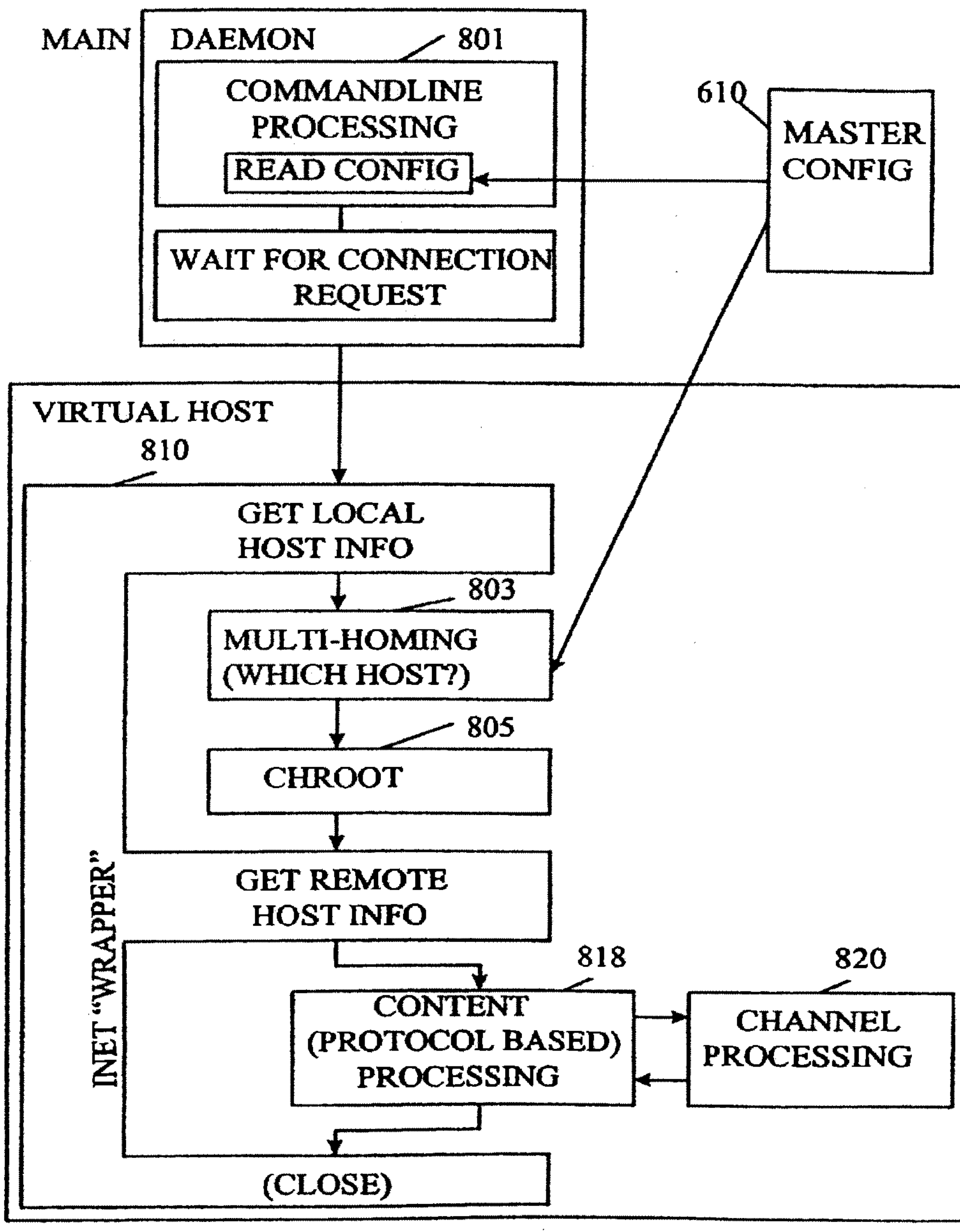

MAIN
DAEMON
801
COMMANDLINE PROCESSING
READ CONFIG
610
MASTER CONFIG
WAIT FOR CONNECTION REQUEST
VIRTUAL HOST
810
GET LOCAL HOST INFO
803
MULTI-HOMING (WHICH HOST?)
805
CHROOT
INET "WRAPPER"
GET REMOTE HOST INFO
818
820
CONTENT (PROTOCOL BASED) PROCESSING
CHANNEL PROCESSING
(CLOSE)

FIG. 8

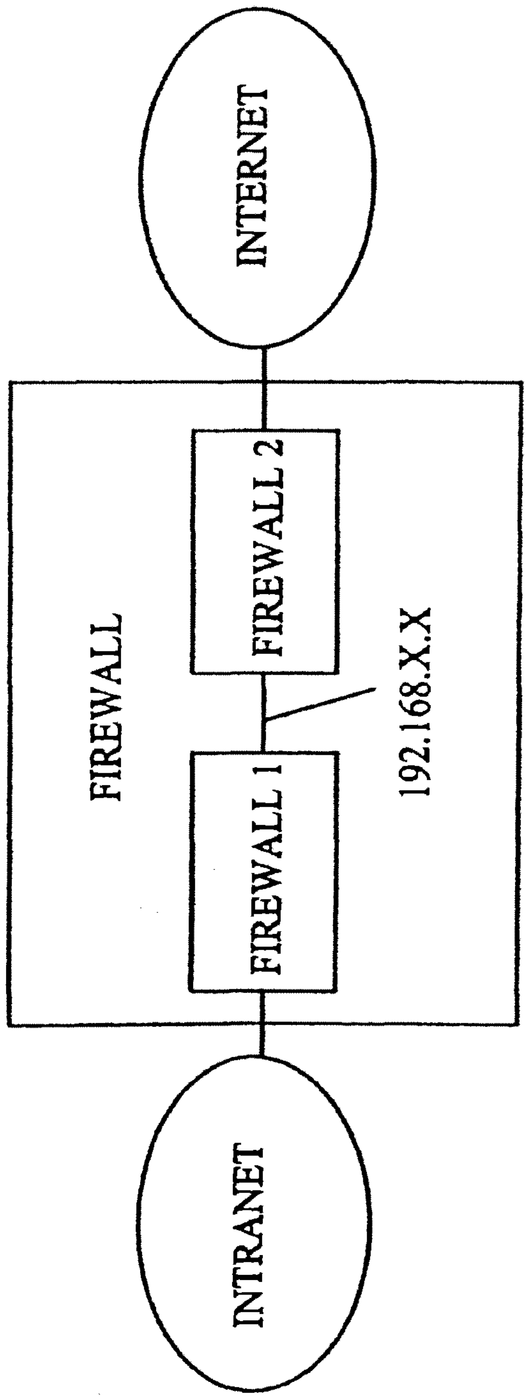

INTERNET
FIREWALL
FIREWALL 2
192.168.X.X
FIREWALL 1
INTRANET

FIG. 9

FIREWALL PROVIDING ENHANCED NETWORK SECURITY AND USER TRANSPARENCY

This is a continuation of patent application Ser. No. 08/733,361, filed Oct. 17, 1996, now U.S. Pat. No. 5,898, 830, issued on Apr. 27, 1999, entitled, "Firewall Providing Enhanced Network Security And User Transparency", invented by Wesinger, Jr. et al.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to computer network security and more particularly to firewalls, i.e., a combination of computer hardware and software that selectively allows "acceptable" computer transmissions to pass through it and disallows other non-acceptable computer transmissions.

2. State of the Art

In the space of just a few years, the Internet-because it provides access to information, and the ability to publish information, in revolutionary ways-has emerged from relative obscurity to international prominence. Whereas in general an internet is a network of networks, the Internet is a global collection of interconnected local, mid-level, and wide-area networks that use the Internet Protocol (IP) as the network layer protocol. Whereas the Internet embraces many local- and wide-area networks, a given local- or wide-area network may or may not form part of the Internet. For purposes of the present specification, a "wide-area network" (WAN) is a network that links at least two LANs over a wide geographical area via one or more dedicated connections. The public switched telephone network is an example of a wide-area network. A "local-area network" (LAN) is a network that takes advantage of the proximity of computers to typically offer relatively efficient, higher speed communications than wide-area networks.

In addition, a network may use the same underlying technologies as the Internet. Such a network is referred to herein as an "Intranet," an internal network based on Internet standards. Because the Internet has become the most pervasive and successful open networking standard, basing internal networks on the same standard is very attractive economically. Corporate Intranets have become a strong driving force in the marketplace of network products and services.

The present invention is directed primarily toward the connection of an Intranet to the Internet and the connection of intranets to other intranets, and any network connection where security is an issue.

As the Internet and its underlying technologies have become increasingly familiar, attention has become focused on Internet security and computer network security in general. With unprecedented access to information has also come unprecedented opportunities to gain unauthorized access to data, change data, destroy data, make unauthorized use of computer resources, interfere with the intended use of computer resources, etc. As experience has shown, the frontier of cyberspace has its share of scofflaws, resulting in increased efforts to protect the data, resources, and reputations of those embracing intranets and the Internet. Firewalls are intended to shield data and resources from the potential ravages of computer network intruders. In essence, a firewall functions as a mechanism which monitors and controls the flow of data between two networks. All communications, e.g., data packets, which flow between the networks in either direction must pass through the firewall; otherwise, security

is circumvented. The firewall selectively permits the communications to pass from one network to the other, to provide bidirectional security.

Ideally, a firewall would be able to prevent any and all security breaches and attacks. Although absolute security is indeed a goal to be sought after, due to many variables (e.g., physical intrusion into the physical plant) it may be difficult to achieve. However, in many instances, it is of equal if not greater importance to be alerted to an attack so that measures may be taken to thwart the attack or render it harmless, and to avoid future attacks of the same kind. Hence a firewall, in addition to security, should provide timely information that enables attacks to be detected.

Firewalls have typically relied on some combination of two techniques affording network protection: packet filtering and proxy services.

Packet filtering is the action a firewall takes to selectively control the flow of data to and from a network. Packet filters allow or block packets, usually while routing them from one network to another (often from the Internet to an internal network, and vice versa). To accomplish packet filtering, a network administrator establishes a set of rules that specify what types of packets (e.g., those to or from a particular IP address or port) are to be allowed to pass and what types are to be blocked. Packet filtering may occur in a router, in a bridge, or on an individual host computer.

Packet filters are typically configured in a "default permit stance"; i.e., that which is not expressly prohibited is permitted. In order for a packet filter to prohibit potentially harmful traffic, it must know what the constituent packets of that traffic look like. However, it is virtually impossible to catalogue all the various types of potentially harmful packets and to distinguish them from benign packet traffic. The filtering function required to do so is too complex. Hence, while most packet filters may be effective in dealing with the most common types of network security threats, this methodology presents many chinks that an experienced hacker may exploit. The level of security afforded by packet filtering, therefore, leaves much to be desired.

Recently, a further network security technique termed "stateful inspection" has emerged. Stateful inspection performs packet filtering not on the basis of a single packet, but on the basis of some historical window of packets on the same port. Although stateful inspection may enhance the level of security achievable using packet filtering, it is as yet relatively unproven. Furthermore, although an historical window of packets may enable the filter to more accurately identify harmful packets, the filter must still know what it is looking for. Building a filter with sufficient intelligence to deal with the almost infinite variety of possible packets and packet sequences is liable to prove an exceedingly difficult task.

The other principal methodology used in present-day firewalls is proxies. In order to describe prior-art proxy-based firewalls, some further definitions are required. A "node" is an entity that participates in network communications. A subnetwork is a portion of a network, or a physically independent network, that may share network addresses with other portions of the network. An intermediate system is a node that is connected to more than one subnetwork and that has the role of forwarding data from one subnetwork to the other (i.e., a "router").

A proxy is a program, running on an intermediate system, that deals with servers (e.g., Web servers, FTP servers, etc.) on behalf of clients. Clients, e.g. computer applications which are attempting to communicate with a network that is protected by a firewall, send requests for connections to proxy-based intermediate systems. Proxy-based intermediate systems relay approved client requests to target servers and relay answers back to clients.

Proxies require either custom software (i.e., proxy-aware applications) or custom user procedures in order to establish a connection. Using custom software for proxying presents several problems. Appropriate custom client software is often available only for certain platforms, and the software available for a particular platform may not be the software that users prefer. Furthermore, using custom client software, users must perform extra manual configuration to direct the software to contact the proxy on the intermediate system. With the custom procedure approach, the user tells the client to connect to the proxy and then tells the proxy which host to connect to. Typically, the user will first enter the name of a firewall that the user wishes to connect through. The firewall will then prompt the user for the name of the remote host the user wishes to connect to. Although this procedure is relatively simple in the case of a connection that traverses only a single firewall, as network systems grow in complexity, a connection may traverse several firewalls. Establishing a proxied connection in such a situation starts to become a confusing maze, and a significant burden to the user, since the user must know the route the connection is to take.

Furthermore, since proxies must typically prompt the user or the client software for a destination using a specific protocol, they are protocol-specific. Separate proxies are therefore required for each protocol that is to be used.

Another problematic aspect of conventional firewall arrangements, from a security perspective, is the common practice of combining a firewall with other packages on the same computing system. The firewall package itself may be a combination of applications. For example, one well-known firewall is a combination Web server and firewall. In other cases, unrelated services may be hosted on the same computing platform used for the firewall. Such services may include e-mail, Web servers, databases, etc. The provision of applications in addition to the firewall on a computing system provides a path through which a hacker can potentially get around the security provided by the firewall. Combining other applications on the same machine as a firewall also has the result of allowing a greater number of users access to the machine. The likelihood then increases that a user will, deliberately or inadvertently, cause a security breach.

There remains a need for a firewall that achieves both maximum security and maximum user convenience, such that the steps required to establish a connection are transparent to the user. The present invention addresses this need.

SUMMARY OF THE INVENTION

The present invention, generally speaking, provides a firewall that achieves maximum network security and maximum user convenience. The firewall employs "envoys" that exhibit the security robustness of prior-art proxies and the transparency and ease-of-use of prior-art packet filters, combining the best of both worlds. No traffic can pass through the firewall unless the firewall has established an envoy for that traffic. Both connection-oriented (e.g., TCP) and connectionless (e.g., UDP-based) services may be handled using envoys. Establishment of an envoy may be subjected to a myriad of tests to "qualify" the user, the requested communication, or both. Therefore, a high level of security may be achieved.

Security may be further enhanced using out-of-band authentication. In this approach, a communication channel, or medium, other than the one over which the network communication is to take place, is used to transmit or convey an access key. The key may be transmitted from a remote location (e.g, using a pager or other transmission device) or may be conveyed locally using a hardware token, for example. To gain access, a hacker must have access to a device (e.g., a pager, a token etc.) used to receive the out-of-band information. Pager beep-back or similar authentication techniques may be especially advantageous in that, if a hacker attempts unauthorized access to a machine while the authorized user is in possession of the device, the user will be alerted by the device unexpectedly receiving the access key. The key is unique to each transmission, such that even if a hacker is able to obtain it, it cannot be used at other times or places or with respect to any other connection.

Using envoys, the added burden associated with prior-art proxy systems is avoided so as to achieve full transparency—the user can use standard applications and need not even know of the existence of the firewall. To achieve full transparency, the firewall is configured as two sets of virtual hosts. The firewall is, therefore, "multi-homed," each home being independently configurable. One set of hosts responds to addresses on a first network interface of the firewall. Another set of hosts responds to addresses on a second network interface of the firewall. In accordance with one aspect of the invention, programmable transparency is achieved by establishing DNS mappings between remote hosts to be accessed through one of the network interfaces and respective virtual hosts on that interface. In accordance with another aspect of the invention, automatic transparency may be achieved using code for dynamically mapping remote hosts to virtual hosts in accordance with a technique referred to herein as dynamic DNS, or DDNS.

The firewall may have more than two network interfaces, each with its own set of virtual hosts. Multiple firewalls may be used to isolate multiple network layers. The full transparency attribute of a single firewall system remains unchanged in a multi-layered system: a user may, if authorized, access a remote host multiple network layers removed, without knowing of the existence of any of the multiple firewalls in the system.

Furthermore, the firewalls may be configured to also transparently perform any of various kinds of channel processing, including various types of encryption and decryption, compression and decompression, etc. In this way, virtual private networks may be established whereby two remote machines communicate securely, regardless of the degree of proximity or separation, in the same manner as if the machines were on the same local area network.

The problem of Internet address scarcity may also be addressed using multi-layer network systems of the type described. Whereas addresses on both sides of a single firewall must be unique in order to avoid routing errors, network segments separated by multiple firewalls may reuse the same addresses.

BRIEF DESCRIPTION OF THE DRAWING

The present invention may be further understood from the following description in conjunction with the appended drawing. In the drawing:

FIG. 1 is a block diagram of a multi-layered computer enterprise network in which the present invention may be used;

FIG. 2 is a block diagram of a network similar to the network of FIG. 1 but in which a two-sided firewall has been replaced by a three-sided firewall;

FIG. 3 is a block diagram showing in greater detail a special-purpose virtual host used for configuration of a firewall;

FIG. 4 is a block diagram of a load-sharing firewall;

FIG. 5 is a block diagram of one embodiment of the firewall of the present invention;

FIG. 6 is a block diagram illustrating the manner in which the present firewall handles connection requests;

FIG. 7 is an example of a portion of the master configuration file of FIG. 5;

FIG. 8 is a block diagram illustrating in greater detail the structure of the present firewall; and

FIG. 9 is a block diagram of a combination firewall that allows the bulk of the entire Internet address space to be used on both sides of the firewall.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The following terms are used in the present specification in accordance with the following definitions:

Concept/Feature	Definition
Multi-homing	Multiple virtual hosts running on a single physical machine, using multiple network addresses on a single network interface. A virtual host assumes the identity of one of multiple, independently-configurable "homes" to handle a particular connection at a particular time.
Programmable transparency	The ability to establish a connection through a firewall without requiring that the user be aware of the firewall.
Envoy	An intervening program that functions as a transparent applications gateway.
Multi-layering	The use of programmable transparency to achieve end-to-end connection across an arbitrary number of networks that are connected by multiple multi-homing firewalls.
Configurator	Code that provides a Web-like interface, accessible remotely through a secure port, for configuring a firewall.
N-dimensional firewall	A firewall having N network interfaces and configured to provide multiple virtual hosts for each interface.
Out-of-band authentication	In deciding whether to allow or disallow a connection by a user, the use of information communicated to the user through means other than the desired connection.
Channel processing	Processing performed on data flowing through a communications channel to enhance some attribute of the data, such as security, reproduction quality, content, etc.
Virtual private network	An internet in which envoys (intervening programs) are used to perform encrypted communications from one secure network to another through a non-secure network.
DDNS	The dynamic assignment of network addresses to virtual hosts on a time-limited basis.
Load sharing	The use of DDNS to assign a network address for a particular connection to a virtual host on one of multiple machines based on the load of the machines.

-continued

Concept/Feature	Definition
Address reuse	The use of the same network address within different networks separated by firewalls.
Programmable transparency-connectionless protocols	The use of envoys for connectionless (e.g, UDP) communications in which a time-out value is used to achieve the equivalent of a connection.

The present firewall provides a choke point used to control the flow of data between two networks. One of the two networks may be the Internet, or both of the two networks may be intranets—the nature and identity of the two networks is immaterial. The important point is that all traffic between the two networks must pass through a single, narrow point of controlled access. A firewall therefore brings a great deal of leverage to bear on the problem of network security, allowing security measures to be concentrated on this controlled access point. To avoid possible security compromises, the firewall should ideally run on a dedicated computer, i.e. one which does not have any other user-accessible programs running on it that could provide a path via which communications could circumvent the firewall.

One environment in which firewalls are particularly desirable is in enterprise network systems, in which a number of individual networks that may be respectively associated with different departments or divisions of a company, for example, are connected with one another. In such an environment, firewalls can be employed to restrict access to the individual networks. While not limited to this particular situation, the present invention will be described hereinafter in such a context, to facilitate an understanding of its underlying principles.

Referring now to FIG. 1, assume that the accounting departments of two remote corporate sites are networked, and that these two different accounting networks are to be connected via the Internet or a similar non-secure, wide-area network. For purposes of illustration, a first site 101 having a first accounting network 103 might be located in California, and a second site 151 having a second accounting network 153 might be located in Japan. Within each site, each accounting network may be part of a larger corporate network (109, 159). Precautions are required to safeguard sensitive accounting data such that it cannot be accessed over the general corporate network. A first firewall (105, 155) is used for this purpose. The first firewall is interposed between the accounting network and the general corporate network.

A convenient way to place the two accounting networks in communication with each other is through the Internet 120, which comprises another layer of a multi-layer network. As compared to other forms of connection, the Internet may be more economical, more easily accessible, and more robust. Connecting to the Internet, however, requires that access between the Internet and the respective sites be strictly controlled. A second firewall (107, 157) is used at each site for this purpose.

In the following description, the present firewall is illustrated most often as a rectangle having along each of two edges thereof a network connection and a row of boxes representing multiple "homes," corresponding to respective virtual hosts. A virtual host along one edge may be used to initiate a connection only in response to a request from the network connection that enters the firewall at that edge. The connection, once established, is fully bi-directional, with the same virtual host passing data between the originating network connection and the network connection at the opposite edge of the firewall.

More generally, the firewall may be N-sided, having N network connections and being illustrated as an N-sided polygon. Any virtual host may establish a connection between any pair of network connections so long as the connection originated from the network connection adjoining that virtual host. Again, the connection, once established, is fully bi-directional.

The firewalls **105**, **107**, **155** and **157** are each of a construction to be more particularly described hereinafter. Each firewall is multi-homing. This means that each firewall is configured as multiple virtual hosts running on a physical computer. In the example of FIG. **1**, a firewall is depicted as a single computer having multiple virtual hosts on each of its two interfaces. In practice, the multiple virtual hosts can be configured in this manner or, alternatively, implemented in any number of computers, as explained in detail hereinafter. Each virtual host corresponds to a "home", i.e. a site via which a connection is made between the two networks on either side of the firewall. At different times, the same virtual host might correspond to different homes associated with different connections. At any given time, however, a virtual host represents one home. In the following description of the particular example illustrated in FIG. **1**, therefore, homes and virtual hosts are described as being synonymous with one another. Each virtual host is fully independently configurable and unique from each of the other virtual hosts. Considering the firewall **105** as being exemplary of each of the firewalls **105**, **107**, **155** and **157**, one set of hosts **105*a*** responds to addresses on a first network interface of the firewall. Another set of hosts **105*b*** responds to addresses on a second network interface of the firewall.

Normally, in accordance with the prior art, connecting from one computer to another remote computer along a route traversing one or more firewalls would require the user to configure a prior-art proxy for each firewall to be traversed. In accordance with one aspect of the invention, however, programmable transparency is achieved by establishing DNS mappings between remote hosts to be accessed through one of the network interfaces and respective virtual hosts on that interface.

DNS is a distributed database system that translates host names to IP addresses and IP addresses to host names (e.g, it might translate host name homer.odyssey.com to 129.186.424.43). The information required to perform such translations is stored in DNS tables. Any program that uses host names can be a DNS client. DNS is designed to translate and forward queries and responses between clients and servers.

When a client needs a particular piece of information (e.g., the IP address of homer.odyssey.com), it asks its local DNS server for that information. The local DNS server first examines its own local memory, such as a cache, to see if it already knows the answer to the client's query. If not, the local DNS server asks other DNS servers, in turn, to discover the answer to the client's query. When the local DNS server gets the answer (or decides that for some reason it cannot), it stores any information it received and answers the client. For example, to find the IP address for homer.odyssey.com, the local DNS server first asks a public root name server which machines are name servers for the corn domain. It then asks one of those "com" name servers which machines are name servers for the odyssey.com domain, and then it asks one of those name servers for the IP address of homer.odyssey.com.

This asking and answering is all transparent to the client. As far as the client is concerned, it has communicated only with the local server. It does not know or care that the local server may have contacted several other servers in the process of answering the original question.

Referring still to FIG. **1**, the firewall **105** is associated with a respective domain name server **115**. Each of the other firewalls **107**, **155**, **157** is also associated with a respective domain name server **117**, **165**, **167**. The domain name server may be a dedicated virtual host on the same physical machine as the firewall. Alternatively, the domain name server may be a separate machine. A domain name server is provided for each layer in the multi-layer network.

In operation, assume now that a client C on the accounting network **103** is to connect to a host D on the accounting network **153** on a repeated basis. The DNS tables of each of the firewalls may then be programmed so as to enable such a connection to be established transparently, without the user so much as being aware of any of the firewalls **105**, **107**, **155**, **157**—hence the term programmable transparency. Both forward and reverse table entries are made in the domain name servers. Within a domain name server **115**, for example, D (the name of the remote host, e.g., mach1.XYZcorp.com) might be mapped to a virtual host having a network address that concludes with the digits 1.1, and vice versa. Within the domain name server **117**, D might be mapped to 5.4, within the domain name server **167**, D might be mapped to 3.22, and within the domain name server **165**, D might be mapped to 4.5, where each of the foregoing addresses has been randomly chosen simply for purposes of illustration. Finally, within a conventional DNS server (not shown), D is mapped to the "real" network address (e.g, the IP address) of D, say, 55.2.

When client C tries to initiate a connection to host D using the name of D, DNS operates in the usual manner to propagate a name request to successive levels of the network until D is found. The DNS server for D returns the network address of D to a virtual host on the firewall **155**. The virtual host returns its network address to the virtual host on the firewall **157** from which it received the lookup request, and so on, until a virtual host on the firewall **105** returns its network address (instead of the network address of D) to the client C. This activity is all transparent to the user.

Note that at each network level, the virtual host handling a connection is indistinguishable to the preceding virtual (or real) host from D itself. Thus, to the client C, the virtual host 1.1 is D, to the virtual host 1.1, the virtual host 5.4 is D, etc. There is no limit to the number of network layers that may be traversed in this fashion, or any difference in operation as the number of network layers increases. This multi-layering capability allows two remote machines to communicate with the same ease as if the machines were on the same local area network, regardless of the degree of proximity or separation.

Programmable transparency is based upon what may be termed "envoys." Important differences exist between envoys as described herein and conventional proxies. Normally, a prior-art proxy would have to prompt the user to enter a destination. To enable such prompting to occur, different proxy code has conventionally been required for each protocol to be proxied. Using programmable transparency, the destination is provided to an envoy using DNS and/or DDNS as described more fully hereinafter. There is therefore no need to always prompt the user for a destination and no need for the user to always enter a destination (although a mode of operation may be provided in which the user is prompted for and does enter a destination). Instead of a collection of conventional protocol-specific proxies, a single generic envoy program may be used.

The foregoing discussion has focused on the programmable transparency aspects of the present firewall. Of course, a primary function of a firewall is to selectively allow and disallow communications. Hence, in the course of establishing a connection, each virtual host examines a configuration table to determine, based on the particulars of the requested connection—source, destination, protocol, time-of-day, port number, etc.—whether such a connection will be allowed or disallowed. The process by which connection requests may be scrutinized is described in greater detail in U.S. patent application Ser. No. 08/595,957 entitled FIREWALL SYSTEM FOR PROTECTING NETWORK ELEMENTS CONNECTED TO A PUBLIC NETWORK, filed Feb. 6, 1996 and incorporated herein by reference.

The firewall may have more than two network interfaces, each with its own set of virtual hosts. Referring to FIG. 2, for example, the two-sided firewall discussed previously in relation to FIG. 1 has been replaced by a three-sided firewall **205**. An accounting department network **203** and a general corporate network **209** are connected to the firewall **205** as previously described. Also connected to the firewall **205** is an engineering department network **202**. In general, a firewall may be N-sided, having N different network connections. For each network connection there may be multiple virtual hosts which operate in the manner described above.

Referring again to FIG. 1, configuration of the firewalls may be easily accomplished by providing on each firewall a special-purpose virtual host that runs "Configurator" software—software that provides a Web-based front-end for editing configuration files for the other virtual hosts on the firewall. The special-purpose virtual host (**116**, **118**, **166** and **168** in FIG. 1) is preferably configured so as to allow only a connection from a specified secure client. The Configurator software running on the special-purpose virtual host is HTML-based in order to provide an authorized system administrator a familiar "point-and-click" interface for configuring the virtual firewalls in as convenient a manner as possible using a standard Web browser. Since Web browsers are available for virtually every platform, there results a generic GUI interface that takes advantage of existing technology.

Referring more particularly to FIG. 3, there is shown a firewall **305** having a first set of virtual hosts **305***a*, a second set of virtual hosts **305***b*, and a DNS/DDNS module **315**. The virtual hosts do not require and preferably do not have access to the disk files of the underlying machine. Instead, virtual host processes are spawned from a daemon process that reads a master configuration file from disk once at start-up. The DNS/DDNS module and the special-purpose virtual host **317** do have access to disk files **316** of the underlying physical machine. The special-purpose virtual host **317**, shown in exploded view, runs an HTML-based Configurator module **319**. Access to the special-purpose virtual host is scrutinized in accordance with rules stored on disk within configuration files **321**. Typically, these rules will restrict access to a known secure host, will require at least username/password authentication and optionally more rigorous authentication. Once access is granted, the Configurator module will send to the authorized accessing host a first HTML page. From this page, the user may navigate through different HTML pages using a conventional Web browser and may submit information to the special-purpose virtual host. The special-purpose virtual host will then use this information to update the configuration files **321**.

As will be appreciated more fully from the description of FIG. 7 hereinafter, configuration is based on host names, not IP addresses. As a result, two mappings are required in order to handle a connection request. The requestor needs an IP address. To this end, a first mapping maps from the host name received in the connection request to the IP address of a virtual host. The virtual host, however, needs the host name of the host to be connected to. To this end, the second mapping maps back to the host name in order to read an appropriate configuration file or sub-file based on the host name. Thus, when a connection request is received for homer. odyssey.com, DNS/DDNS in effect says to the requestor "Use virtual host X.X.X.X," where X.X.X.X represents an IP address. Then, when the virtual host receives the request, it performs a reverse lookup using DNS/DDNS, whereupon DNS/DDNS in effect says "Virtual host X.X.X.X, use the configuration information for homer.odyssey.com."

Security may be further enhanced, both with respect to connections to the special-purpose virtual host for configuration purposes and also with respect to connections generally, by using out-of-band user authentication. Out-of-band authentication uses a channel, a device or any other communications method or medium which is different from that over which the inter-network communication is to take place to transmit or convey an access key. Hence, in the example of FIG. 1, the firewall **155**, upon receiving a connection request from a particular source, might send a message, including a key, to a pager **119** of the authorized user of the source client. The user might be requested to simply enter the key. In more sophisticated arrangements, the user may be required to enter the key into a special hardware token to generate a further key. To gain access, a hacker must therefore steal one or more devices (e.g, a pager used to receive the out-of-band transmissions, a hardware token, etc.). Furthermore, if a hacker attempts unauthorized access to a machine while the authorized user is in possession of the pager or other communications device, the user will be alerted by the device unexpectedly receiving a message and access key.

Other methods may be used to communicate out-of-band so as to deliver the required access key. For example, the firewall **155** might send a fax to the fax number of the user of the source machine. Alternatively, identifying information may be sent to the user across the network, after which the user may be required to dial an unpublished number and enter the identifying information in order to receive a voice message containing the required key.

In each of the foregoing methodologies, the key is connection-specific. That is, once the connection is closed or the attempt to establish a connection is abandoned, if a user again attempts to establish a connection, the key that previously applied or would have applied is no longer applicable.

The different virtual hosts may also be configured to perform channel processing of various sorts as traffic traverses different network segments. Channel processing may include encryption, decryption, compression, decompression, image or sound enhancement, content filtering, etc. Channel processing is the processing performed on data flowing through a communications channel to enhance some attribute of the data, such as security, reproduction quality, etc. In some instances, channel processing may actually affect the content of the data, for example "bleeping" obscenities by replacing them with a distinctive character string. Alternatively, channel processing may intervene to cause a connection to be closed if the content to be sent on that connection is found to be objectionable.

Channel processing may be performed using existing standard software modules. In the case of encryption and decryption, for example, modules for DES, RSA, Cylink, SET, SSL, and other types of encryption/decryption and authentication may be provided on the firewall. In the case of compression and decompression, standard modules may include MPEG, JPEG, LZ-based algorithms, etc. Based on information contained in the configuration files, information passing through the firewall may be processed using one or more such modules depending on the direction of data flow.

Channel processing may be used to perform protocol translation, for example between IP and some other protocol or protocols. One problem that has recently received attention is that of using IP for satellite uplink and downlink transmissions. The relatively long transit times involved in satellite transmissions can cause problems using IP. One possible solution is to perform protocol translation between IP and an existing protocol used for satellite transmissions. Such protocol translation could be performed transparently to the user using a firewall of the type described.

Channel processing may also be used to perform virus detection. Blanket virus detection across all platforms is a daunting task and may not be practical in most cases. A system administrator may, however, configure the system to perform specified virus checking for specified hosts.

Encryption and decryption are particularly important to realizing the potential of the Internet and network communications. In the example just described, on the network segment between firewall **105** and **107**, DES encryption might be used, in accordance with the configuration file on firewalls **105** and **107**. Across the Internet, between firewall **107** and firewall **155**, triple DES may be applied. On the network segment between firewall **155** and **157** RSA encryption may be used. Alternatively, encryption could be performed between firewalls **105** and **155** and also between **107** and **155** and also between **157** and **155**. Thus the firewall **157** may then decrypt the cumulative results of the foregoing multiple encryptions to produce clear text to be passed on to host D. Combining encryption capabilities with programmable transparency as described above allows for the creation of virtual private networks—networks in which two remote machines communicate securely through cyberspace in the same manner as if the machines were on the same local area network.

Using DDNS, mappings between a host machine and a virtual host are performed dynamically, on-the-fly, as required. Any of various algorithms may be used to select a virtual host to handle a connection request, including, for example, a least-recently-used strategy. A time-out period is established such that, if a connection has been closed and is not reopened within the time-out period, the virtual host that was servicing that connection may be re-mapped so as to service another connection—i.e., it becomes associated with a different node. In this manner, the number of clients that may be serviced is vastly increased. In particular, instead of the number of clients that may use a particular network interface being limited to the number of virtual hosts on that interface as would be the case using static DNS entries, using DDNS, any number of hosts may use a particular network interface subject to availability of a virtual host. Moreover, instead of making static DNS entries at each level of a multi-level network, using DDNS, such entries are rendered unnecessary.

DDNS allows for dynamic load sharing among different physical machines. Hence, instead of a single physical machine, one or more of the firewalls in FIG. **1** might be realized by two or more physical machines. When performing mapping, DDNS can take account of the load on the physical machine using conventional techniques. If one physical machine fails, the functions of that machine may still be performed by virtual hosts running on another physical machine. DDNS likewise allows a firewall to be scaled-up very easily, by adding one or more additional physical machines and configuring those machines as additional virtual hosts having identical configurations as on the existing physical machine or machines, but different network addresses.

Referring more particularly to FIG. **4**, a load-sharing firewall is realized using a first firewall **407** and a second firewall **408** connected in parallel to a network **420** such as the Internet. Redundancy is provided by conventional DNS procedures. That is, in DNS, redundant name servers are required by the DNS specification. If a query addressed to one of the redundant name servers does not receive a response, the same query may then be addressed to another name server. The same result holds true in FIG. **4**. If one of the physical firewall machines **407** or **408** is down, the other machine enables normal operation to continue.

The configuration of FIG. **4**, however, further allows the physical firewall machines **407** and **408** to share the aggregate processing load of current connections. Load sharing may be achieved in the following manner. Each of the DNS modules of all of the machines receive all DNS queries, because the machines are connected in parallel. Presumably, the DNS module of the machine that is least busy will be the first to respond to a query. An ensuing connection request is then mapped to a virtual host on the responding least-busy machine.

As the popularity and use of the Internet continues to grow, there is a concern that all available addresses will be used, thereby limiting further expansion. An important result of DDNS is that network addresses may be reused on network segments between which at least one firewall intervenes. More particularly, the addresses which are employed on opposite sides of a firewall are mutually exclusive of one another to avoid routing errors. Referring again to the example of FIG. **1**, users of the Internet **120** are unaware of the addresses employed on a network segment **110**. Certain addresses can be reserved for use behind a firewall. As shown in FIG. **1**, for example, the subset of addresses represented as 192.168.X.X can be used on the network segment **110**. So long as an address is not used on both sides of the same firewall, no routing errors will be introduced. Therefore, the same set of addresses can be used on the network segment **160**, which is separated from the Internet via the firewall **157**. On network segment **102** and network segment **152**, the entire address space may be used, less those addresses used on the segments **110**, **120** of the respective firewalls **105** and **155**. Thus by isolating Internet Service Providers (ISPs) from the Internet at large using firewalls of the type described, each ISP could enjoy use of almost the full address space of the Internet (232 addresses). Exhaustion of network addresses, presently a grave concern within the Internet community, is therefore made highly unlikely.

Address reuse may be further facilitated by providing multiple multi-homing firewall programs running on a single physical machine and defining a virtual network connection between the two firewall programs using an IP address within the range 192.168.X.X as described previously. To the user and to the outside world, this "compound firewall" appears as a single multi-homing firewall of the type previously described. However, since internally the firewall is really two firewalls, the entire Internet address space may be used on both sides of the firewall, except for the addresses 192.168.X.X. This configuration is illustrated in FIG. 9.

In essence, the use of firewalls as presently described allows the prevailing address model of network communications to be transformed from one in which IP addresses are used for end-to-end transport to one in which host names are used for end-to-end transport, with IP addresses being of only local significance. The current use of IP addresses for end-to-end transport may be referred to as address-based routing. Using address-based routing, address exhaustion becomes a real and pressing concern. The use of host names for end-to-end transport as presently described may be referred to as name-based routing. Using name-based routing, the problem of address exhaustion is eliminated.

The firewall as described also allows for envoys to handle connectionless (e.g, UDP—User Datagram Protocol) traffic, which has been problematic in the prior art. UDP is an example of a connectionless protocol in which packets are launched without any end-to-end handshaking. In the case of many prior-art firewalls, UDP traffic goes right through the firewall unimpeded. The present firewall handles connectionless traffic using envoys. Rules checking is performed on a first data packet to be sent from the first computer to the second computer. If the result of this rules checking is to allow the first packet to be sent, a time-out limit associated with communications between the first computer and the second computer via UDP is established, and the first packet is sent from one of the virtual hosts to the second computer on behalf of the first computer. Thereafter, for so long as the time-out limit has not expired, subsequent packets between the first computer and the second computer are checked and sent. A long-lived session is therefore created for UDP traffic. After the time-out limit has expired, the virtual host may be remapped to a different network address to handle a different connection.

The construction of a typical firewall in accordance with the present invention will now be described in greater detail. Referring to FIG. 5, the firewall is a software package that runs on a physical machine **500**. One example of a suitable machine is a super-minicomputer such as a SparcServer machine available from Sun Microsystems of Menlo Park, Calif. The firewall may, however, run on any of a wide variety of suitable platforms and operating systems. The present invention is not dependent upon a particular choice of platform and operating system.

Conventionally, the logical view of the firewall on the Internet, an intranet, or some other computer network is the same as the physical view of the underlying hardware. A single network address has been associated with a single network interface. As a result, no mechanism has existed for distinguishing between communications received on a single network interface and hence directing those communications to different logical machines.

As described previously, this limitation may be overcome by recognizing multiple addresses on a single network interface, mapping between respective addresses and respective virtual hosts, and directing communications to different addresses to different virtual hosts. Therefore, the present firewall, although it runs on a limited number of physical machines, such as a single computer **500**, appears on the network as a larger number of virtual hosts VH1 through VHn. Each virtual host has a separate configuration sub-file (sub-database) C1, C2, etc., that may be derived from a master configuration file, or database, **510**. The configuration sub-files are text files that may be used to enable or disable different functions for each virtual host, specify which connections and types of traffic will be allowed and which will be denied, etc. Because the configuration files are text files, they may be easily modified at any time following initial installation.

Preferably, each virtual host also has its own separate log file L1, L2, etc. This feature allows for more precise and more effective security monitoring.

The firewall is capable of servicing many simultaneous connections. The number of allowable simultaneous connections is configurable and may be limited to a predetermined number, or may be limited not by number but only by the load currently experienced by the physical machine. The number of maximum allowable connections or the maximum allowable machine load may be specified in the configuration file.

As described in greater detail in connection with FIG. 7, each configuration file C1, C2, etc., may have an access rules database **513**, including an Allow portion **515**, a Deny portion **517**, or both. Using the access rules database **513**, the firewall selectively allows and denies connections to implement a network security policy.

The firewall is self-daemoning, meaning that it is not subject to the limitations ordinarily imposed by the usual Internet meta-daemon, INETD, or other operating- system limitations. Referring to FIG. 6, when the firewall is brought up, it first reads in the master configuration file and then becomes a daemon and waits for connection requests. When a connection request is received, the firewall spawns a process, or execution thread, to create a virtual host VHn to handle that connection request. Each process runs off the same base code. However, each process will typically use its own sub-database from within the master configuration database to determine the configuration of that particular virtual host. Processes are created "on demand" as connection requests are received and terminate as service of those connection requests is completed.

An example of a portion of a master configuration file is shown in FIG. 7. Within the master configuration file database, different portions of the file form sub-databases for different virtual hosts. Each sub-database may specify a root directory for that particular virtual host. Also as part of the configuration file of each virtual host, an access rules database is provided governing access to and through the virtual host, i.e., which connections will be allowed and which connections will be denied. The syntax of the access rules database is such as to allow greater flexibility in specifying not only what machines are or are not to be allowed access, but also when such access is allowed to occur and which users are authorized. The access rules database may have an Allow portion, a Deny portion or both.

Processing with respect to the Allow database is performed prior to processing with respect to the Deny database. Therefore, if there is an entry for a the requested connection in the Allow database and no entry for that connection in the Deny database, then the connection will be allowed. If there is no Allow database and no entry in the Deny database, then the connection will also be allowed. If there is an entry for the requested connection in the Deny database, then the connection will be denied regardless. Machines may be specified by name or by IP address, and may include "wildcards," address masks, etc., for example: MisterPain.com, *.srmc.com, 192.168.0.*, 192.168.0.0/24, and so on.

Time restrictions may be included in either the Allow rules or the Deny rules. For example, access may be allowed from 1 am to 12 pm; alternatively, access may be denied from 12 pm to 1 am. Also, rules may be defined by identifiers, such as RULE1, RULE2, etc., and used elsewhere within the configuration sub-file of the virtual host to simplify and alleviate the need for replication.

All access rules must be satisfied in order to gain access to a virtual host. Depending on the virtual host, however, and as specified within the configuration sub-file, separate access scrutiny may be applied based on DNS entries. The accessing machine may be required to have a DNS (Domain Name Services) entry. Having a DNS entry lends at least some level of legitimacy to the accessing machine. Furthermore, the accessing machine may in addition be required to have a reverse DNS entry. Finally, it may be required that the forward DNS entry and the reverse DNS entry match each other, i.e., that an address mapped to from a given host name map back to the same host name.

If access is granted and a connection is opened, when the connection is later closed, a log entry is made recording information about that access. Log entries may also be made when a connection is opened, as data transport proceeds, etc.

Referring now to FIG. **8**, the logical structure of the present firewall is shown in greater detail. The main execution of the firewall is controlled by a daemon. In FIG. **8**, the daemon includes elements **801**, **803** and **805**. Although the daemon mode of operation is the default mode, the same code can also be run interactively under the conventional INETD daemon. Hence, when the firewall is first brought up, command-line processing is performed in block **801** to determine the mode of operation (daemon or interactive), which configuration file to read, etc. For purposes of the present discussion, the daemon mode of operation, which is the default, will be assumed.

In the daemon mode of operation, a process first reads the configuration file before becoming a daemon. By daemonizing after the configuration file (e.g., the master configuration file) has been read, the configuration file in effect becomes "hard coded" into the program such that the program no longer has to read it in. The daemon then waits to receive a connection request.

When a connection request is received, the daemon spawns a process to handle the connection request. This process then uses a piece of code referred to herein as an INET Wrapper **810** to check on the local side of the connection and the remote side of the connection to determine, in accordance with the appropriate Allow and Deny databases, whether the connection is to be allowed.

First the address and name (if possible) are obtained of the virtual host for which a connection is requested. Once the virtual host has been identified by name or at least by IP address, the master configuration database is scanned to see if a corresponding sub-database exists for that virtual host. If so, the sub-database is set as the configuration database of the virtual host so that the master configuration database need no longer be referred to. If no corresponding sub-database is found, then by default the master configuration database is used as the configuration database. There may be any number of virtual hosts, all independently configurable and all running on the same physical machine. The determination of which virtual host the process is to become is made in block **803**, under the heading of "multi-homing."

Once the process has determined which host it is, immediately thereafter, the process changes to a user profile in block **805** as defined in the configuration, so as to become an unprivileged user. This step of becoming an unprivileged user is a security measure that avoids various known security hazards. The INET Wrapper is then used to check on the remote host, i.e., the host requesting the connection. First, the configuration database is consulted to determine the level of access scrutiny that will be applied. (The default level of access scrutiny is that no DNS entry is required.) Then, the address and name (if possible) are obtained of the machine requesting the connection, and the appropriate level of access scrutiny is applied as determined from the configuration database.

If the remote host satisfies the required level of access scrutiny insofar as DNS entries are concerned, the INET Wrapper gets the Allow and Deny databases for the virtual host. First the Allow database is checked, and if there is an Allow database but the remote host is not found in it, the connection is denied. Then the Deny database is checked. If the remote host is found in the Deny database, then the connection is denied regardless of the allow database. All other rules must also be satisfied, regarding time of access, etc. If all the rules are satisfied, then the connection is allowed.

Once the connection has been allowed, the virtual host process invokes code **818** that performs protocol-based connection processing and, optionally, code **823** that performs channel processing (encryption, decryption, compression, decompression, etc.). When processing is completed, the connection is closed, if it has not already been closed implicitly.

It will be appreciated by those of ordinary skill in the art that the invention can be embodied in other specific forms without departing from the spirit or essential character thereof. The presently disclosed embodiments are therefore considered in all respects to be illustrative and not restrictive. The scope of the invention is indicated by the appended claims rather than the foregoing description, and all changes which come within the meaning and range of equivalents thereof are intended to be embraced therein.

What is claimed is:

1. In a computer networking environment having a plurality of firewall nodes on a path between a first terminal and a host terminal, where the firewall nodes delineate one network segment from another network segment, a method of establishing a communication link comprising the steps of:

providing a plurality of virtual hosts on each of the plurality of firewall nodes;

forming forward and reverse DNS tables for each of said plurality of firewall nodes wherein the DNS entries correspond to addresses of the virtual hosts on a given network segment and the virtual hosts correspond to actual hosts;

in response to the first terminal's DNS query to determine the address of the host, providing the address of the virtual host assigned to handle requests for the host terminal;

transmitting a connection request using the address of the virtual host;

at the virtual host assigned to handle requests for the host, and subsequently, at each successive virtual host located on firewall nodes on the path:

receiving a connection request;

obtaining a host name using reverse DNS, the host name corresponding to the requested address;

obtaining an address for use on the next network segment using DNS corresponding to the host name;

requesting a connection using the address for the next network segment;

receiving a connection request at the host and responding to the request; and,

transmitting the response in the reverse direction traversing the same path from virtual host to virtual host until the response reaches the first terminal.

2. The method of claim 1 wherein the virtual hosts of a given firewall node resides on more than one physical machine.

3. The method of claim 2 wherein the DNS service is dynamically updated depending upon the load associated with the physical machines.

4. The method of claim 1 wherein the virtual hosts perform channel processing.

5. The method of claim 1 wherein each virtual host has a set of configuration parameters.

6. The method of claim 1 wherein one of the virtual hosts on each firewall node is a configuration host allowing for the configuration of the firewall node.

* * * * *

US006006259A

United States Patent [19]

Adelman et al.

[11] **Patent Number: 6,006,259**

[45] **Date of Patent: Dec. 21, 1999**

[54] **METHOD AND APPARATUS FOR AN INTERNET PROTOCOL (IP) NETWORK CLUSTERING SYSTEM**

[75] Inventors: **Kenneth Allen Adelman**, Corralitos; **David Lyon Kashtan**, La Selva Beach; **William L. Palter**; **Derrell D. Piper, II**, both of Santa Cruz, all of Calif.

[73] Assignee: **Network Alchemy, Inc.**, Santa Cruz, Calif.

[21] Appl. No.: **09/196,941**

[22] Filed: **Nov. 20, 1998**

[51] **Int. Cl.**^{6} **G06F 13/00**

[52] **U.S. Cl.** **709/223**

[58] **Field of Search** 709/200, 201, 709/202, 203, 206, 208, 209, 217, 218, 219, 220, 221, 223, 224, 225

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,817,091	3/1989	Katzman et al.	714/8
4,860,201	8/1989	Stolfo et al.	712/11
5,083,265	1/1992	Valiant	712/21
5,301,337	4/1994	Wells et al.	709/104
5,553,239	9/1996	Heath et al.	713/201
5,612,865	3/1997	Dasgupta	700/79
5,666,486	9/1997	Alfieri et al.	709/217
5,699,500	12/1997	Dasgupta	714/1
5,708,659	1/1998	Rostoker et al.	370/392
5,712,981	1/1998	McKee et al.	709/241
5,805,785	9/1998	Dias et al.	714/4
5,822,531	10/1998	Gorczyca et al.	709/221
5,828,876	10/1998	Fish et al.	709/216
5,832,222	11/1998	Dziadosz et al.	709/216
5,848,233	12/1998	Radia et al.	713/201

OTHER PUBLICATIONS

Andresen, D., et al., "Toward a Scalable Distributed WWW Server on Workstation Clusters," Journal of Parallel and Distributed Computing, Article No. PC971305, vol. 42, 1997 at 91–100.

Cardellini, V., et al., "Efficient State Estimators for Load Control Policies in Scalable Web Server Clusters," 22nd Annual International Computer Software & Applications Conference (Compsac '98), Aug. 19–21, 1998 at 449–455.

Cardoza, W., et al., "Overview of Digital UNIX cluster System Architecture," Proceedings of COMPCON '96, Feb. 25–28, 1996 at 254–259.

(List continued on next page.)

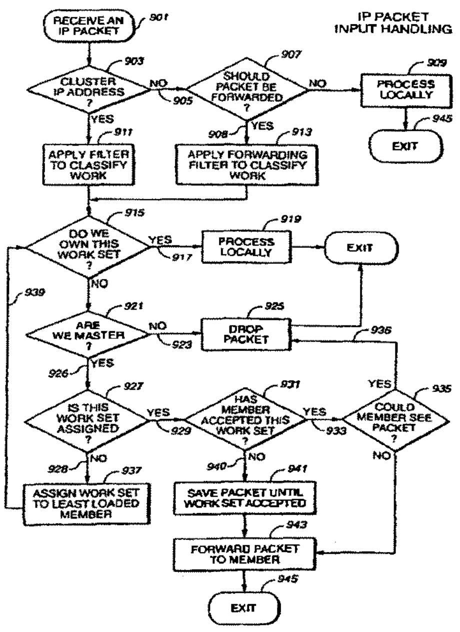
Primary Examiner—Moustafa M. Meky

Attorney, Agent, or Firm—Erwin J. Basinski; Morrison & Foerster LLP

[57] **ABSTRACT**

The present invention is an Internet Protocol (IP) network clustering system which can provide a highly scalable system which optimizes message throughput by adaptively load balancing its components, and which minimizes delay and packet loss especially in the TCP mode by a controlled fail-over process. An innovative process of defining the essential portion of TCP state required to maintain reliable message connections across the cluster is also disclosed.

16 Claims, 12 Drawing Sheets


IP PACKET INPUT HANDLING
RECEIVE AN IP PACKET
901
CLUSTER IP ADDRESS ?
903
NO
905
YES
SHOULD PACKET BE FORWARDED ?
907
NO
PROCESS LOCALLY
909
945
EXIT
YES
908
913
APPLY FILTER TO CLASSIFY WORK
911
APPLY FORWARDING FILTER TO CLASSIFY WORK
DO WE OWN THIS WORK SET ?
915
YES
917
PROCESS LOCALLY
919
EXIT
NO
939
ARE WE MASTER ?
921
NO
923
DROP PACKET
925
936
YES
926
IS THIS WORK SET ASSIGNED ?
927
YES
929
HAS MEMBER ACCEPTED THIS WORK SET ?
931
YES
933
COULD MEMBER SEE PACKET ?
935
YES
NO
NO
928
937
ASSIGN WORK SET TO LEAST LOADED MEMBER
NO
940
941
SAVE PACKET UNTIL WORK SET ACCEPTED
943
FORWARD PACKET TO MEMBER
945
EXIT

OTHER PUBLICATIONS

Chun, B.N., et al., "Virtual Network Transport Protocols for Myrinet," IEEE Micro, Jan./Feb. 1998 at 53–63.

Cisco Systems Inc., "LocalDirector Hot–Standby Faiilover: Product Overview," www.cisco.com/univercd/cc/td/doc/product/iaabu/localdir/ld20rns/ldicgd/ld3_ch5.htm, 1998.

Damani, O., et al., "ONE–IP: techniques for hosting a service on a cluster of machines," Computer Networks and ISDN Systems, vol. 29, 1997 at 1019–1027.

Fox, A., et al., "Cluster–Based Scalable Network Services," 16th ACM Symposium on Operating systems Principles, Oct. 5–8, 1997 at 78–91.

Ghormley, D.P., et al., "GLUnix: A Global Layer Unix for a Network of Workstations," Software–Practice and Experience, vol. 28, No. 9, Jul. 25, 1998 at 929–961.

Huang, Y., et al., "Software Implemented Fault Tolerance: Technologies and Experience," 23rd International Symposium on Fault–Tolerant Computing, Jun. 22–24, 1993 at 2–9.

Hunt, G.D.H., et al., "Network Dispatcher: a connection router for scalable Internet services," Computer Networks and ISDN Systems, vol. 30, 1998 at 347–357.

Hwang, K., et al., *Scalable Parallel Computing*, WCB McGraw–Hill, 1998 at 366–377, 453–564.

Kurcewicz, M., et al., "A Distributed WWW Cache," Computer Networks and ISDN Systems, vol. 30, 1998 at 2261–2267.

Mendiratta, V.B., "Reliability Analysis of Clustered Computing Systems," 9th International Symposium on Software Reliability Engineering, Nov. 4–7, 1998 at 268–272.

Parker, T., "QualixHA+," Unix Review, Mar. 1998 at 59–61.

Short, R., et al., "Windows NT Clusters for Availability and Scalabilty," Proceedings, IEEE COMPCON '97, Feb. 23–26, 1997 at 8–13.

Taschek, J., "ZD Internet Lab; A Well–Balanced Web," www.zdnet.com/icom/zdlabs/load.balance/, Feb. 23, 1998.

Thaler, D.G., et al., "Using Name–Based Mappings to Increase Hit Rates," IEEE/ACM Transactions on Networking, vol. 6, No. 1, Feb. 1998 at 1–14.

Valence Research, Inc., "Convoy Cluster Software; Product Overview," www.valence.com/convoy/main.html.

Venkataraman, S., et al., "Memory Management for Scalable Web Data Servers," 13th International Conference on Data Engineering, Apr. 7–11, 1997 at 510–519.

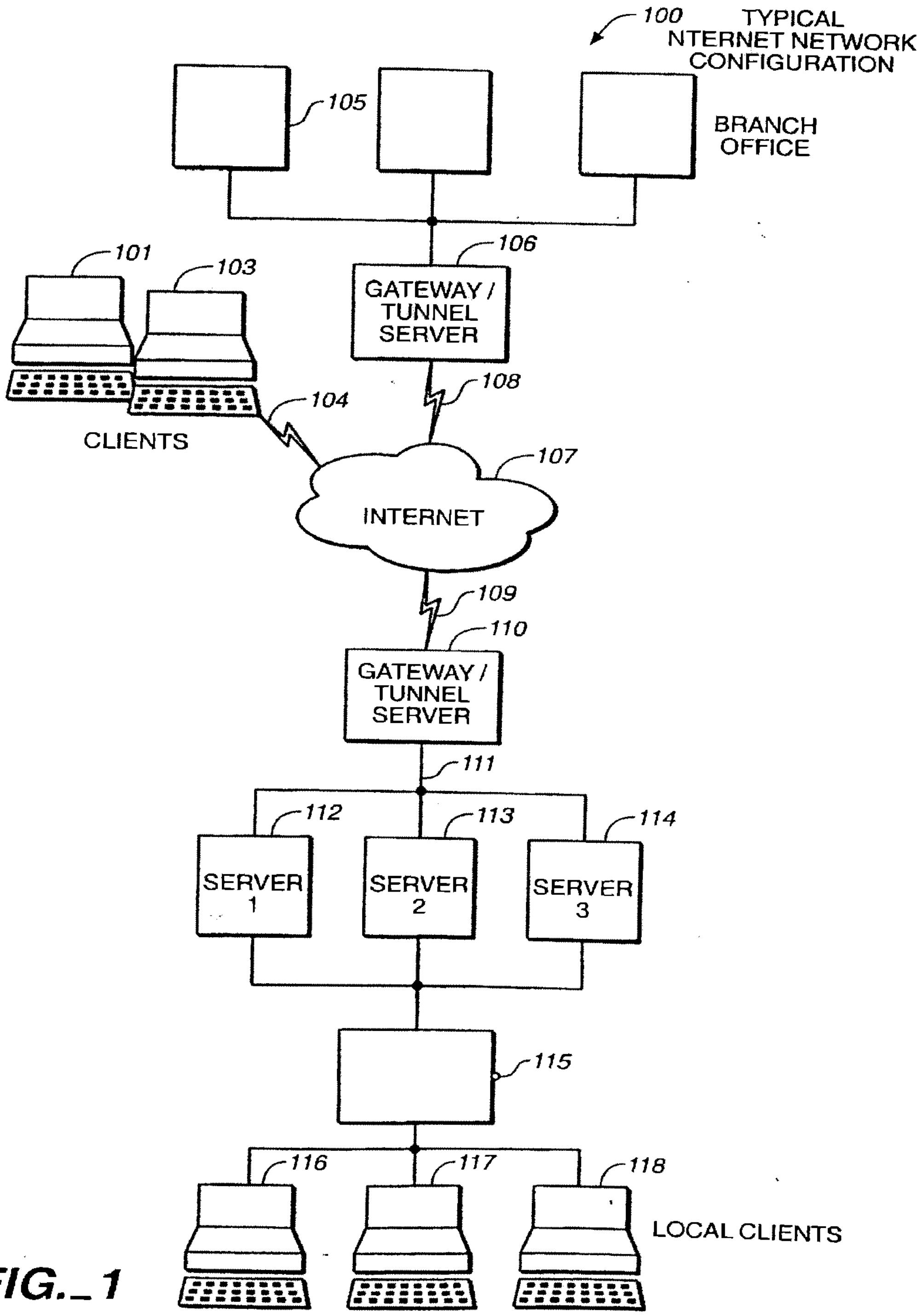

100 TYPICAL NTERNET NETWORK CONFIGURATION
105
BRANCH OFFICE
106
GATEWAY / TUNNEL SERVER
101
103
CLIENTS
104
108
107
INTERNET
109
110
GATEWAY / TUNNEL SERVER
111
112
113
114
SERVER 1
SERVER 2
SERVER 3
115
116
117
118
LOCAL CLIENTS

FIG._1

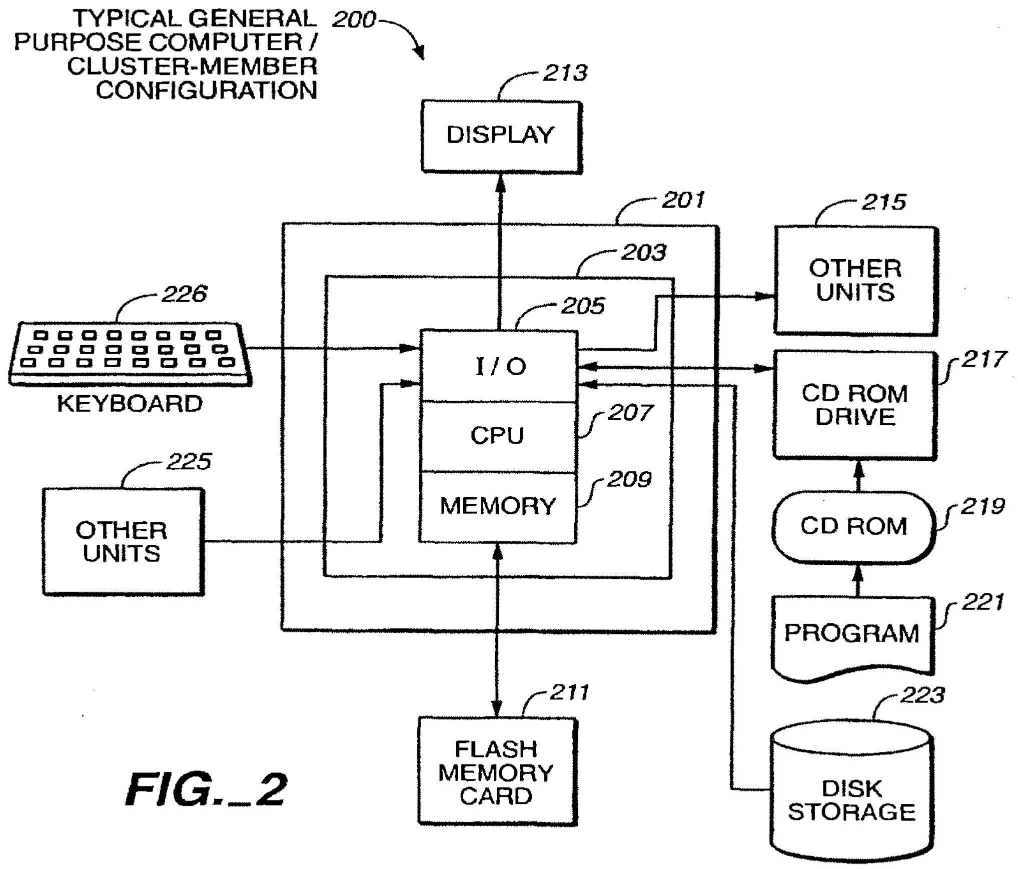

TYPICAL GENERAL PURPOSE COMPUTER / CLUSTER-MEMBER CONFIGURATION
200
213
DISPLAY
201
203
205
I / O
CPU
207
MEMORY
209
226
KEYBOARD
225
OTHER UNITS
215
OTHER UNITS
217
CD ROM DRIVE
219
CD ROM
221
PROGRAM
223
DISK STORAGE
211
FLASH MEMORY CARD

FIG._2

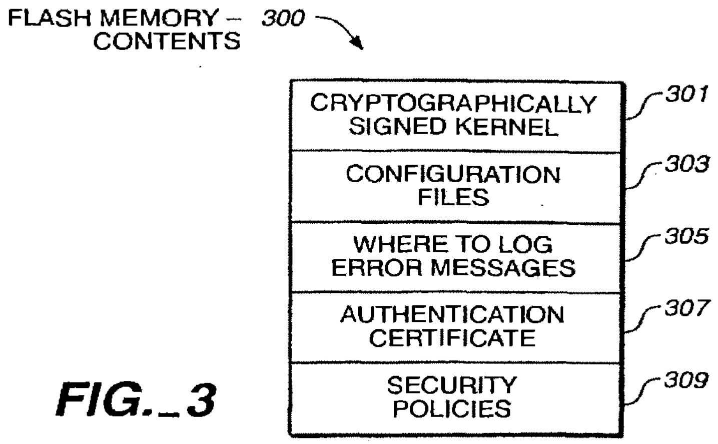

FLASH MEMORY - CONTENTS
300
CRYPTOGRAPHICALLY SIGNED KERNEL
301
CONFIGURATION FILES
303
WHERE TO LOG ERROR MESSAGES
305
AUTHENTICATION CERTIFICATE
307
SECURITY POLICIES
309

FIG._3

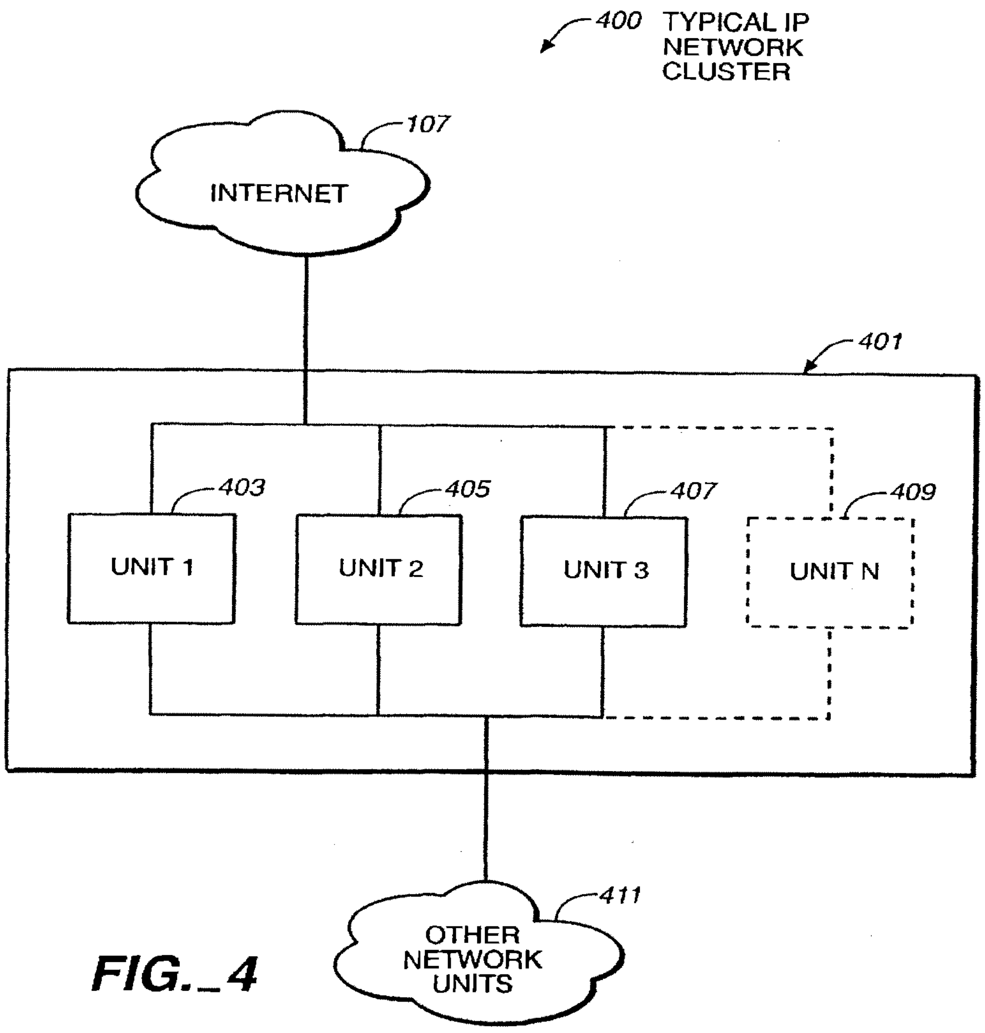

400 TYPICAL IP NETWORK CLUSTER
107
INTERNET
401
403
UNIT 1
405
UNIT 2
407
UNIT 3
409
UNIT N
411
OTHER NETWORK UNITS

FIG._4

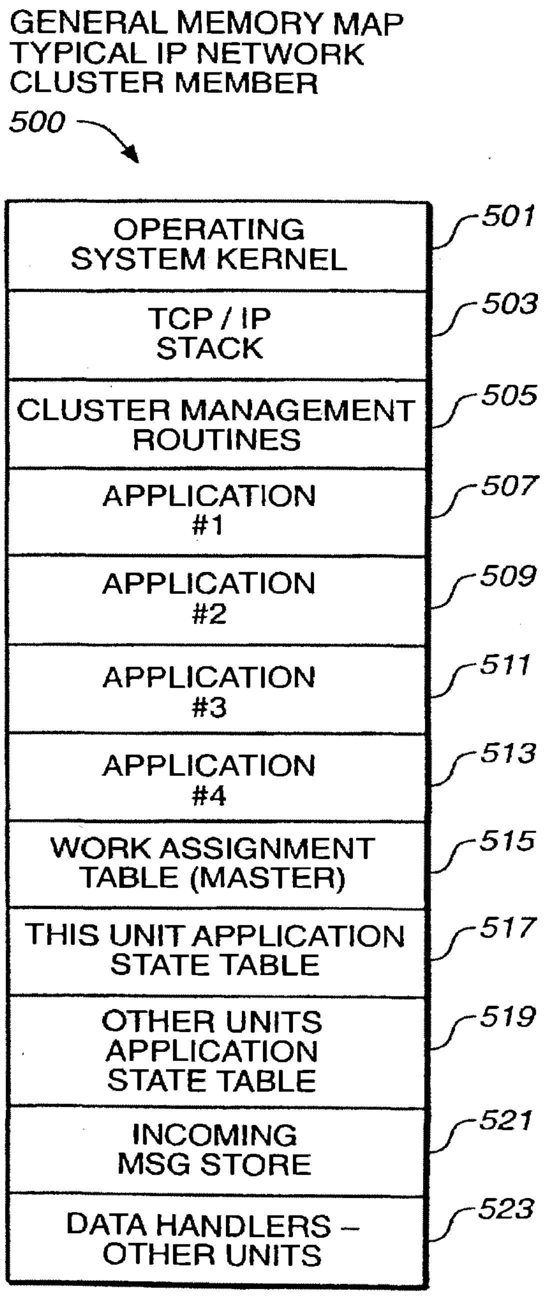

GENERAL MEMORY MAP
TYPICAL IP NETWORK
CLUSTER MEMBER
500
OPERATING SYSTEM KERNEL
501
TCP / IP STACK
503
CLUSTER MANAGEMENT ROUTINES
505
APPLICATION #1
507
APPLICATION #2
509
APPLICATION #3
511
APPLICATION #4
513
WORK ASSIGNMENT TABLE (MASTER)
515
THIS UNIT APPLICATION STATE TABLE
517
OTHER UNITS APPLICATION STATE TABLE
519
INCOMING MSG STORE
521
DATA HANDLERS – OTHER UNITS
523

FIG._5

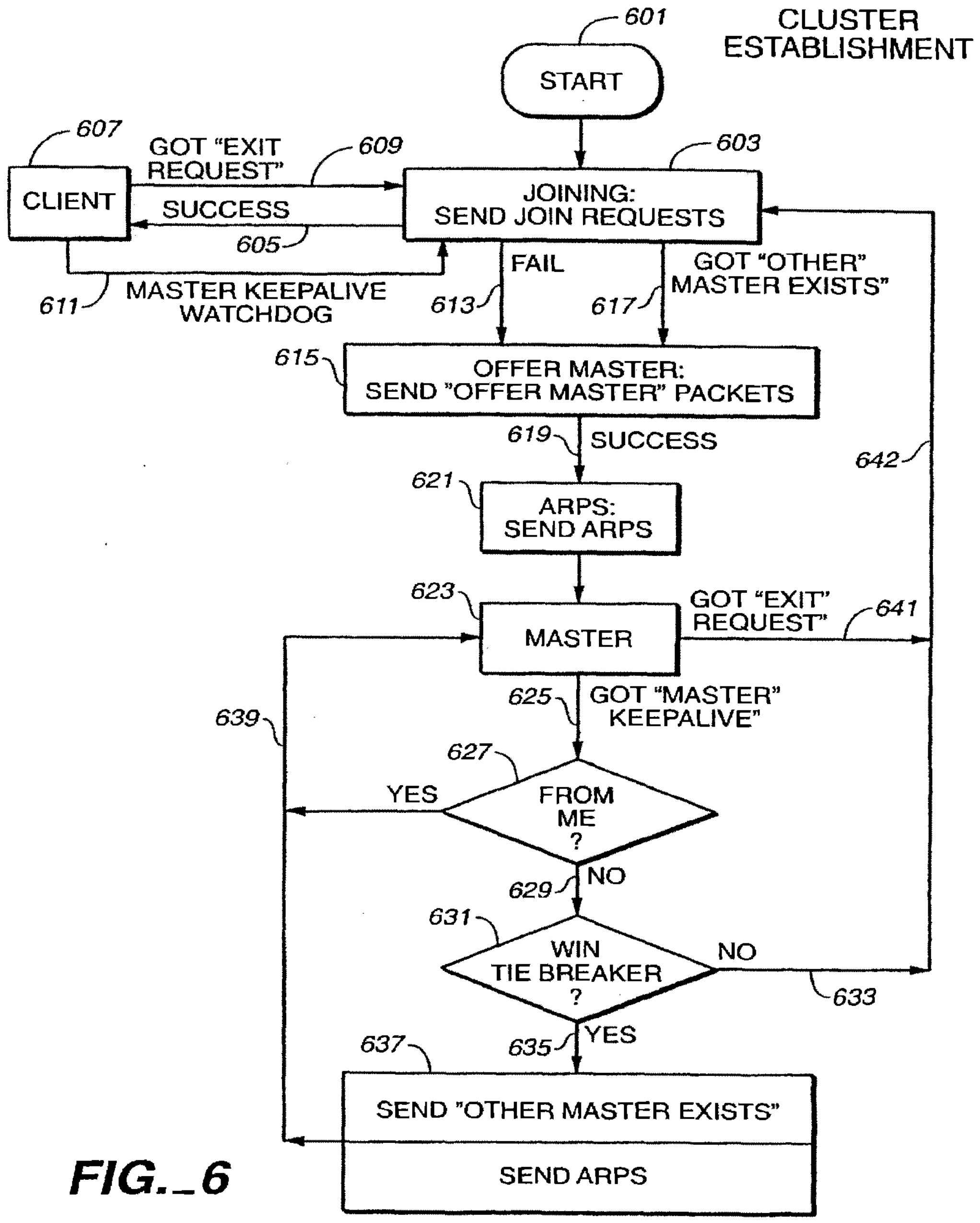

601
CLUSTER ESTABLISHMENT
START
607
CLIENT
GOT "EXIT REQUEST"
609
SUCCESS
605
603
JOINING: SEND JOIN REQUESTS
611
MASTER KEEPALIVE WATCHDOG
FAIL
613
GOT "OTHER" MASTER EXISTS"
617
615
OFFER MASTER: SEND "OFFER MASTER" PACKETS
619
SUCCESS
642
621
ARPS: SEND ARPS
623
MASTER
GOT "EXIT" REQUEST"
641
625
GOT "MASTER" KEEPALIVE"
639
627
YES
FROM ME ?
629
NO
631
WIN TIE BREAKER ?
NO
633
637
635
YES
SEND "OTHER MASTER EXISTS"
SEND ARPS

FIG._6

FIG._7

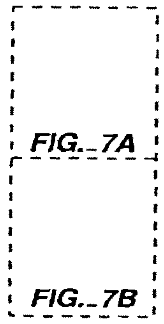


FIG._7A

TCP FAILOVER STATE 700

Initial State (Only need to send once) 702

Source IP Address + Port
Destination IP Address + Port
Maximum Segment Size
MSS + Options Size

Essential State (Send on each state change) 701

Flags: No Delay, No Options, Request Window Scaling,
Receive Window Scaling, Request Timestamp,
Receive Timestamp, Permit Selective ACK
Send "Next" Sequence Number
Window Update Segment Sequence Number
Window Update Segment Acknowledgement Number
Send Window
Receive "Next" Sequence Number
Receive "Advertized" Window
Send Window Scaling
Receive Window Scaling
Recent Timestamp Echo Data

703

Calculable State (Don't Send)

State = ESTABLISHED
Retransmit Time = None
Probe Time = Now
TCP Keepalive Time = Now
2MSL Time = None
Retransmit Time Shift = 0
Current Retransmit = Initial Value
Consecutive Duplicate Acks Received = 0
Force Output = 0;
Send "Unacknowledged" Sequence Number = Send "Next" Sequence Number
Send "Urgent Pointer" = Send "Unacknowledged" Sequence Number
Highest Sequence Number Sent = Send "Next" Sequence Number
Send Initial Segment Sequence Number = 0
Receive Window = Amount of space left in socket receive buffer
Receive "Urgent Pointer" = Receive "Next" Sequence Number
Receive Initial Segment Sequence Number = 0
Congestion Control Window = Initial Value
Congestion Control Window Linear/Exponential Threshold = Initial Value
Inactivity Time = 0
Estimated Round Trip Time = 0
Sequence Number Being Timed = 0
Smoothed Round Trip Time = Initial Value
Variance In Round Trip Time = Initial Value
Minimum Round Trip Time Allowed = Initial Value
Largest Window Offered by Peer = 0
Out Of Band Data = None
Send Pending Window Scaling = Send Window Scaling
Receive Pending Window Scaling = Receive Window Scaling
Timestamp Echo Data Update Time = 0
Last Ack Sent Sequence Number = Receive "Next" Sequence Number
Send Connection Count = 0
Receive Connection Count = 0;
Connection Duration = 0;
Number of Round Trip Time Samples = 0;
Number of TCP Keepalive Probes = Initial Value
Interval Between TCP Keepalive Probes = Initial Value
Time Before First TCP Keepalive Probe = Initial Value
Maximum Idle Time = Initial Value

FIG._7B

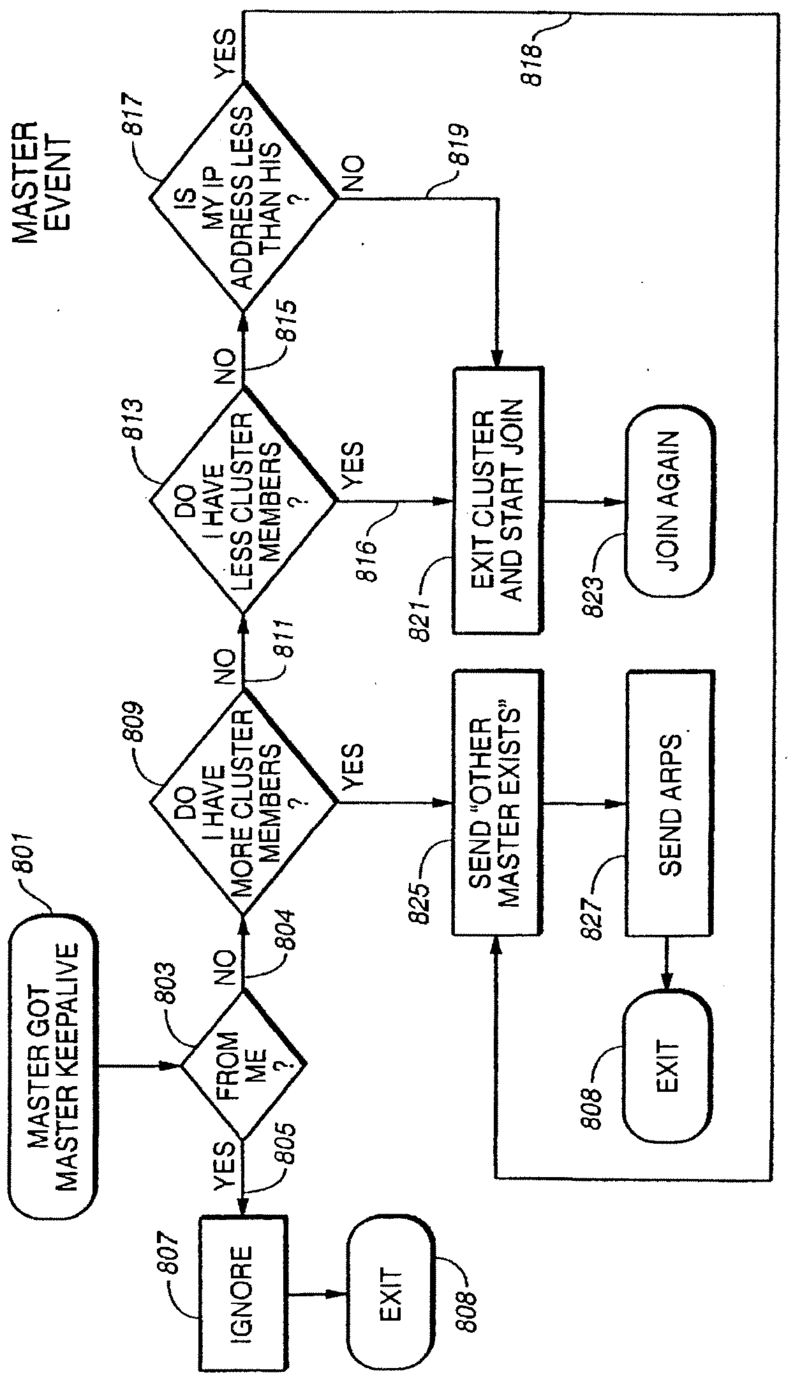

MASTER EVENT
MASTER GOT MASTER KEEPALIVE
801
FROM ME ?
803
YES
805
NO
804
IGNORE
807
EXIT
808
DO I HAVE MORE CLUSTER MEMBERS ?
809
YES
NO
811
DO I HAVE LESS CLUSTER MEMBERS ?
813
YES
816
NO
815
IS MY IP ADDRESS LESS THAN HIS ?
817
YES
818
NO
819
SEND "OTHER MASTER EXISTS"
825
SEND ARPS
827
EXIT
808
EXIT CLUSTER AND START JOIN
821
JOIN AGAIN
823

FIG._8A

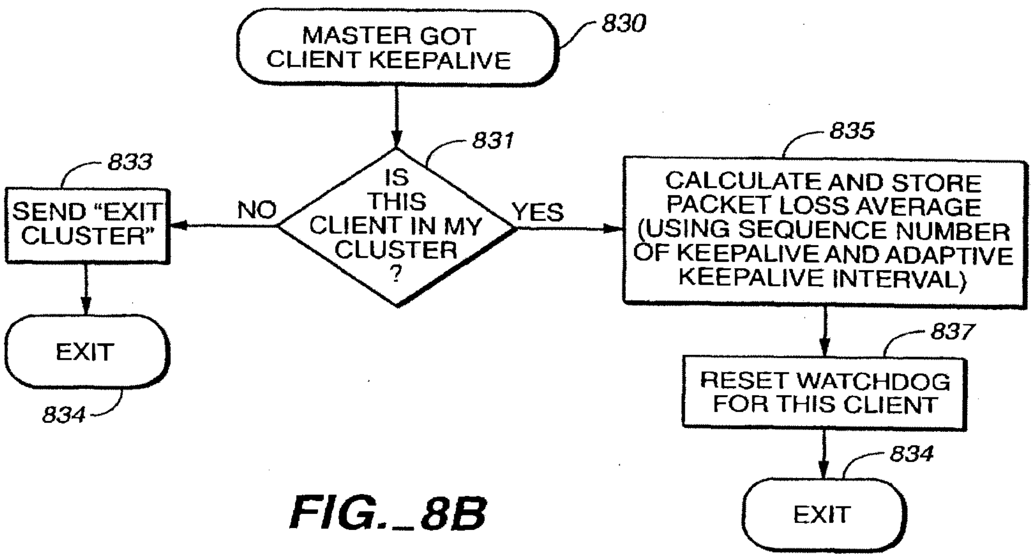

MASTER GOT CLIENT KEEPALIVE
830
IS THIS CLIENT IN MY CLUSTER ?
831
NO
YES
SEND "EXIT CLUSTER"
833
EXIT
834
CALCULATE AND STORE PACKET LOSS AVERAGE (USING SEQUENCE NUMBER OF KEEPALIVE AND ADAPTIVE KEEPALIVE INTERVAL)
835
RESET WATCHDOG FOR THIS CLIENT
837
EXIT
834

FIG._8B

MASTER EVENTS

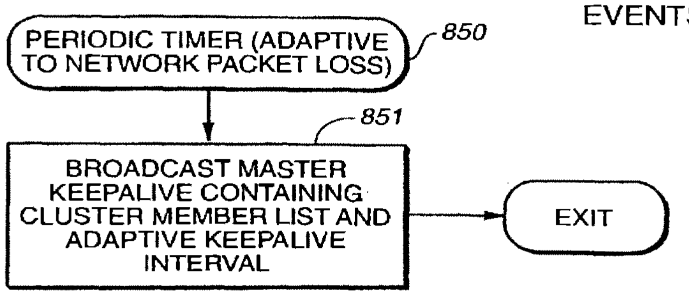

PERIODIC TIMER (ADAPTIVE TO NETWORK PACKET LOSS)
850
BROADCAST MASTER KEEPALIVE CONTAINING CLUSTER MEMBER LIST AND ADAPTIVE KEEPALIVE INTERVAL
851
EXIT

FIG._8C

FIG._8D

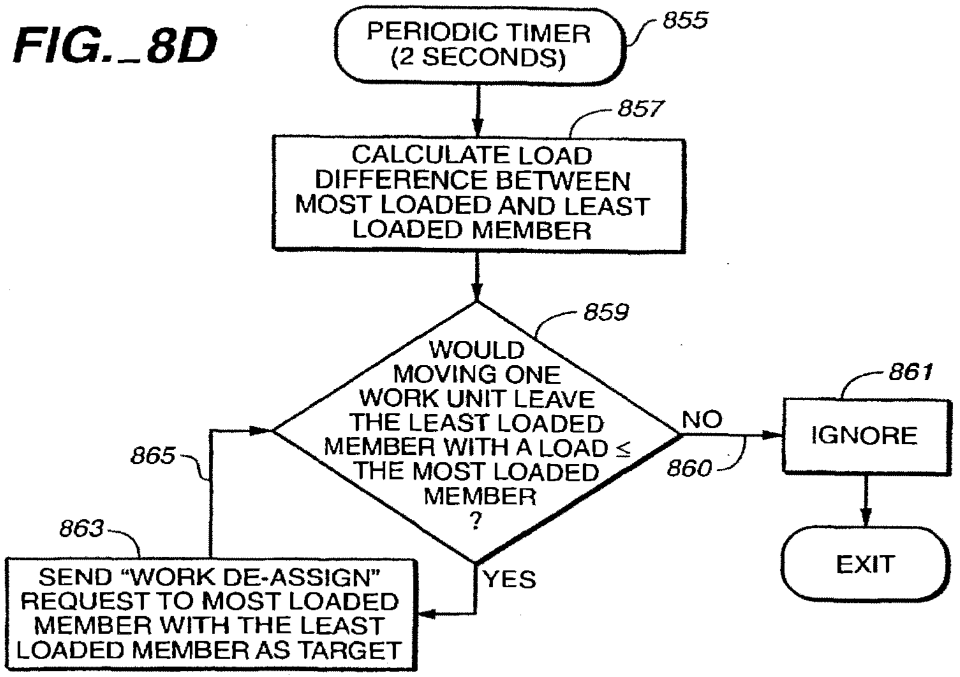

PERIODIC TIMER (2 SECONDS)
855
857
CALCULATE LOAD DIFFERENCE BETWEEN MOST LOADED AND LEAST LOADED MEMBER
859
WOULD MOVING ONE WORK UNIT LEAVE THE LEAST LOADED MEMBER WITH A LOAD ≤ THE MOST LOADED MEMBER ?
NO
860
861
IGNORE
EXIT
YES
865
863
SEND "WORK DE-ASSIGN" REQUEST TO MOST LOADED MEMBER WITH THE LEAST LOADED MEMBER AS TARGET

FIG._8E

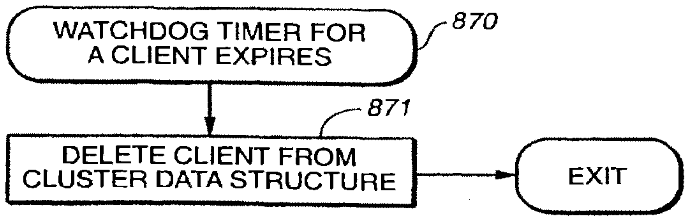
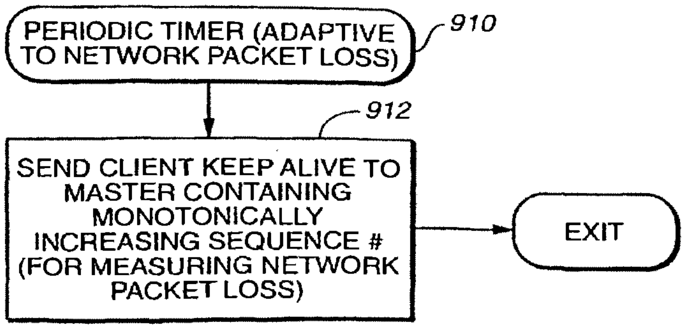

WATCHDOG TIMER FOR A CLIENT EXPIRES
870
871
DELETE CLIENT FROM CLUSTER DATA STRUCTURE
EXIT

FIG._8H


PERIODIC TIMER (ADAPTIVE TO NETWORK PACKET LOSS)
910
912
SEND CLIENT KEEP ALIVE TO MASTER CONTAINING MONOTONICALLY INCREASING SEQUENCE # (FOR MEASURING NETWORK PACKET LOSS)
EXIT

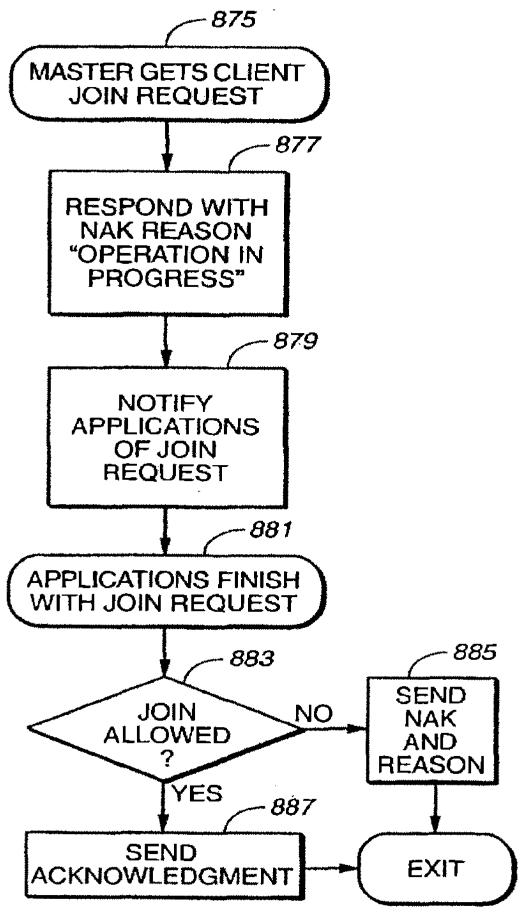
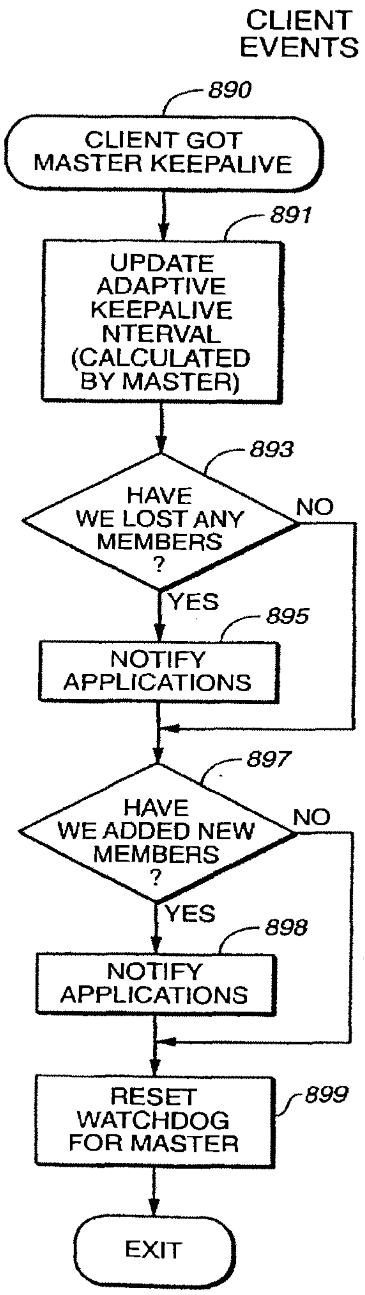

875
MASTER GETS CLIENT JOIN REQUEST
877
RESPOND WITH NAK REASON "OPERATION IN PROGRESS"
879
NOTIFY APPLICATIONS OF JOIN REQUEST
881
APPLICATIONS FINISH WITH JOIN REQUEST
883
JOIN ALLOWED ?
NO
885
SEND NAK AND REASON
YES
887
SEND ACKNOWLEDGMENT
EXIT

FIG._8F


CLIENT EVENTS
890
CLIENT GOT MASTER KEEPALIVE
891
UPDATE ADAPTIVE KEEPALIVE NTERVAL (CALCULATED BY MASTER)
893
HAVE WE LOST ANY MEMBERS ?
NO
YES
895
NOTIFY APPLICATIONS
897
HAVE WE ADDED NEW MEMBERS ?
NO
YES
898
NOTIFY APPLICATIONS
899
RESET WATCHDOG FOR MASTER
EXIT

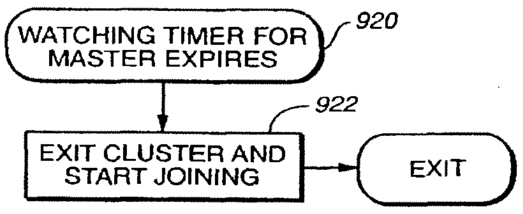

WATCHING TIMER FOR MASTER EXPIRES
920
922
EXIT CLUSTER AND START JOINING
EXIT

FIG._8I

FIG._8G

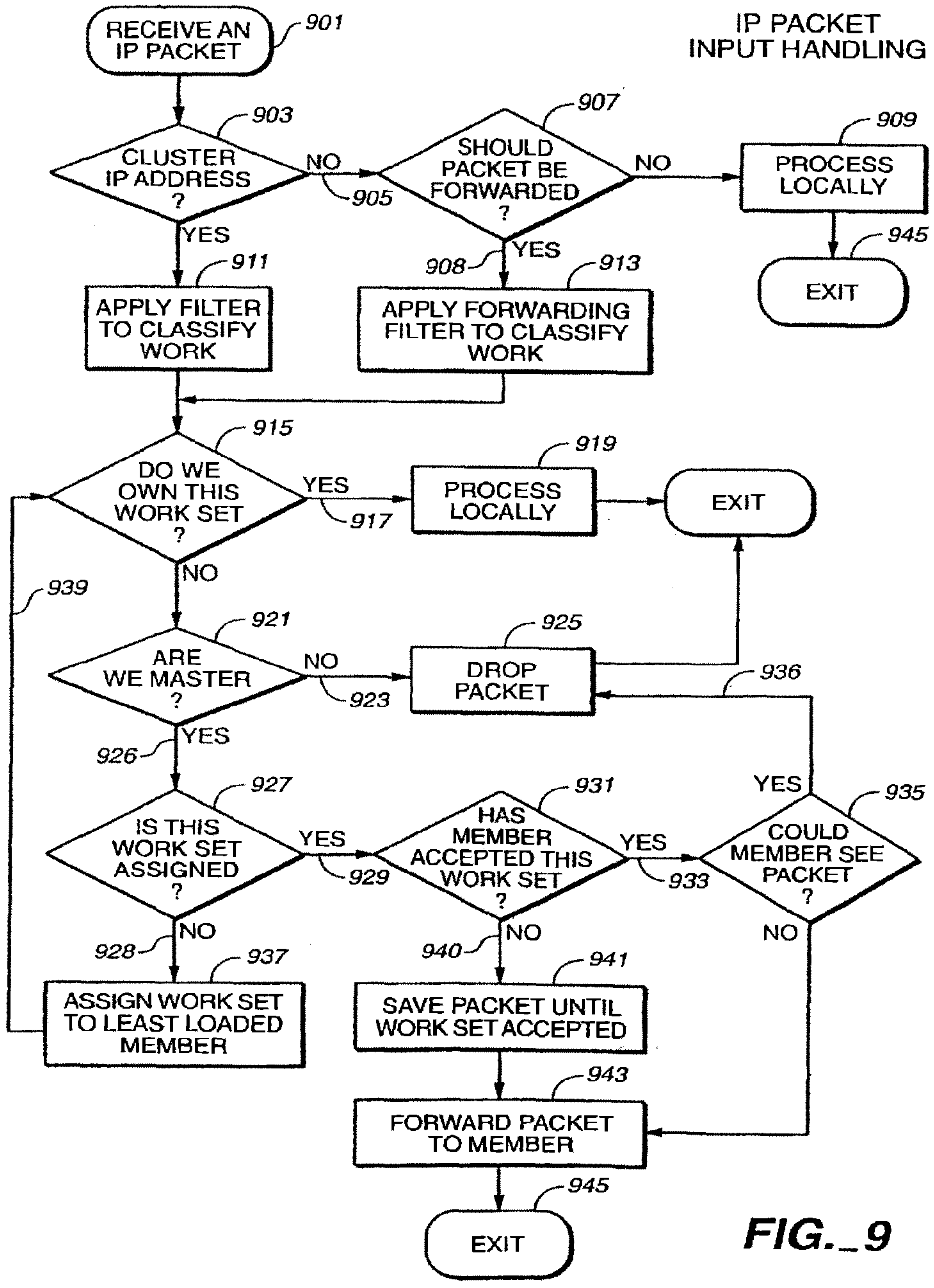

IP PACKET INPUT HANDLING
RECEIVE AN IP PACKET
901
CLUSTER IP ADDRESS ?
903
NO
905
SHOULD PACKET BE FORWARDED ?
907
NO
PROCESS LOCALLY
909
EXIT
945
YES
908
YES
APPLY FILTER TO CLASSIFY WORK
911
APPLY FORWARDING FILTER TO CLASSIFY WORK
913
DO WE OWN THIS WORK SET ?
915
YES
917
PROCESS LOCALLY
919
EXIT
NO
939
ARE WE MASTER ?
921
NO
923
DROP PACKET
925
936
YES
926
IS THIS WORK SET ASSIGNED ?
927
YES
929
HAS MEMBER ACCEPTED THIS WORK SET ?
931
YES
933
COULD MEMBER SEE PACKET ?
935
YES
NO
928
NO
ASSIGN WORK SET TO LEAST LOADED MEMBER
937
940
NO
SAVE PACKET UNTIL WORK SET ACCEPTED
941
FORWARD PACKET TO MEMBER
943
EXIT
945

FIG._9

METHOD AND APPARATUS FOR AN INTERNET PROTOCOL (IP) NETWORK CLUSTERING SYSTEM

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is related to application Ser. No. 09/197,018 entitled "Method and Apparatus for TCP/IP load balancing in an IP Network Clustering System," concurrently filed Nov. 20, 1998, and still pending.

TECHNICAL FIELD

This invention relates to the field of Computer Systems in the general Network Communications sector. More specifically, the invention is a method and apparatus for an Internet Protocol (IP) Network clustering system.

BACKGROUND ART

As more and more businesses develop electronic commerce applications using the Internet in order to market and to manage the ordering and delivery of their products, these businesses are searching for cost-effective Internet links that provide both security and high availability. Such mission-critical applications need to run all day, every day with the network components being highly reliable and easily scalable as the message traffic grows. National carriers and local Internet Service Providers (ISPs) are now offering Virtual Private Networks (VPN)—enhanced Internet-based backbones tying together corporate workgroups on far-flung Local Area Networks (LANs)—as the solution to these requirements.

A number of companies have recently announced current or proposed VPN products and/or systems which variously support IPSec, IKE (ISAKMP/Oakley) encryption-key management, as well as draft protocols for Point-to-Point Tunneling protocol (PPTP), and Layer 2 Tunneling protocol (L2TP) in order to provide secure traffic to users. Some of these products include IBM's Nways Multiprotocol Routing Services™2.2, Bay Networks Optivity™ and Centillion™ products, Ascend Communication's MultiVPN™ package, Digital Equipment's ADI VPN product family, and Indus River's RiverWorks™ VPN planned products. However, none of these products are known to offer capabilities which minimizes delay and session loss by a controlled fail-over process.

These VPNs place enormous demands on the enterprise network infrastructure. Single points of failure components such as gateways, firewalls, tunnel servers and other choke points that need to be made highly reliable and scaleable are being addressed with redundant equipment such as "hot standbys" and various types of clustering systems.

For example, CISCO™ Inc. now offers a new product called LocalDirector™ which functions as a front-end to a group of servers, dynamically load balances TCP traffic between servers to ensure timely access and response to requests. The LocalDirector provides the appearance, to end users, of a "virtual" server. For purposes of providing continuous access if the LocalDirector fails, users are required to purchase a redundant LocalDirector system which is directly attached to the primary unit, the redundant unit acting as a "hot" standby. The standby unit does no processing work itself until the master unit fails. The standby unit uses the failover IP address and the secondary Media Access Control (MAC) address (which are the same as the primary unit), thus no Address Resolution Protocol

(ARP) is required to switch to the standby unit. However, because the standby unit does not keep state information on each connection, all active connections are dropped and must be re-established by the clients. Moreover, because the "hot standby" does no concurrent processing it offers no processing load relief nor scaling ability.

Similarly, Valence™ Research Inc. (recently purchased by Microsoft® Corporation) offers a software product called Convoy Cluster™ (Convoy). Convoy installs as a standard Windows NT networking driver and runs on an existing LAN. It operates in a transparent manner to both server applications and TCP/IP clients. These clients can access the cluster as if it is a single computer by using one IP address. Convoy automatically balances the networking traffic between the clustered computers and can rebalance the load whenever a cluster member comes on-line or goes off-line. However this system appears to use a compute intensive and memory wasteful method for determining which message type is to be processed by which cluster member in that the message source port address and destination port address combination is used as an index key which must be stored and compared against the similar combination of each incoming message to determine which member is to process the message. Moreover, this system does not do failover.

There is a need in the art for an IP network cluster system which can easily scale to handle the exploding bandwidth requirements of users. There is a further need to maximize network availability, reliability and performance in terms of throughput, delay and packet loss by making the cluster overhead as efficient as possible, because more and more people are getting on the Internet and staying on it longer. A still further need exists to provide a reliable failover system for TCP based systems by efficiently saving the state information on all connections so as to minimize packet loss and the need for reconnections.

Computer cluster systems including "single-system-image" clusters are known in the art. See for example, "Scalable Parallel Computing" by Kai Hwang & Zhiwei Xu, McGraw-Hill, 1998, ISBN 0-07-031798-4, Chapters 9 & 10, Pages 453–564, which are hereby incorporated fully herein by reference. Various Commercial Cluster System products are described therein, including DEC's TruClusters™ system, IBM's SP™ system, Microsoft's Wolfpack™ system and The Berkeley NOW Project. None of these systems are known to provide efficient IP Network cluster capability along with combined scalability, load-balancing and controlled TCP fail-over.

SUMMARY OF THE INVENTION

The present invention overcomes the disadvantages of the above-described systems by providing an economical, high-performance, adaptable system and method for an IP Network cluster.

The present invention is an IP Network clustering system which can provide a highly scalable system which optimizes message throughput by adaptively load balancing its components, and which minimizes delay and packet loss especially in the TCP mode by a controlled fail-over process. No other known tunnel-server systems can provide this combined scalability, load-balancing and controlled fail-over.

The present invention includes a cluster apparatus comprising a plurality of cluster members, with all cluster members having the same internet machine name and IP address, and each member having a general purpose processor, a memory unit, a program in the memory unit, a display and an input/output unit; and the apparatus having a filter mechanism in each cluster member which uses a highly efficient hashing mechanism to generate an index number for each message session where the index number is used to determine whether a cluster member is to process a particular message or not. The index number is further used to designate which cluster member is responsible for processing the message and is further used to balance the processing load over all present cluster members.

The present invention further includes a method for operating a plurality of computers in an IP Network cluster which provides a single-system-image to network users, the method comprising steps to interconnect the cluster members, and assigning all cluster members the same internet machine name and IP address whereby all cluster members can receive all messages arriving at the cluster and all messages passed on by the members of the cluster appear to come from a single unit, and to allow them to communicate with each other; to adaptively designate which cluster member will act as a master unit in the cluster; and the method providing a filter mechanism in each cluster member which uses a highly efficient hashing mechanism to generate an index number for each message session where the index number is used to determine whether a cluster member is to process a particular message or not. The index number is further used to designate which cluster member is responsible for processing which message type and is further used to balance the processing load over all present cluster members.

Other embodiments of the present invention will become readily apparent to those skilled in these arts from the following detailed description, wherein is shown and described only the embodiments of the invention by way of illustration of the best mode known at this time for carrying out the invention. The invention is capable of other and different embodiments some of which may be described for illustrative purposes, and several of the details are capable of modification in various obvious respects, all without departing from the spirit and scope of the present invention.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the system and method of the present invention will be apparent from the following description in which:

FIG. 1 illustrates a typical Internet network configuration.

FIG. 2 illustrates a representative general purpose computer/cluster-member configuration.

FIG. 3 illustrates a representative memory map of data contained on a related Flash Memory card.

FIG. 4 illustrates a typical IP Network cluster

FIG. 5 illustrates a general memory map of the preferred embodiment of a cluster member acting as a tunnel-server.

FIG. 6 illustrates a flow-chart of the general operation of the cluster indicating the cluster establishment process.

FIG. 7 illustrates an exemplary TCP state data structure.

FIGS. 8A–8I illustrate flow-charts depicting the events which the master processes and the events which the non-master cluster members (clients) must process.

FIGS. 9 illustrates a flow-chart depicting the normal packet handling process after establishing the cluster.

BEST MODE FOR CARRYING OUT THE INVENTION

A method and apparatus for operating an Internet Protocol (IP) Network cluster is disclosed. In the following description for purposes of explanation, specific data and configurations are set forth in order to provide a thorough understanding of the present invention. In the presently preferred embodiment the IP Network cluster is described in terms of a VPN tunnel-server cluster. However, it will be apparent to one skilled in these arts that the present invention may be practiced without the specific details, in various applications such as a firewall cluster, a gateway or router cluster, etc. In other instances, well-known systems and protocols are shown and described in diagrammatical or block diagram form in order not to obscure the present invention unnecessarily.

Operating Environment

The environment in which the present invention is used encompasses the general distributed computing scene which includes generally local area networks with hubs, routers, gateways, tunnel-servers, applications servers, etc. connected to other clients and other networks via the Internet, wherein programs and data are made available by various members of the system for execution and access by other members of the system. Some of the elements of a typical internet network configuration are shown in FIG. 1, wherein a number of client machines **105** possibly in a branch office of an enterprise, are shown connected to a Gateway/hub/tunnel-server/etc. **106** which is itself connected to the internet **107** via some internet service provider (ISP) connection **108**. Also shown are other possible clients **101**, **103** similarly connected to the internet **107** via an ISP connection **104**, with these units communicating to possibly a home office via an ISP connection **109** to a gateway/tunnel-server **110** which is connected **111** to various enterprise application servers **112**, **113**, **114** which could be connected through another hub/router **115** to various local clients **116**, **117**, **118**.

The present IP Network cluster is made up of a number of general purpose computer units each of which includes generally the elements shown in FIG. 2, wherein the general purpose system **201** includes a motherboard **203** having thereon an input/output ("I/O") section **205**, one or more central processing units ("CPU") **207**, and a memory section **209** which may have a flash memory card **211** related to it. The I/O section **205** is connected to a keyboard **226**, other similar general purpose computer units **225**, **215**, a disk storage unit **223** and a CD-ROM drive unit **217**. The CD-ROM drive unit **217** can read a CD-ROM medium **219** which typically contains programs **221** and other data. Logic circuits or other components of these programmed computers will perform series of specifically identified operations dictated by computer programs as described more fully below.

Flash memory units typically contain additional data used for various purposes in such computer systems. In the preferred embodiment of the present invention, the flash memory card is used to contain certain unit "personality" information which is shown in FIG. 3. Generally the flash card used in the current embodiment contains the following type of information:

Cryptographically signed kernel—**(301)**

Configuration files (such as cluster name, specific unit IP address, cluster address, routing information configuration, etc.)—**(303)**

Pointer to error message logs—**(305)**

Authentication certificate—**(307)**.

Security policies (for example, encryption needed or not, etc.)—**(309)**

The Invention

The present invention is an Internet Protocol (IP) clustering system which can provide a highly scalable system which optimizes throughput by adaptively load balancing its components, and which minimizes delay and session loss by a controlled fail-over process. A typical IP cluster system of the preferred embodiment is shown in FIG. 4 wherein the internet **107** is shown connected to a typical IP cluster **401** which contains programmed general purpose computer units **403, 405, 407, 409** which act as protocol stack processors for message packets received. The IP cluster **401** is typically connected to application servers or other similar type units **411** in the network. In this figure it is shown that there purposes of further illustration the cluster will be depicted as having three units, understanding that the cluster of the present invention is not limited to only three units. Also for purposes of illustration the preferred embodiment will be described as a cluster whose applications may be VPN tunnel protocols however it should be understood that this cluster invention may be used as a cluster whose application is to act as a Firewall, or to act as a gateway, or to act as a security device, etc.

In the preferred embodiment of the present invention, each of the cluster members is a computer system having an Intel motherboard, two Intel Pentium™ processors, a 64 megabyte memory and two Intel Ethernet controllers, and two HiFn cryptographic processors. The functions performed by each processor are generally shown by reference to the general memory map of each processor as depicted in FIG. 5. Each cluster member has an Operating System kernel **501**, TCP/IP stack routines **503** and various cluster management routines (described in more detail below) **505**, program code for processing application #1 **507**, which in the preferred embodiment is code for processing the IPSec protocol, program code for processing application #2 **509**, which in the preferred embodiment is code for processing the PPTP protocol, program code for processing application #3 **511**, which in the preferred embodiment is code for processing the L2TP protocol, and program code for processing application #4 **513**, which in the preferred embodiment is code space for processing an additional protocol such as perhaps a "Mobile IP" protocol. Detailed information on these protocols can be found through the home page of the IETF at "http://www.ietf.org". The following specific protocol descriptions are hereby incorporated fully herein by reference:

"Point-to-Point Tunneling Protocol—PPTP", Glen Zorn, G. Pall, K. Hamzeh, W. Verthein, J. Taarud, W. Little, Jul. 28, 1998;

"Layer Two Tunneling Protocol", Allan Rubens, William Palter, T. Kolar, G. Pall, M. Littlewood, A. Valencia, K. Hamzeh, W. Verthein, J. Taarud, W. Mark Townsley, May 22, 1998;

Kent, S., Atkinson, R., "IP Authentication Header," draft-ietf-ipsec-auth-header-07.txt.

Kent, S., Atkinson, R., "Security Architecture for the Internet Protocol," draft-ietf-ipsec-arch-sec-07.txt.

Kent, S., Atkinson, R., "IP Encapsulating Security Payload (ESP)," draft-ietf-ipsec-esp-v2-06.txt.

Pereira, R., Adams, R., "The ESP CBC-Mode Cipher Algorithms," draft-ietf-ipsec-ciph-cbc-04.txt.

Glenn, R., Kent, S., "The NULL Encryption Algorithm and Its Use With IPsec," draft-ietf-ipsec-ciph-null-0.1.txt.

Madson, C., Doraswamy, N., "The ESP DES-CBC Cipher Algorithm With Explicit IV," draft-ietf-ipsec-ciph-des-expiv-02.txt.

Madson, C., Glenn, R., "The Use of HMAC-MD5 within ESP and AH," draft-ietf-ipsec-auth-hmac-md5-96-03.txt.

Madson, C., Glenn, R., "The Use of HMAC-SHA-1-96 within ESP and AH," draft-ietf-ipsec-auth-hmac-sha 196-03.txt.

Harkins, D., Carrel, D., "The Internet Key Exchange (IKE)," draft-ietf-ipsec-isakmp-oakley-08.txt.

Maughan, D., Schertler, M., Schmeider, M., and Turner, J., "Internet Security Association and Key Management Protocol (ISAKMP)," draft-ietf-ipsec-isakmp-10.{ps,txt}.

H. K. Orman, "The OAKLEY Key Determination Protocol," draft-ietf-ipsec-oakley-02.txt.

Piper, D. "The Internet IP Security Domain of Interpretation for ISAKMP," draft-ietf-ipsec-ipsec-doi-10.txt.

Tunneling protocols such as the Point-to-Point Tunneling Protocol (PPTP) and Layer 2 Tunneling Protocol (L2TP) although currently only "draft" standards, are expected to be confirmed as official standards by the Internet Engineering Task Force (IETF) in the very near future, and these protocols together with the Internet Security Protocol (IPSec), provide the basis for the required security of these VPNs.

Referring again to FIG. 5, the preferred embodiment in a cluster member also contains a work assignment table **515** which contains the message/session work-unit hash numbers and the cluster member id assigned to that work-unit; a table containing the application state table for this cluster member **517**; a similar application state table for the other members of the cluster **519**; an area for containing incoming messages **521**; and data handler routines for handling data messages from other members of the cluster **523**. Those skilled in the art will recognize that various other routines and message stores can be implemented in such a cluster member's memory to perform a variety of functions and applications.

The general operation of the preferred embodiment of the IP cluster is now described in terms of (1) cluster establishment (FIG. 6) including processes for members joining the cluster and leaving the cluster; (2) master units events processing (FIGS. 8A–8F) and client units events processing (FIGS. 8G–8I); and (3) finally, normal message processing activity (FIG. 9).

Referring now to FIG. 6 the cluster establishment activity is depicted. At system start-up **601** cluster members try to join the cluster by sending (broadcasting) a "join request" message **603**. This "join" message contains an authentication certificate obtained from a valid certificate authority. When the master unit receives this 'join" message it checks the certificate against a list of valid certificates which it holds and if it finds no match it simply tells him the join has failed. Note that normally when a system administrator plans to add a hardware unit to an existing cluster, he requests that his security department or an existing security certificate authority issue a certificate to the new unit and send a copy of the certificate to the master unit in the cluster. This process guarantees that someone could not illegally attach a unit to a cluster to obtain secured messages. If the master unit does match the certificate from the join message with a certificate it holds in its memory it sends an "OK to join" message. If a "OK to join" message is received **605** then this unit is designated a cluster member (client or non-master) **607**. Note that each cluster member has a master-watchdog timer (i.e. a routine to keep track of whether the member got a keepalive message from the master during a certain interval, say within the last 200 milliseconds) and if the timer expires (i.e. no keepalive message from the master during the interval) it will mean that the master unit is dead **607** and the cluster member/client will try to join the cluster again (**611**). Another event that will cause the cluster member/client **607** to try to join up again is if it gets an "exit request" message (i.e. telling it to "leave the cluster") **609** If the member sending out the join request message (**603**) does not get a "OK to join" message **613** the member sends out (broadcasts) packets offering to become the master unit **615**. If the member gets a "other master exists" message **617** the member tries to join again **603**. If after the member sends out the packets offering to become the master, he gets no response for 100 milliseconds **619** he sends broadcast Address Resolution Protocol (ARP) responses to tell anyone on the network what Ethernet address to use for the cluster IP address **621** and now acts as the cluster master unit **623**. If in this process the cluster member got no indication that another master exists (at **617**) and now thinking it is the only master **623** but yet gets a message to "exit the cluster" **641** the member must return to try to join up again **642**. This could happen for example, if this new master's configuration version was not correct. He would return, have an updated configuration and attempt to rejoin. Similarly, if this member who thinks he is the new master **623** gets a "master keepalive" message **625** (indicating that another cluster member thinks he is the master unit) then he checks to see if somehow the master keepalive message was from him **627** (normally the master doesn't get his own keepalive messages but it could happen) and if so he just ignores the message **639**. If however the master keepalive message was not from himself **629** it means there is another cluster member who thinks he is the master unit and somehow this "tie" must be resolved. (This tie breaker process is described in more detail below with respect to "Master event" processing). If the tie is resolved in favor of the new cluster member who thinks he is the master **635** he sends an "Other master exists" message to the other master and once again sends broadcast Address Resolution Protocol (ARP) responses to tell anyone on the network what Ethernet address to use for the cluster IP address **637** (because that other master could have done the same). If this new cluster member who thinks he is the master loses the tie-breaker **633** then he must go and join up again to try to get the cluster stabilized. This process produces a single cluster member acting as the master unit and the other cluster members understanding they are merely members.

Master Unit Events Processing

After a cluster has formed, there are various events that occur which the master unit must address. How these are handled in the preferred embodiment are now described with reference to FIGS. **8A**–**8F**. Referring to FIG. **8A** the first master unit event describes the "tie-breaker" process when two cluster members claim to be the "master" unit. Recalling from above that the master normally does not receive his own "keepalive" message so that if a master gets a "master keepalive" message **801** it likely indicates that another cluster member thinks he is the master. In the preferred embodiment, the "master keepalive" message contains the cluster member list, the adaptive keepalive interval (which is described in more detail below) and the current set of work assignments for each member which is used only for diagnostic purposes. So when a master gets a master keepalive message **801** he first asks "is it from me?" **803** and if so he just ignores this message **807** and exits **808**. If the master keepalive message is not from this master unit **804** then the "tie-breaker" process begins by asking "Do I have more cluster members than this other master?" **809** If this master does then he sends a "other master exists" message **825** telling the other master to relinquish the master role and rejoin the cluster. The remaining master then once again sends broadcast Address Resolution Protocol (ARP) responses to tell anyone on the network what Ethernet

address to use for the cluster IP address **827** and exits **808**. If the current master does not have more cluster members than this other master **811** he asks "do I have less cluster members than the other master?" **813** and if so **816** he must give up the master role to the other one by exiting the cluster **821** and rejoining the cluster as a member/non-master **823**) exiting to **601** in FIG. **6**. If the current master does not have less members than the other master **815** (which indicates they both have the same number) then the final tie-breaker occurs by asking "is my IP address less than his?" **817** and if so then again the current master wins the tie-breaker **818** and sends the "other master exists" message as before **825** If however he loses this final tie-breaker **819** then he exits the cluster to rejoin as a non-master member **821**.

Referring now to FIG. **8B** another master event occurs when the master gets a "client keepalive message" (that is one from a non-master cluster member) **830**. The master asks "is this client in my cluster?" **831** and if not the master sends the client an "exit cluster" message **833** telling the client to exit from this cluster. If the client is from this master's cluster the master calculates and stores a packet loss average value using the sequence number of the client keepalive message and the calculated adaptive keepalive interval. **835** The master then resets the watchdog timer for this client **837**. The watchdog timer routine is an operating system routine that checks a timer value periodically to see if the failover detection interval has elapsed since the value was last reset and if so the watchdog timer is said to have expired and the system then reacts as if the client in question has left the cluster and reassigns that clients work-load to the remaining cluster members.

As indicated above, the master periodically sends out a master keepalive message containing the cluster member list, the adaptive keepalive interval (which is described in more detail below) and the current set of work assignments for each member which is used only for diagnostic purposes. (See FIG. **8C**). In addition, the master periodically (in the preferred embodiment every 2 seconds) checks the load-balance of the cluster members. In FIG. **8D** when the timer expires **855** the master calculates the load difference between most loaded (say "K") and least loaded (say "J") cluster member **857** and then asks "would moving 1 work unit from most loaded (K) to least loaded (J) have any effect?" that is, if K>J is $K-1 \geqq J-1$? **859**. If so then the master sends a "work de-assign" request to the most loaded member with the least loaded member as the target recipient **863** and then the master checks the load numbers again **865**. If the result of moving 1 work unit would not leave the least loaded less than or equal to the most loaded **860** then the master makes no reassignments and exits **861**.

Another master event occurs when a watchdog timer for a client/cluster member expires wherein the master deletes that client from the cluster data list and the deleted unit's work goes into a pool of unassigned work to get reassigned normally as the next message arrives. (See FIG. **8E**).

Referring now to FIG. **8F** another master event in the preferred embodiment occurs when the master gets a client join request message **875**. The master initially tells the client to wait by sending a NAK with an "operation in progress" reason. **877** The master then notifies the applications that are present that a client is trying to join the cluster as some applications want to know about it. **879**. For example if IPSec is one of the applications then IPSec may want to validate this client before agreeing to let it join the cluster. If any application rejects the join request the master sends a NAK with the reason **855** and exits. If all applications approve the join request the master sends an ACK and the join proceeds as normal. **887**.

Client Cluster Member Events

The non-master cluster members (clients) must also send keepalive messages and monitor the watchdog timer for the master. Referring now to FIG. 8G when a client gets a master keepalive message 890 it updates its adaptive keepalive interval 891, and checks the list of cluster members to see if any members have been lost 893. If so this client notifies its applications that a cluster member has departed 895 (for example, IPSec wants to know). The client also checks to see if any members have been added to the cluster 897 and if so notifies the applications 898 and finally resets the watchdog timer for monitoring the master 899 and exits. Each client also has a periodic timer which is adaptive to the network packet loss value sent by the master which requires the client to send a client keepalive message (containing a monotonically increasing numeric value) to the master periodically (See FIG. 8H). Also each client has a master watchdog timer it must monitor and if it expires the client must exit the cluster and send a new join message to re-enter the cluster. (See FIG. 8I).

Normal IP Packet Processing

In order for a cluster member to correctly process only its share of the workload, one of three methods is used:

1. The MAC address of the master is bound to the cluster IP address (using the ARP protocol). The master applies the filtering function (described in more detail below) to classify the work and forward each packet (if necessary) to the appropriate cluster member.

2. A cluster-wide Unicast MAC address is bound to the cluster IP address (using the ARP protocol). Each cluster member programs its network interface to accept packets from this MAC destination address. Now each cluster member can see all packets with the cluster IP address destination. Each member applies the filtering function and discards packets that are not part of its workload.

3. method 2 is used but with a Multicast MAC address instead of a Unicast MAC address. This method is required when intelligent packet switching devices are part of the network. These devices learn which network ports are associated with each Unicast MAC address when they see packets with a Unicast MAC destination address, and they only send the packets to the port the switching device has determined is associated with that MAC address (only 1 port is associated with each Unicast MAC address). A Multicast MAC address will cause the packet switching device to deliver packets with the cluster IP destination address to all cluster members.

In the preferred embodiment, there is a mechanism for designating which cluster member is to process a message and allow the other members to disregard the message without inadvertently sending a "reset" message to the originating client. The preferred embodiment makes use of a "filter" process in each cluster member which calculates a hash function using certain fields of the incoming message header. This hash calculation serves as a means of both assigning a work unit number to a message and assigning a work unit to a particular cluster member for processing. This technique allows a cluster member to tell whether the incoming message must be processed by it, therefore the possibility of an inadvertent "reset" message is precluded. It is noted that other solutions to this problem of "how to get the work to the right member of the cluster with minimum overhead" could include a hardware filter device sitting between the network and the cluster wherein the hardware filter would do the member assignment and load balancing

function. Note that since all cluster members have the same MAC address, all cluster members get all messages and the way they tell whether they must process the message further is to calculate the work unit number using the hashing method shown above and then to check the resulting work unit number against their work load table to see if it is assigned to them. If not they dump the message from their memory. This is a fast and efficient scheme for dumping messages that the units need not process further and yet it provides an efficient basis for load-balancing and efficient fail-over handling when a cluster member fails.

The normal processing of IP packets is described with reference to FIG. 9. Upon the receipt of a packet 901 a determination is made as to whether the packet is addressed to a cluster IP address 903 or not. If not 905 then it is determined if the IP address is for this cluster member and if so it is processed by the IP stack locally 909. If the packet is to be forwarded (here the system is acting like a router) 908 a forward filter is applied in order to classify the work 913.

This designates whether the packet is for normal work for the cluster clients or is forwarding work. If at step 903 where the address was checked to see if it was a cluster IP address, the answer was yes then a similar work set filter is applied 911 wherein the IP source and destination addresses are hashed modulo 1024 to produce an index value which is used for various purposes. This index value calculation (the processing filter) is required in the current embodiment and is described more fully as follows;

Basically the fields containing the IP addresses, IP protocol, and TCP/UDP port numbers, and if the application is L2TP, the session and tunnel ID fields are all added together (logical XOR) and then shifted to produce a unique "work unit" number between 0 and 1023.

For example, in the preferred embodiment the index could be calculated as follows:

```
    /*
     * Sample Cluster Filtering function
    */
    static int Cluster_Filtering_Function(voil*Packet, int Forwarding)
    {
      struct ip *ip = (struct ip *)Packet;
      int i, length;
      /*
       * Select filtering scheme based on whether or not we are
forwarding this packet
      */
       if (Forwarding) {
           /*
            * Filter Forwarded packets on source & destination
IP address
           i = ip->ip_dst.s_addr;
           i ^=ip->ip_src.s_addr;
       } else {
           /*
            * Not forwarding: Put in the IP source address
           */
           i = ip->ip_src.s_addr;
           /*
            * Get the packet header length and dispatch on protocol
           */
           length = ip->ip_hl << 2;
           if (ip->ip_p==IPPROTO_UDP) {
              /*
               * UDP: Hash on UDP Source Port and Source IP
Address
              */
              i ^=((struct udphdr *)((char *)ip + length))->uh_sport;
           } else if (ip->ip_p==IPPROTO_TCP) {
              /*
```

-continued

```
                 * Hash on the TCP Source Port and Source IP Address
                 */
                i ^=((struct tcphdr *)((char *)ip + length))->th__sport;
            } else {
                /*
                 * Any other protocol: Hash on the Destination and
Source IP Addresses
                 */
                i ^=ip->ip__dst.s__addr;
            }
        }
        /*
         * Collapse it into a work-set number
         */
        return(IP__CLUSTER__HASH(i));
    }
```

Referring again to FIG. **9**, and having the work set index value calculated each member making this calculation uses the index value as an indirect pointer to determine for this work set if it is his assigned work set **915**, **917**. If the index value does not indicate that this work set has been assigned to this cluster member, if this cluster member is not the cluster master, then the packet is simply dropped by this cluster member **921**, **923**, **925**. If on the other hand this cluster member is the master unit **926** then the master must check to see if this work set has been assigned to one of the other cluster members for processing **927**. If it has been assigned to another cluster member **929** the master checks to see if that cluster member has acknowledged receiving the assignment **931** and if so the master checks to see if he was in the multicast mode or unicast/forwarding mode **933**, **935**. If he is in the unicast or multicast mode the master drops the packet because the assigned cluster member would have seen it **936**. If however, the master was in the forwarding mode the master will forward the packet to the assigned member for processing **943**. If the assigned cluster member has not acknowledged receiving the assignment yet **940** then save the packet until he does acknowledge the assignment **941** and then forward the packet to him to process **943**. If when the master checked to see if this work set had been assigned at **927** the answer is no **928** then the master will assign this work set to the least loaded member **937** and then resume its previous task **939** until the assigned member acknowledges receipt of the assignment as described above. If work is for this member, the packet is passed on to the local TCP/IP stack.

State Maintenance

RFC 1180 A TCP/IP Tutorial, T. Socolofsky and C. Kale, January 1991 generally describes the TCP/IP protocol suite and is incorporated fully herein by reference. In the present invention, a key element is the ability to separate the TCP state into an essential portion of the state and a calculable portion of the state. For example, the state of a TCP message changes constantly and accordingly it would not be practical for a cluster member to transfer all of this TCP state to all of the other members of the cluster each time the state changed. This would require an excessive amount of storage and processing time and would essentially double the traffic to the members of the cluster. The ability of the member units to maintain the state of these incoming messages is critical to their ability to handle the failure of a member unit without requiring a reset of the message session. FIG. 7 depicts the preferred embodiment's definition of which elements of the TCP state are considered essential and therefore must be transferred to each member of the cluster **701** when it changes, and which elements of the TCP state are considered to be calculable from the essential state **703** and therefore need not be transferred to all members of the cluster when it changes. The TCP Failover State **700** in the present embodiment actually comprises three portions, an Initial State portion **702** which only needs to be sent once to all cluster members; the Essential State Portion **701** which must be sent to all cluster members for them to store when any item listed in the Essential portion changes; and the Calculable State portion **703** which is not sent to all members. The data to the right of the equals sign ("=") for each element indicates how to calculate that elements value whenever it is needed to do so.

Failover Handling

As indicated above, the preferred embodiment of the IP cluster apparatus and method also includes the ability to monitor each cluster member's operation in order to manage the cluster operation for optimal performance. This means insuring that the cluster system recognize quickly when a cluster member becomes inoperative for any reason as well as have a reasonable process for refusing to declare a cluster member inoperative because of packet losses which are inherent in any TCP/IP network. This monitoring process is done in the preferred embodiment by a method whereby each non-member cluster member keeps a "master watchdog timer" and the master keeps a "client watchdog timer" for all cluster members. These watchdog timers are merely routines whereby the cluster member's OS periodically checks a "watchdog time-value" to see if it is more than "t" time earlier than the current time (that is, to see if the watchdog time value has been reset within the last "t" time). If the routine finds that the difference between the current time and the watchdog time value is greater than "t" time then it declares the cluster member related to the watchdog timer to be inoperative. These watchdog time values are reset whenever a cluster member sends a "keepalive" packet (sometimes called a "heartbeat" message) to the other members.

Generally a "keepalive" message is a message sent by one network device to inform another network device that the virtual circuit between the two is still active. In the preferred embodiment the master unit sends a "master keepalive" packet that contains a list of the cluster members, an "adaptive keepalive interval" and a current set of work assignments for all members. The non-master cluster members monitor a Master watchdog timer to make sure the master is still alive and use the "adaptive keepalive interval" value supplied by the master to determine how frequently they (the non-master cluster members) must send their "client keepalive" packets so that the master can monitor their presence in the cluster. The "client keepalive" packets contain a monotonically increasing sequence number which is used to measure packet loss in the system and to adjust the probability of packet loss value which is used to adjust the adaptive keepalive interval. Generally these calculations are done as follows in the preferred embodiment, however it will be understood by those skilled in these arts that various programming and logical circuit processes may be used to accomplish equivalent measures of packet loss and related watchdog timer values.

Each client includes a sequence number in its "client keepalive" packet. When the master gets this keepalive packet for client "x" he makes the following calculations:

$$S_\Delta=[\text{this sequence number}]-[\text{last sequence number}]-1$$

This value S_Δ is typically=0 or 1 and represents the number of dropped packets between the last two keepalive messages, or the current packet loss for client "x".

This value is then used in an exponential smoothing formula to calculate current average packet loss "P" as follows;

$$P_{new}=P_{old}X[127/128]+S_{n}X[1/128]$$

This P_{new} then represents the probability of a lost packet, and

P^n (P to the nth power) would represent the probability of getting "n" successive packet losses. And $1/P^n$ would be how often we would lose "n" packets in a row.

So "n" is defined as the number of lost packets per interval, and P^n then is the probability of losing "n" packets in an interval. Obviously if we lose more than some number of packets in a given interval the cluster member is either malfunctioning, inoperative or the network is having problems. In the preferred embodiment we assume "n" is a number between 2 and 20 and calculate its value adaptively as follows

We call the interval "K" and set $1/K=n\ P^n$. By policy we set K=3600 (which is equivalent to a period of 1 week) and then calculate the smallest integer value of "n" for which $n\ P^n. <1/3600$. In the preferred embodiment this is done by beginning the calculation with n=2 and increasing n by 1 iteratively until the condition is met. The resulting value of "n" is the adaptive keepalive interval which the master then sends to all of the cluster members to use in determining how often they are to send their "Client keepalive" messages.

Having described the invention in terms of a preferred embodiment, it will be recognized by those skilled in the art that various types of general purpose computer hardware may be substituted for the configuration described above to achieve an equivalent result. Similarly, it will be appreciated that arithmetic logic circuits are configured to perform each required means in the claims for processing internet security protocols and tunneling protocols; for permitting the master unit to adaptively distribute processing assignments for incoming messages and for permitting cluster members to recognize which messages are theirs to process; and for recognizing messages from other members in the cluster. It will be apparent to those skilled in the art that modifications and variations of the preferred embodiment are possible, which fall within the true spirit and scope of the invention as measured by the following claims.

What is claimed is:

1. An Internet Protocol (IP) Network cluster apparatus comprising:

a. a plurality of cluster members with all cluster members being addressable by a single dedicated Internet machine name and IP address for the cluster, each cluster member comprising a computer system having a processor, a memory, a program in said memory, a display screen and an input/output unit;

b. a filter mechanism in each cluster member, the filter mechanism using a hashing mechanism to generate an index number for each message session received by the cluster member, the index number being used to indicate to which workset a message belongs, worksets being assigned to cluster members to balance processing load, each cluster member checking whether the workset has been assigned to it in order to determine whether the cluster member must process the message received or ignore it.

2. The apparatus of claim 1 further comprising an assignment mechanism in each cluster member, for use by a cluster member designated as a master unit, the assignment mechanism used when a message of an unassigned message session is received by the master unit, the assignment mechanism using the index number calculated by the filter mechanism to assign sets of message sessions to cluster

members for further processing in order to load balance processing of incoming messages.

3. The apparatus of claim 1 further comprising a first program code mechanism in each of the plurality of cluster members configured to save state for each message session including TCP state.

4. The apparatus of claim 3 further comprising a second program code mechanism in each of the plurality of cluster members configured to transfer an essential portion of the saved state for each message session to each of the other cluster members, whenever required.

5. The apparatus of claim 4 further comprising a third program code mechanism in each of the plurality of cluster members configured to permit a cluster member acting as a master unit to recognize an equipment failure in one of the other members in the cluster, to reassign the work of the failed cluster member to remaining members in the cluster thereby rebalancing the processing load and maintaining the message sessions.

6. The apparatus of claim 5 further comprising a fourth program code mechanism in each of the plurality of cluster members configured to permit units which are not acting as the master unit to recognize an equipment failure in the master unit, to immediately and cooperatively designate one of the remaining cluster members as a new master unit, the new master unit to reassign the work of the failed cluster member to remaining cluster members thereby rebalancing the processing load and maintaining the message sessions.

7. The apparatus of claim 1 wherein the memory of each of the cluster members includes a flash memory card containing a program code mechanism which describes the personality of the cluster member including its cluster address.

8. A method for operating a plurality of computers in an Internet Protocol (IP) Network cluster, the cluster providing a single-system-image to network users, the method comprising the steps of;

a. providing a plurality of cluster members, each cluster member comprising a computer system having a processor, a memory, a program in said memory, a display screen and an input/output unit;

b. interconnecting the cluster members together, and assigning all cluster members a same internet machine name and a same IP address whereby a message arriving at the cluster will be recognized by the appropriate member in the cluster and an output from any cluster member will be recognized as coming from the cluster, and whereby the cluster members can communicate with each other; and

c. providing a filter mechanism in each cluster member, the filter mechanism using a hashing mechanism to generate an index number for each message session received by the cluster member, the index number being used to indicate to which workset a message belongs, worksets being assigned to cluster members to balance processing load, each cluster member checking whether the workset has been assigned to it in order to determine whether the cluster member must process the message received or ignore it.

9. The method of claim 8 further comprising an assignment mechanism in each cluster member, for use by a cluster member designated as a master unit, the assignment mechanism used when a message of an unassigned message session is received by the master unit, the assignment mechanism using the index number calculated by the filter mechanism to assign sets of message sessions to cluster members for further processing in order to load balance processing of incoming messages.

10. The method of claim 8 comprising the additional step of each cluster member saving state for each message session connection including TCP state, and for segregating this state into an essential state portion and a non-essential state portion.

11. The method of claim 10 comprising the additional step of each cluster member transferring to each other cluster member the saved essential state portion for message sessions for which that cluster member is responsible, such transfer to be made whenever the essential portion of the state changes, whereby all cluster members maintain essential state for all message session connections.

12. The method of claim 11 comprising the additional step of each cluster member recognizing the equipment failure of one of the cluster members, immediately reassigning a task of being the master if it is the master unit that failed, the master unit reassigning the work which was assigned to the failed cluster member, rebalancing the load on the remaining tunnel-servers.

13. An Internet Protocol (IP) network cluster apparatus comprising:

a. a plurality of interconnected cluster members, each cluster member comprising a computer system having a processor, a memory, a program in said memory, a display screen and an input/output unit;

b. means in each of the plurality of cluster members for recognizing other members of the plurality of cluster members which are connected together and cooperating with the other members to adaptively designate a master unit; and

c. means for generating an index number for each message session received by a cluster member, the index number being used to indicate whether the cluster member must process the message received or ignore it.

14. The apparatus of claim 13 further comprising means in each of the plurality of cluster members for saving essential state for each message session.

15. The apparatus in claim 14 further comprising means in each of the plurality of cluster members for periodically transferring the saved essential state for each message session to each of the other members in the cluster.

16. The apparatus of claim 15 further comprising means in each of the plurality of cluster members for permitting a cluster member acting as a master unit to recognize an equipment failure in one of the other cluster members, and for reassigning work of the failed cluster member to remaining members in the cluster thereby rebalancing the processing load and maintaining message session connections, and for permitting cluster members which are not acting as a master unit to recognize an equipment failure in the master unit, to immediately and cooperatively designate one of the remaining cluster members as a new master unit, the new master unit to reassign work of the failed cluster member to remaining members in the cluster thereby rebalancing the processing load and maintaining message session connections.

* * * * *

US005905859A

United States Patent [19]

Holloway et al.

[11] **Patent Number: 5,905,859**

[45] **Date of Patent: May 18, 1999**

[54] **MANAGED NETWORK DEVICE SECURITY METHOD AND APPARATUS**

[75] Inventors: **Malcolm H. Holloway**, Durham; **Thomas Joseph Prorock**, Raleigh, both of N.C.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **08/775,536**

[22] Filed: **Jan. 9, 1997**

[51] **Int. Cl.**^{6} **G06F 11/00**

[52] **U.S. Cl.** **395/187.01**

[58] **Field of Search** 395/187.01, 186, 395/185.09, 200.53, 200.54, 200.59, 200.55; 380/3, 25

[56] **References Cited**

U.S. PATENT DOCUMENTS

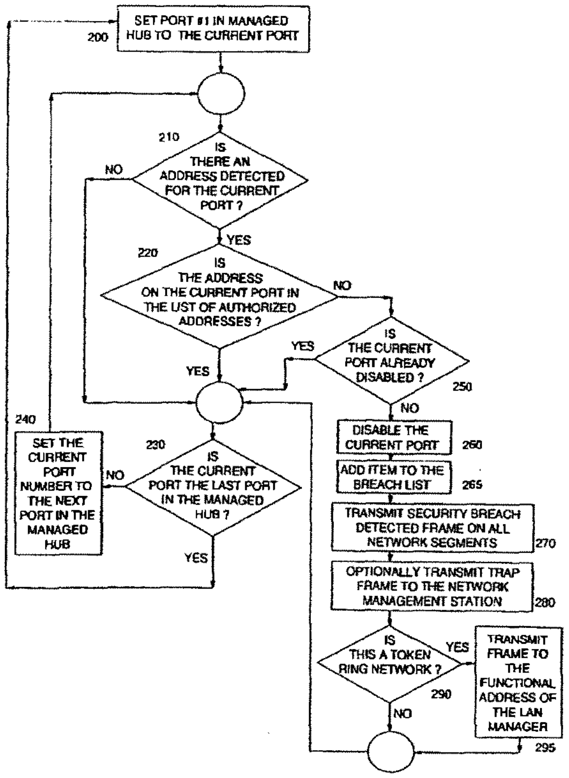
4,930,159	5/1990	Kravitz et al.	380/23
5,177,788	1/1993	Schanning et al.	380/23
5,305,385	4/1994	Schanning et al.	380/49
5,311,593	5/1994	Carmi	380/23
5,337,309	8/1994	Faulk, Jr.	370/60
5,414,833	5/1995	Hershey et al.	395/575
5,421,024	5/1995	Faulk, Jr. et al.	395/800
5,440,723	8/1995	Arnold et al.	395/183.14
5,495,580	2/1996	Osman	395/187.01
5,537,099	7/1996	Liang	340/825.07
5,606,668	2/1997	Shwed	395/187.01
5,615,340	3/1997	Dai et al.	395/187.01
5,727,146	3/1998	Savoldi et al.	395/187.01

Primary Examiner—Robert W. Beausoliel, Jr.
Assistant Examiner—Scott T. Baderman
Attorney, Agent, or Firm—John J. Timar

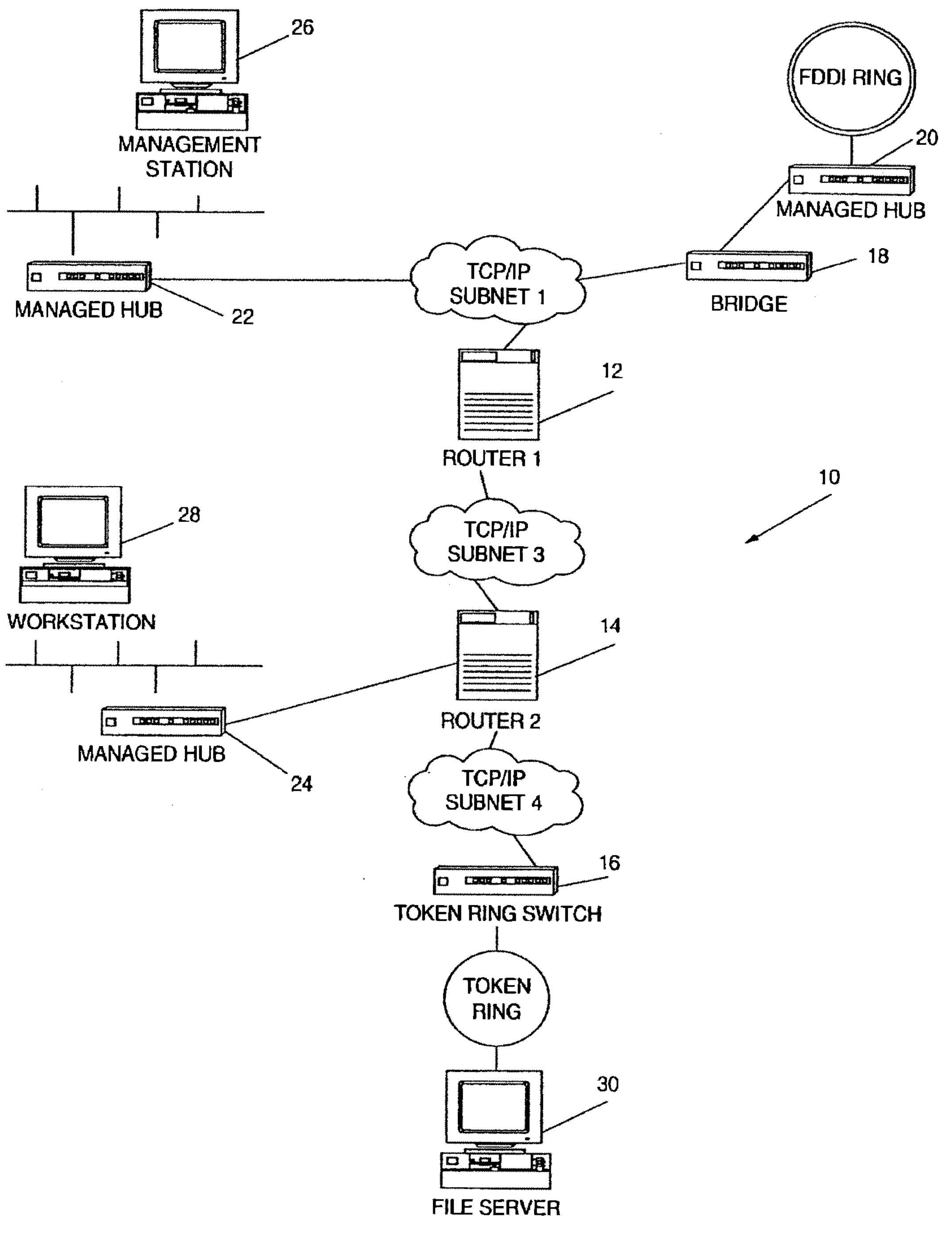
[57] **ABSTRACT**

An apparatus and method for providing security against intrusion in the managed devices of a campus LAN network is provided. A managed hub discovers each interconnect device in the network that supports the security feature and maintains an interconnect device list of such devices, which may include token ring switches, Ethernet switches, bridges and routers. The managed hub detects an intrusion by an unauthorized address on one of its ports and notifies the interconnect devices of the intrusion by transmitting a security breach detected frame. After each interconnect device sets a filter on its respective ports against the intruding unauthorized address and sends a filter set frame to the managed hub, the port in the managed hub where the security intrusion occurred is reenabled.

35 Claims, 15 Drawing Sheets



SET PORT #1 IN MANAGED HUB TO THE CURRENT PORT
200
210
IS THERE AN ADDRESS DETECTED FOR THE CURRENT PORT ?
NO
YES
220
IS THE ADDRESS ON THE CURRENT PORT IN THE LIST OF AUTHORIZED ADDRESSES ?
NO
YES
IS THE CURRENT PORT ALREADY DISABLED ?
YES
NO
250
240
SET THE CURRENT PORT NUMBER TO THE NEXT PORT IN THE MANAGED HUB
230
IS THE CURRENT PORT THE LAST PORT IN THE MANAGED HUB ?
NO
YES
DISABLE THE CURRENT PORT
260
ADD ITEM TO THE BREACH LIST
265
TRANSMIT SECURITY BREACH DETECTED FRAME ON ALL NETWORK SEGMENTS
270
OPTIONALLY TRANSMIT TRAP FRAME TO THE NETWORK MANAGEMENT STATION
280
IS THIS A TOKEN RING NETWORK ?
YES
NO
290
TRANSMIT FRAME TO THE FUNCTIONAL ADDRESS OF THE LAN MANAGER
295



26
MANAGEMENT STATION
FDDI RING
20
MANAGED HUB
TCP/IP SUBNET 1
18
BRIDGE
MANAGED HUB
22
12
ROUTER 1
10
28
TCP/IP SUBNET 3
WORKSTATION
14
ROUTER 2
MANAGED HUB
24
TCP/IP SUBNET 4
16
TOKEN RING SWITCH
TOKEN RING
30
FILE SERVER

FIG. 1

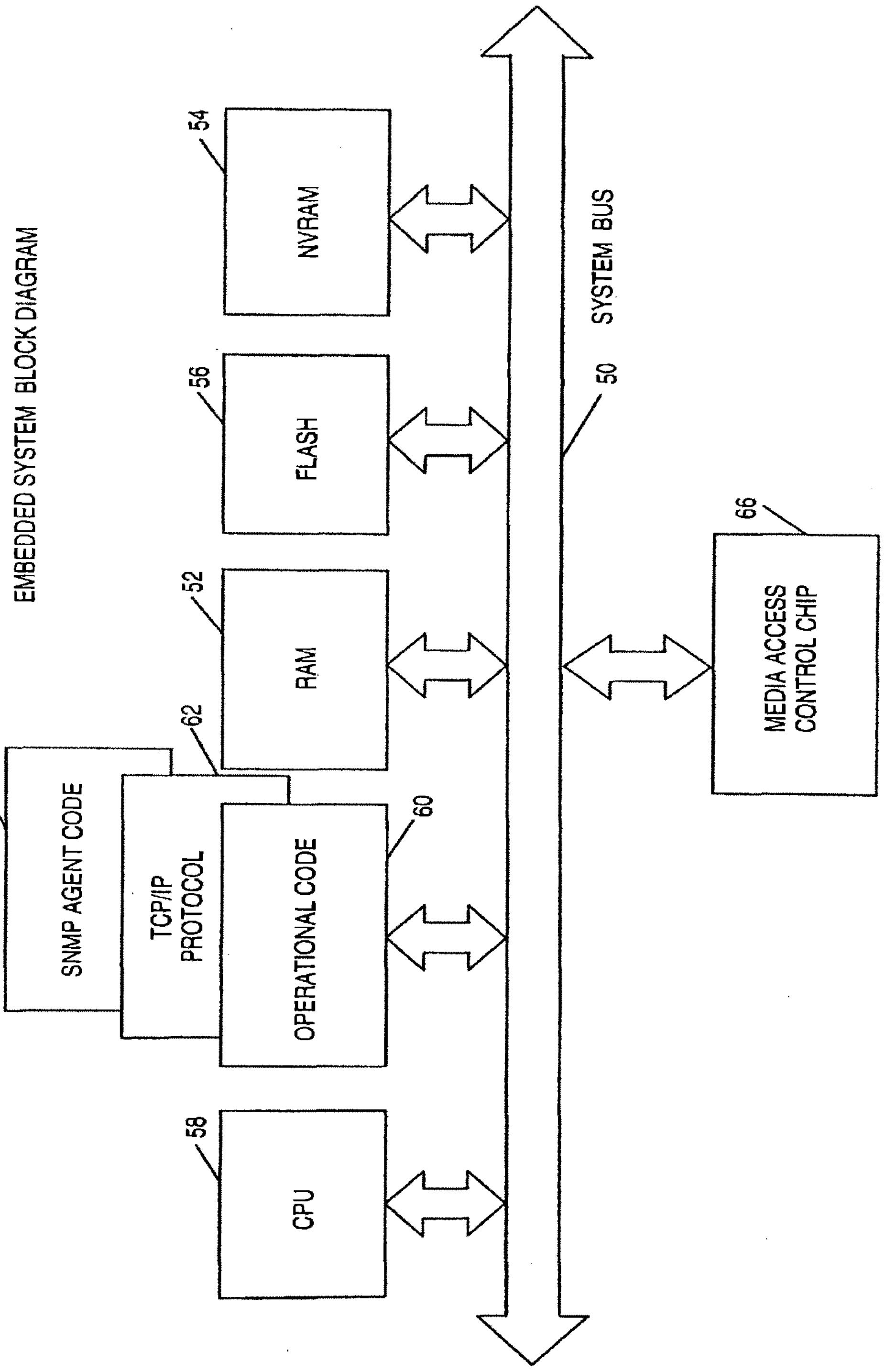

EMBEDDED SYSTEM BLOCK DIAGRAM
54
NVRAM
56
FLASH
52
RAM
62
TCP/IP PROTOCOL
64
SNMP AGENT CODE
60
OPERATIONAL CODE
58
CPU
SYSTEM BUS
50
66
MEDIA ACCESS CONTROL CHIP

FIG. 2

NETWORK MANAGEMENT STATION BLOCK DIAGRAM

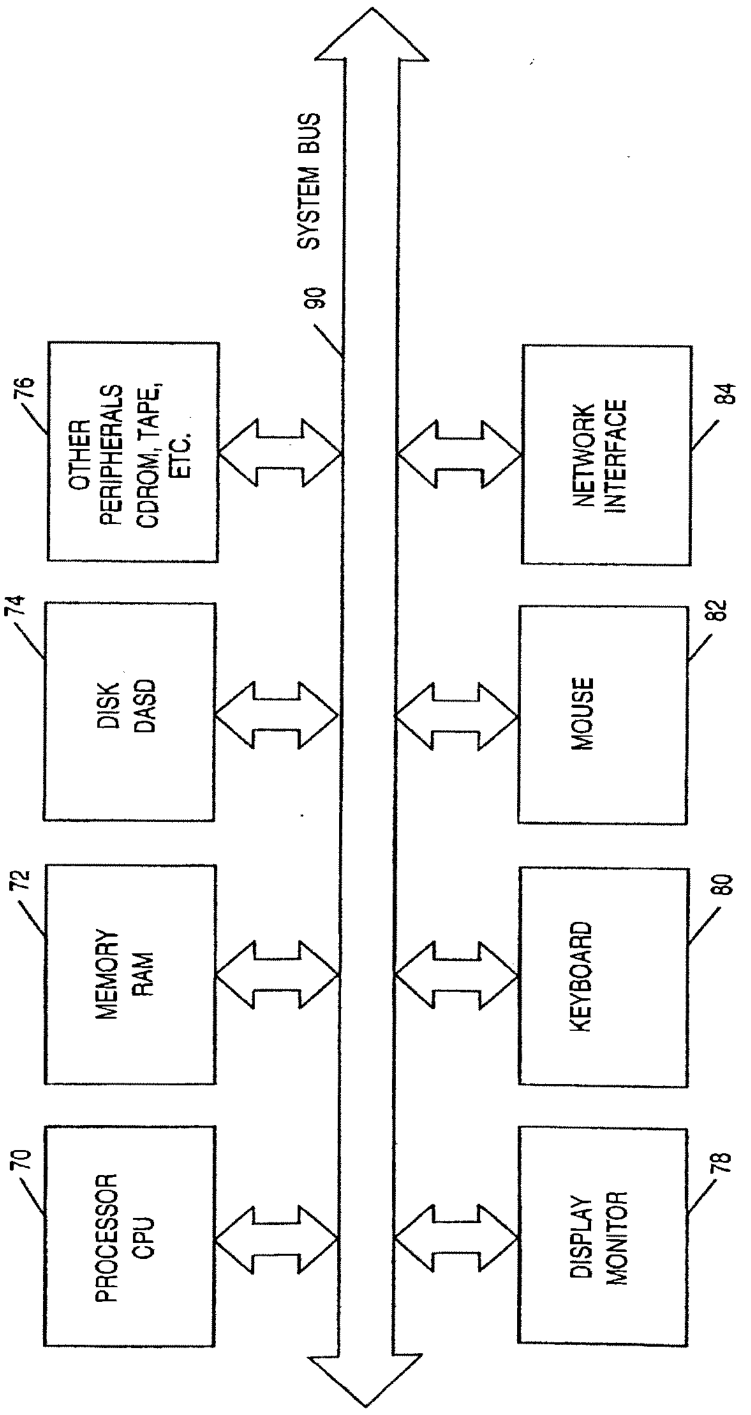


FIG. 3

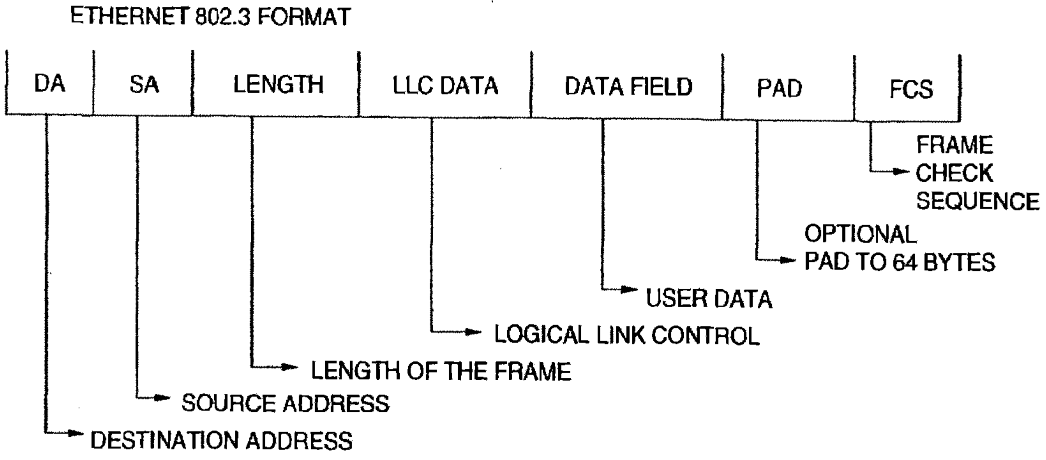


FIG. 4A

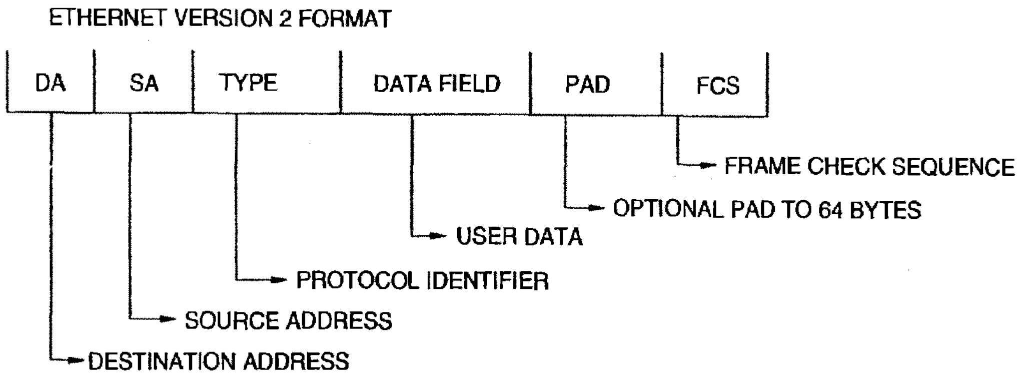


FIG. 4B

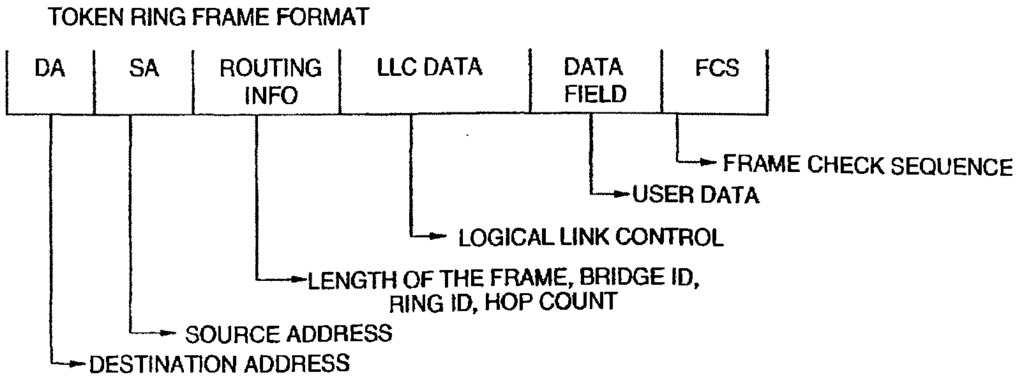


FIG. 4C

DISCOVERY REQUEST

FRAME TYPE IDENTIFIER	TIME STAMP	
1	4	BYTES

FIG. 5A

DISCOVERY RESPONSE

FRAME TYPE IDENTIFIER	INTERCONNECT DEVICE MAC ADDRESS	INTERCONNECT DEVICE DESCRIPTION	TIME STAMP	
1	6	50	4	BYTES

FIG. 5B

SECURITY BREACH DETECTED FRAME

FRAME TYPE IDENTIFIER	INTRUDING MAC ADDRESS	MODULE NUMBER	PORT NUMBER	TIME STAMP	DEVICE FIELD LENGTH	ADDRESSES	
1	6	1	1	4	2	VARIABLE LENGTH	BYTES

FIG. 5C

FILTER SET FRAME

FRAME TYPE IDENTIFIER	INTERCONNECT DEVICE MAC ADDRESS	INTRUDING MAC ADDRESS	MODULE NUMBER	PORT NUMBER	TIME STAMP	
1	6	6	1	1	4	BYTES

FIG. 5D

SECURITY CLEAR CONDITION

FRAME TYPE IDENTIFIER	INTRUDING MAC ADDRESS	
1	6	BYTES

FIG. 5E

INTERCONNECT DEVICE LIST ITEM			
MAC ADDRESS	DEVICE DESCRIPTION	LAST RESPONSE TIME	OUTSTANDING BREACH RESPONSE COUNT

MAC ADDRESS: MAC ADDRESS OF THE INTERCONNECT DEVICE

DEVICE DESCRIPTION: ASCII SELF DESCRIPTION PROVIDED BY THE INTERCONNECT DEVICE

LAST RESPONSE TIME: TIME WHEN LAST RESPONSE RECEIVED FROM INTERCONNECT DEVICE

OUTSTANDING BREACH RESPONSE COUNT: NUMBER OF SECURITY BREACH FRAMES THE INTERCONNECT DEVICE HAS NOT RESPONDED TO

FIG. 6

BREACH LIST ITEM				
MAC ADDRESS	BREACH TIME	BREACH PORT	BREACH MODULE	OUTSTANDING FILTER SET COUNT

MAC ADDRESS: MAC ADDRESS OF THE INTRUDING DEVICE

BREACH TIME: TIME WHEN INTRUSION OCCURRED

BREACH PORT: PORT IN MANAGED HUB WHEN INTRUSION OCCURRED

BREACH MODULE: MODULE IN MANAGED HUB WHEN INTRUSION OCCURRED

OUTSTANDING FILTER SET COUNT: NUMBER OF FILTER SET FRAMES NOT RECEIVED YET

FIG. 7

INTRUSION LIST ITEM			
MAC ADDRESS	BREACH TIME	BREACH PORT	BREACH MODULE

MAC ADDRESS: MAC ADDRESS OF THE INTRUDING DEVICE

BREACH TIME: TIME WHEN INTRUSION OCCURRED

BREACH PORT: PORT IN MANAGED HUB WHEN INTRUSION OCCURRED

BREACH MODULE: MODULE IN MANAGED HUB WHEN INTRUSION OCCURRED

FIG. 8

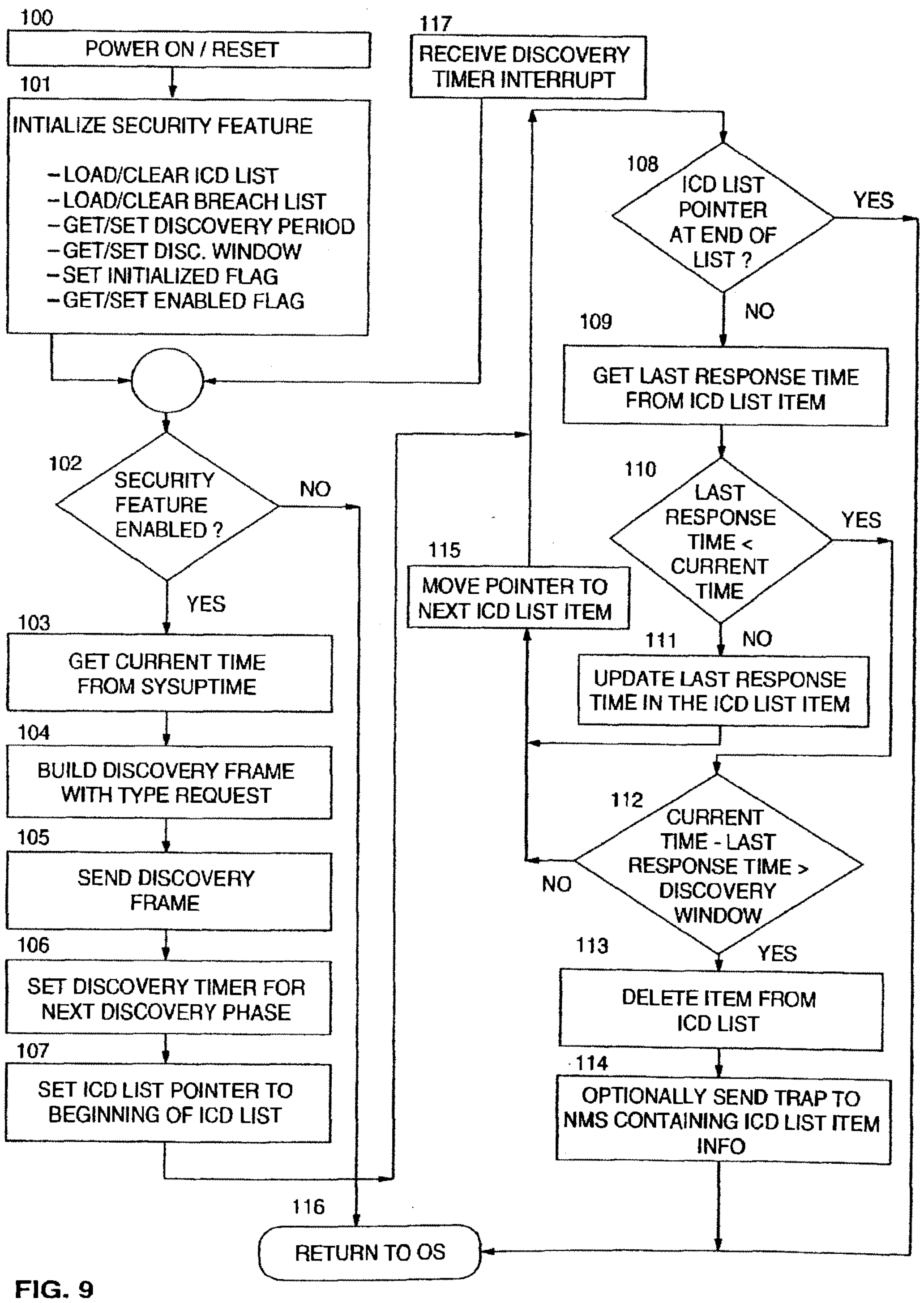

100
POWER ON / RESET
101
INTIALIZE SECURITY FEATURE
-LOAD/CLEAR ICD LIST
-LOAD/CLEAR BREACH LIST
-GET/SET DISCOVERY PERIOD
-GET/SET DISC. WINDOW
-SET INITIALIZED FLAG
-GET/SET ENABLED FLAG
117
RECEIVE DISCOVERY TIMER INTERRUPT
102
SECURITY FEATURE ENABLED ?
NO
YES
103
GET CURRENT TIME FROM SYSUPTIME
104
BUILD DISCOVERY FRAME WITH TYPE REQUEST
105
SEND DISCOVERY FRAME
106
SET DISCOVERY TIMER FOR NEXT DISCOVERY PHASE
107
SET ICD LIST POINTER TO BEGINNING OF ICD LIST
108
ICD LIST POINTER AT END OF LIST ?
YES
NO
109
GET LAST RESPONSE TIME FROM ICD LIST ITEM
110
LAST RESPONSE TIME < CURRENT TIME
YES
NO
115
MOVE POINTER TO NEXT ICD LIST ITEM
111
UPDATE LAST RESPONSE TIME IN THE ICD LIST ITEM
112
CURRENT TIME - LAST RESPONSE TIME > DISCOVERY WINDOW
NO
YES
113
DELETE ITEM FROM ICD LIST
114
OPTIONALLY SEND TRAP TO NMS CONTAINING ICD LIST ITEM INFO
116
RETURN TO OS

FIG. 9

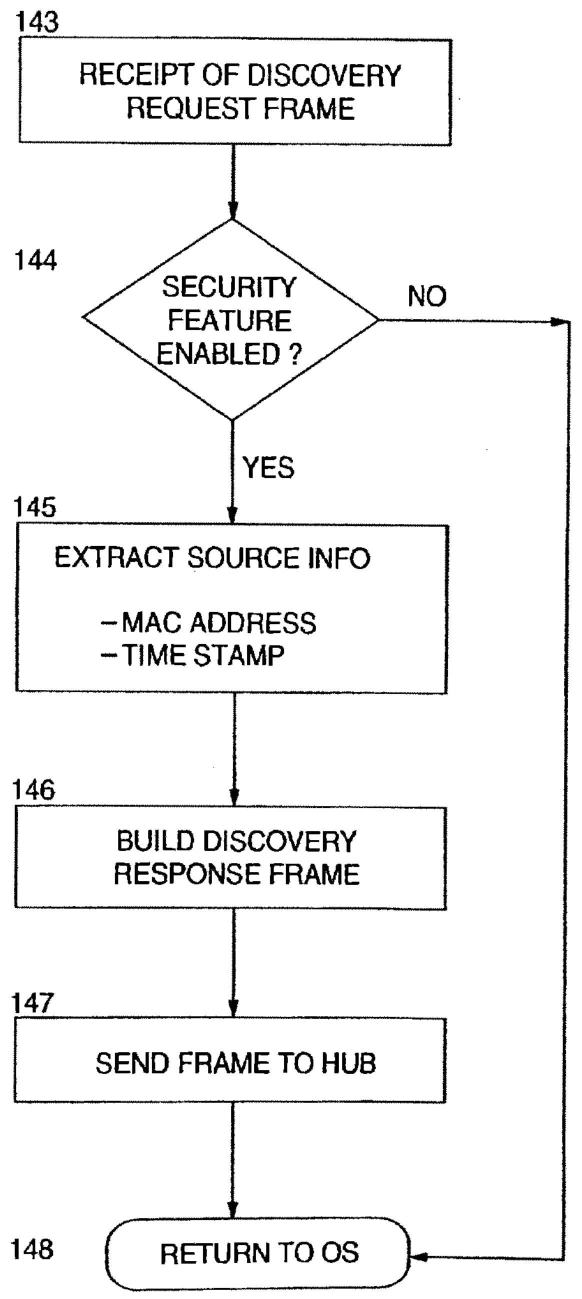

143
RECEIPT OF DISCOVERY REQUEST FRAME
144
SECURITY FEATURE ENABLED ?
NO
YES
145
EXTRACT SOURCE INFO
-MAC ADDRESS
-TIME STAMP
146
BUILD DISCOVERY RESPONSE FRAME
147
SEND FRAME TO HUB
148
RETURN TO OS

FIG. 10

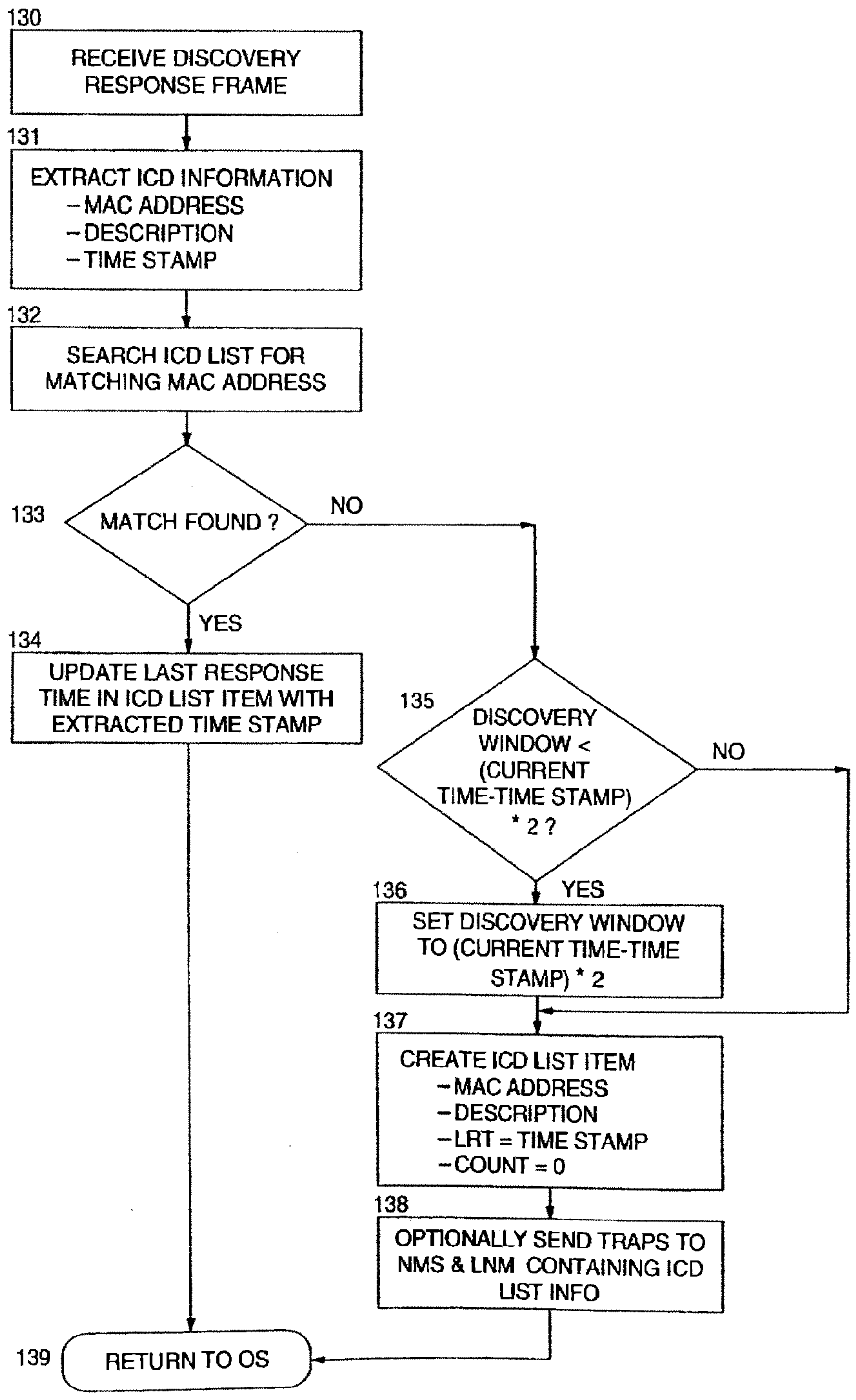
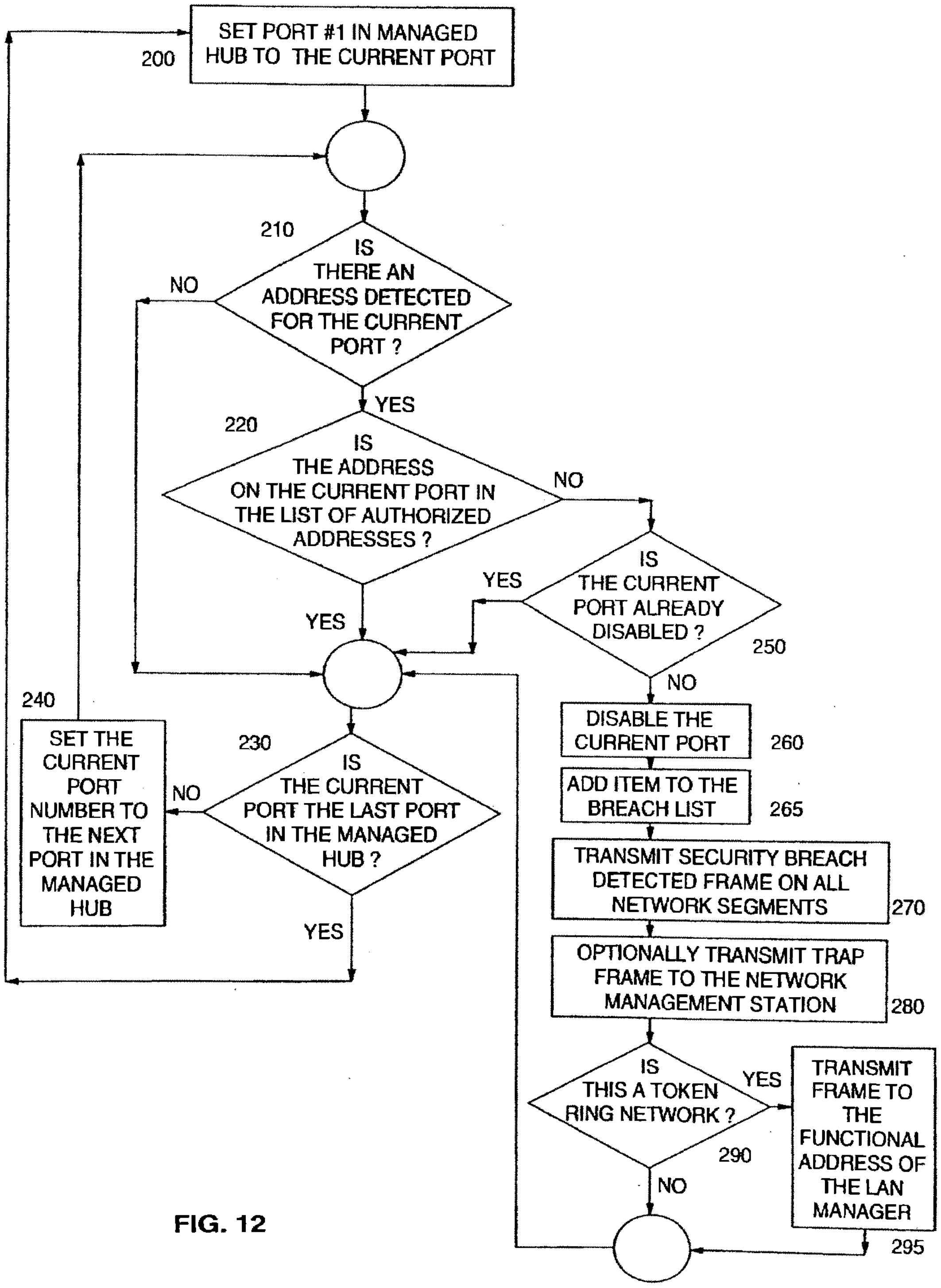
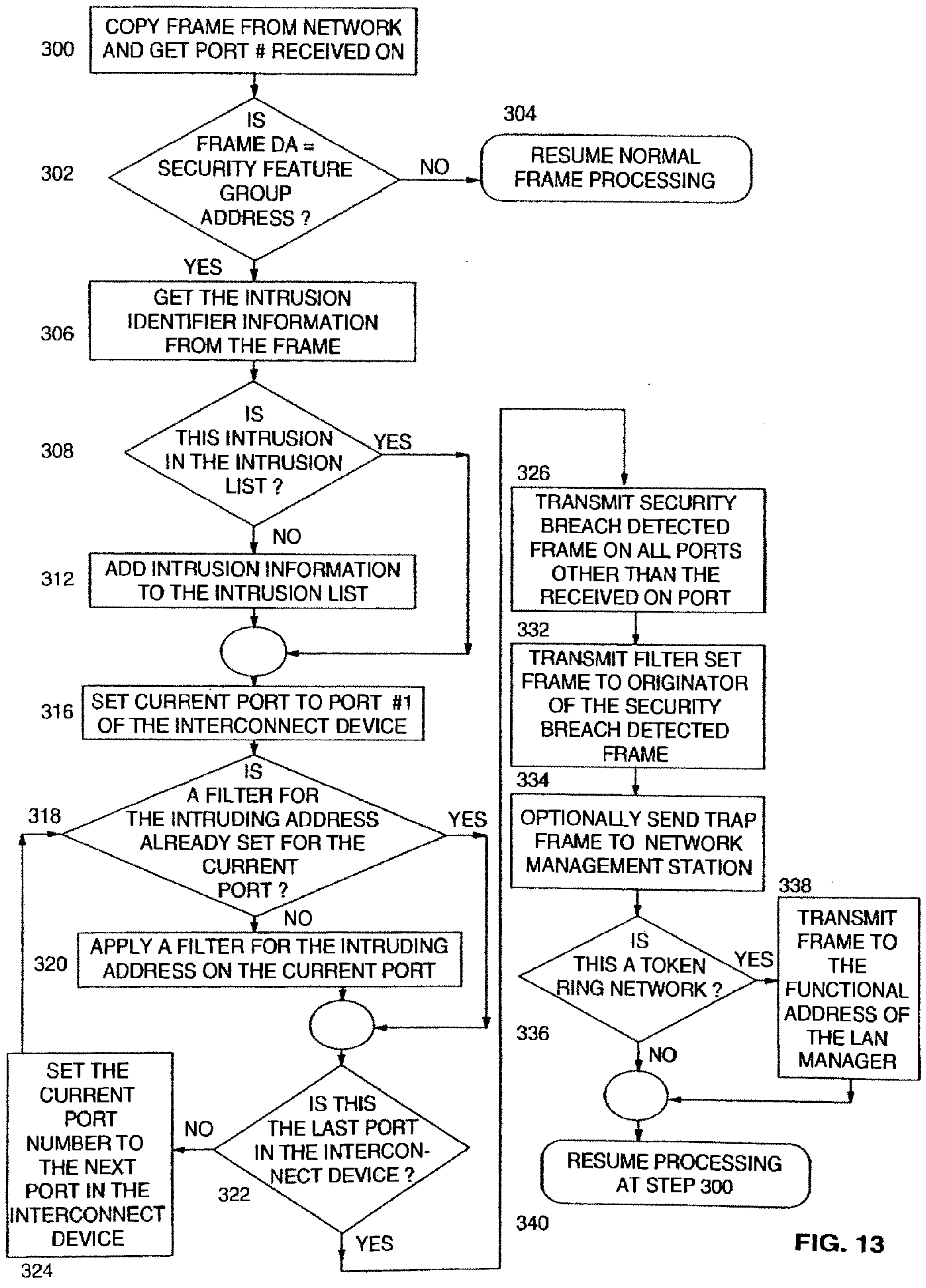

130
RECEIVE DISCOVERY RESPONSE FRAME
131
EXTRACT ICD INFORMATION
–MAC ADDRESS
–DESCRIPTION
–TIME STAMP
132
SEARCH ICD LIST FOR MATCHING MAC ADDRESS
133
MATCH FOUND ?
NO
YES
134
UPDATE LAST RESPONSE TIME IN ICD LIST ITEM WITH EXTRACTED TIME STAMP
135
DISCOVERY WINDOW < (CURRENT TIME-TIME STAMP) * 2 ?
NO
YES
136
SET DISCOVERY WINDOW TO (CURRENT TIME-TIME STAMP) * 2
137
CREATE ICD LIST ITEM
–MAC ADDRESS
–DESCRIPTION
–LRT = TIME STAMP
–COUNT = 0
138
OPTIONALLY SEND TRAPS TO NMS & LNM CONTAINING ICD LIST INFO
139
RETURN TO OS

FIG. 11



SET PORT #1 IN MANAGED HUB TO THE CURRENT PORT
200
210
IS THERE AN ADDRESS DETECTED FOR THE CURRENT PORT ?
NO
YES
220
IS THE ADDRESS ON THE CURRENT PORT IN THE LIST OF AUTHORIZED ADDRESSES ?
NO
YES
IS THE CURRENT PORT ALREADY DISABLED ?
YES
250
NO
DISABLE THE CURRENT PORT
260
ADD ITEM TO THE BREACH LIST
265
TRANSMIT SECURITY BREACH DETECTED FRAME ON ALL NETWORK SEGMENTS
270
OPTIONALLY TRANSMIT TRAP FRAME TO THE NETWORK MANAGEMENT STATION
280
IS THIS A TOKEN RING NETWORK ?
YES
290
NO
TRANSMIT FRAME TO THE FUNCTIONAL ADDRESS OF THE LAN MANAGER
295
240
SET THE CURRENT PORT NUMBER TO THE NEXT PORT IN THE MANAGED HUB
230
IS THE CURRENT PORT THE LAST PORT IN THE MANAGED HUB ?
NO
YES

FIG. 12



300
COPY FRAME FROM NETWORK AND GET PORT # RECEIVED ON
302
IS FRAME DA = SECURITY FEATURE GROUP ADDRESS ?
NO
304
RESUME NORMAL FRAME PROCESSING
YES
306
GET THE INTRUSION IDENTIFIER INFORMATION FROM THE FRAME
308
IS THIS INTRUSION IN THE INTRUSION LIST ?
YES
NO
312
ADD INTRUSION INFORMATION TO THE INTRUSION LIST
316
SET CURRENT PORT TO PORT #1 OF THE INTERCONNECT DEVICE
318
IS A FILTER FOR THE INTRUDING ADDRESS ALREADY SET FOR THE CURRENT PORT ?
YES
NO
320
APPLY A FILTER FOR THE INTRUDING ADDRESS ON THE CURRENT PORT
SET THE CURRENT PORT NUMBER TO THE NEXT PORT IN THE INTERCONNECT DEVICE
324
NO
IS THIS THE LAST PORT IN THE INTERCON-NECT DEVICE ?
322
YES
326
TRANSMIT SECURITY BREACH DETECTED FRAME ON ALL PORTS OTHER THAN THE RECEIVED ON PORT
332
TRANSMIT FILTER SET FRAME TO ORIGINATOR OF THE SECURITY BREACH DETECTED FRAME
334
OPTIONALLY SEND TRAP FRAME TO NETWORK MANAGEMENT STATION
338
TRANSMIT FRAME TO THE FUNCTIONAL ADDRESS OF THE LAN MANAGER
IS THIS A TOKEN RING NETWORK ?
YES
336
NO
RESUME PROCESSING AT STEP 300
340

FIG. 13

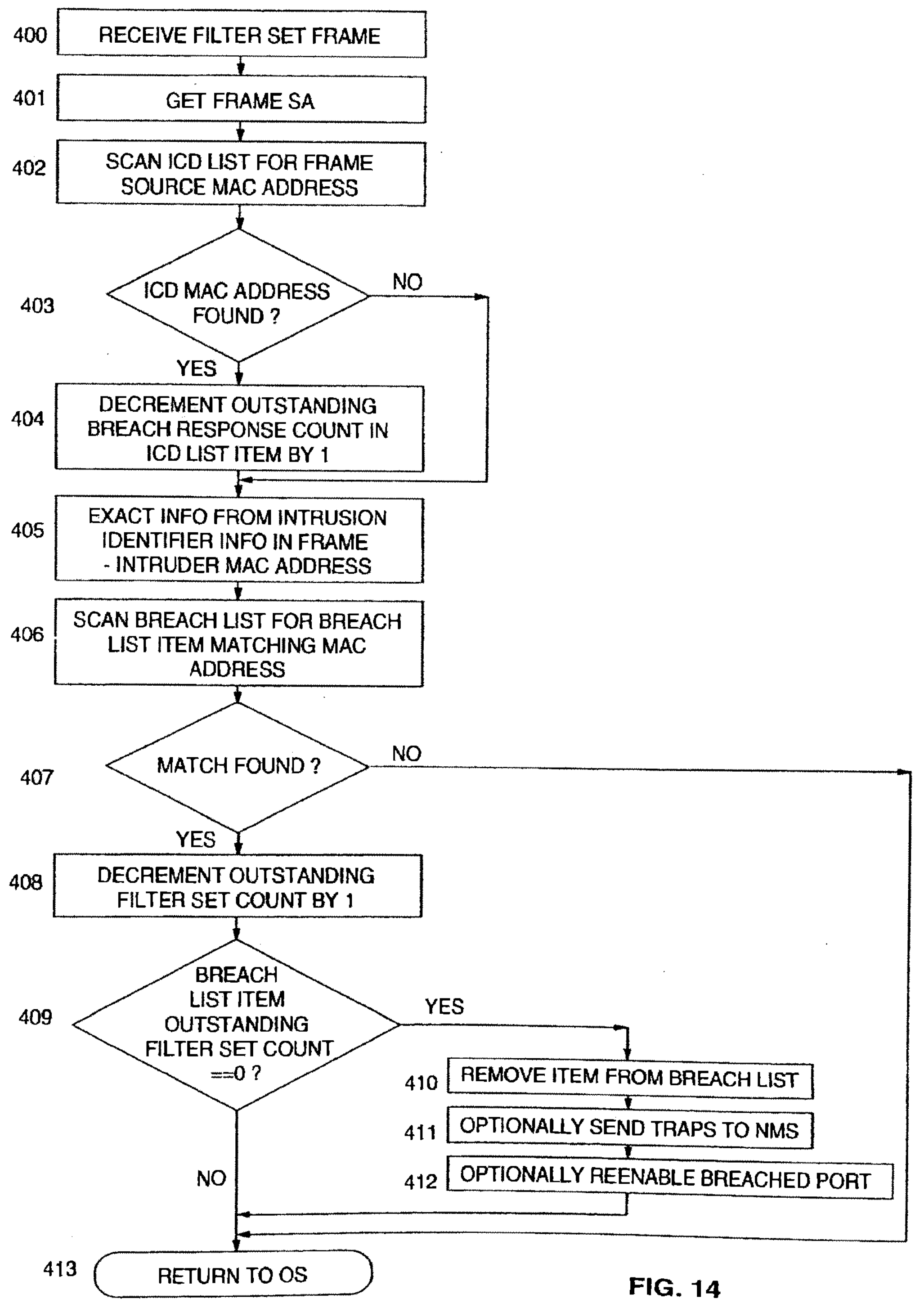

400
RECEIVE FILTER SET FRAME
401
GET FRAME SA
402
SCAN ICD LIST FOR FRAME SOURCE MAC ADDRESS
403
ICD MAC ADDRESS FOUND ?
NO
YES
404
DECREMENT OUTSTANDING BREACH RESPONSE COUNT IN ICD LIST ITEM BY 1
405
EXACT INFO FROM INTRUSION IDENTIFIER INFO IN FRAME - INTRUDER MAC ADDRESS
406
SCAN BREACH LIST FOR BREACH LIST ITEM MATCHING MAC ADDRESS
407
MATCH FOUND ?
NO
YES
408
DECREMENT OUTSTANDING FILTER SET COUNT BY 1
409
BREACH LIST ITEM OUTSTANDING FILTER SET COUNT ==0 ?
YES
410
REMOVE ITEM FROM BREACH LIST
411
OPTIONALLY SEND TRAPS TO NMS
412
OPTIONALLY REENABLE BREACHED PORT
NO
413
RETURN TO OS

FIG. 14

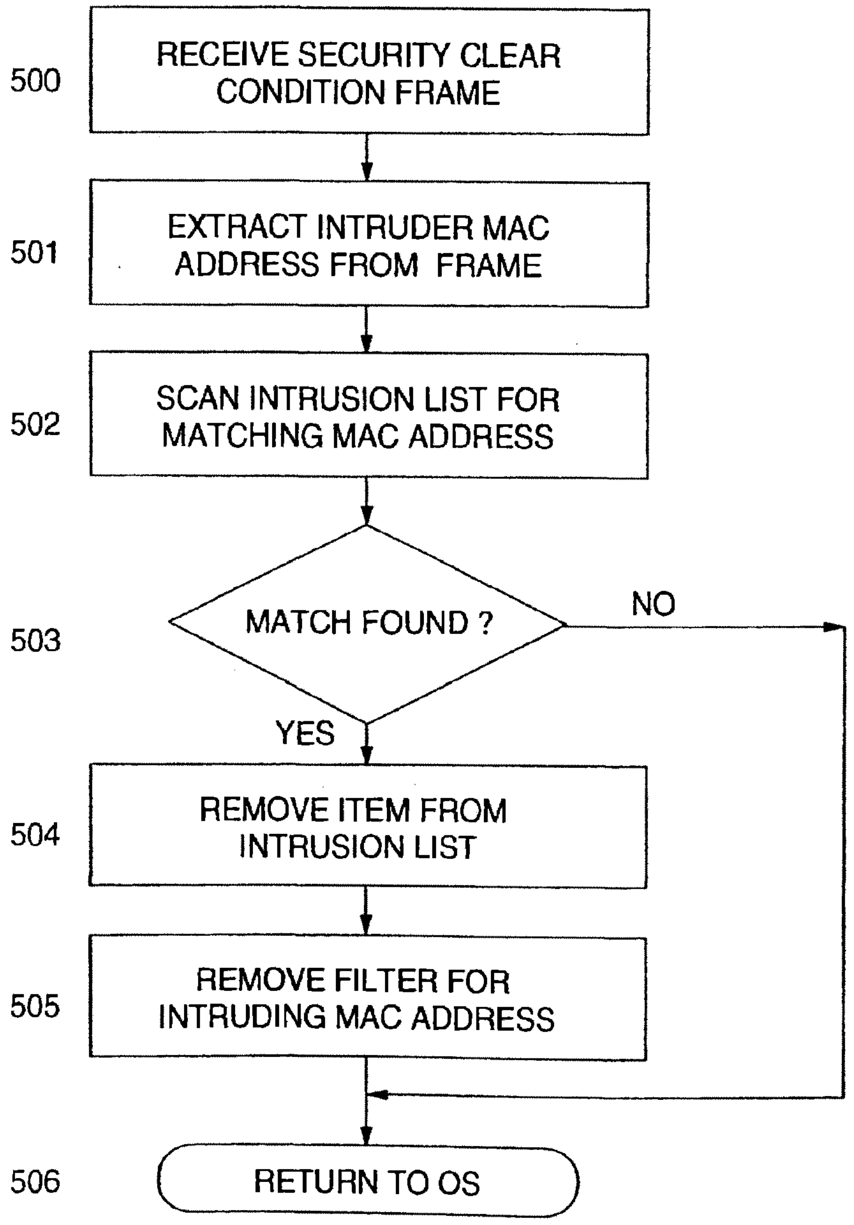

500
RECEIVE SECURITY CLEAR CONDITION FRAME
501
EXTRACT INTRUDER MAC ADDRESS FROM FRAME
502
SCAN INTRUSION LIST FOR MATCHING MAC ADDRESS
503
MATCH FOUND ?
NO
YES
504
REMOVE ITEM FROM INTRUSION LIST
505
REMOVE FILTER FOR INTRUDING MAC ADDRESS
506
RETURN TO OS

FIG. 15

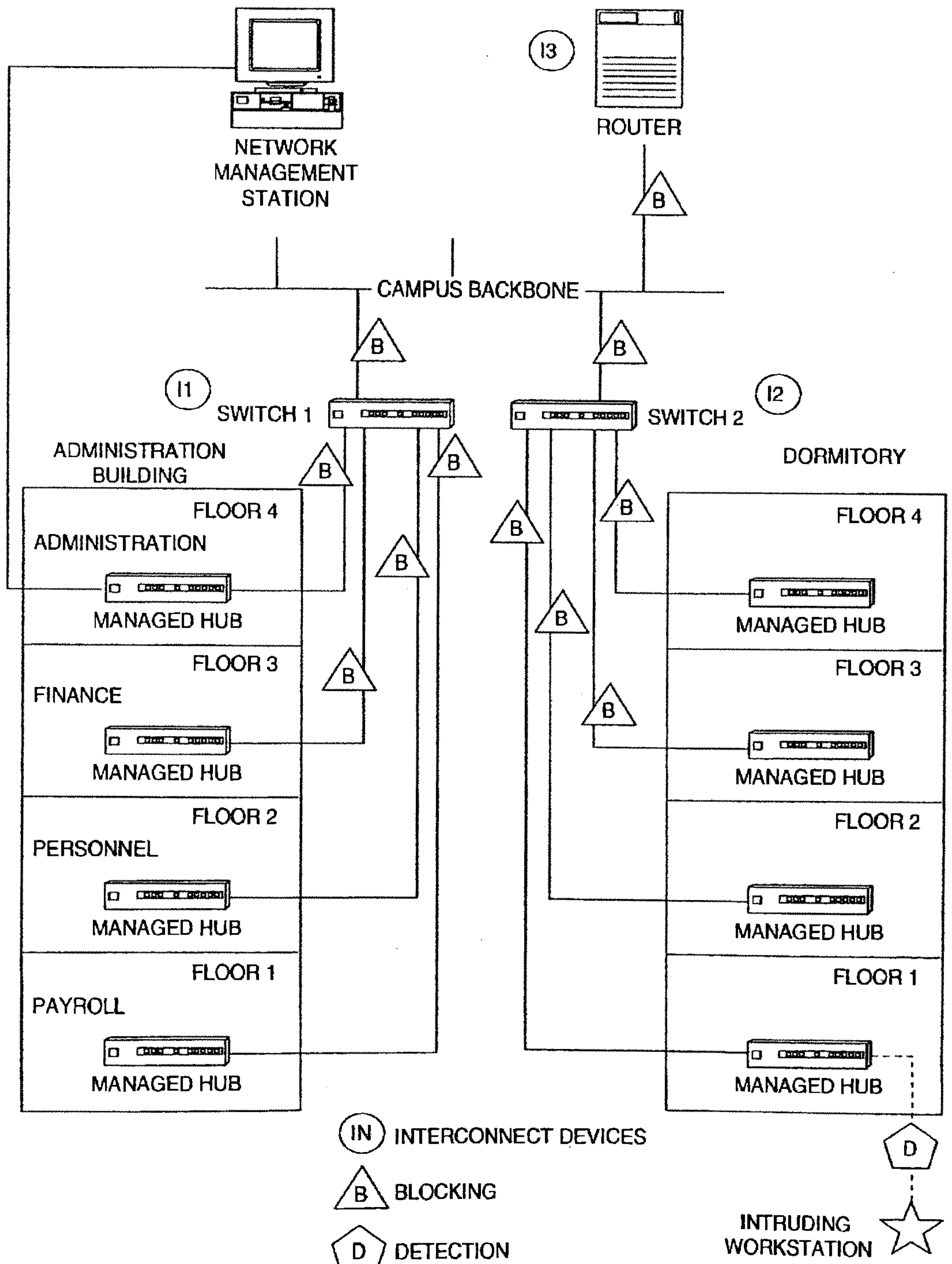

NETWORK MANAGEMENT STATION
I3
ROUTER
B
CAMPUS BACKBONE
B
B
I1
SWITCH 1
SWITCH 2
I2
ADMINISTRATION BUILDING
DORMITORY
B
B
B
B
FLOOR 4
ADMINISTRATION
MANAGED HUB
FLOOR 3
FINANCE
MANAGED HUB
FLOOR 2
PERSONNEL
MANAGED HUB
FLOOR 1
PAYROLL
MANAGED HUB
B
B
B
B
FLOOR 4
MANAGED HUB
FLOOR 3
MANAGED HUB
FLOOR 2
MANAGED HUB
FLOOR 1
MANAGED HUB
IN INTERCONNECT DEVICES
B BLOCKING
D DETECTION
D
INTRUDING WORKSTATION

FIG. 16

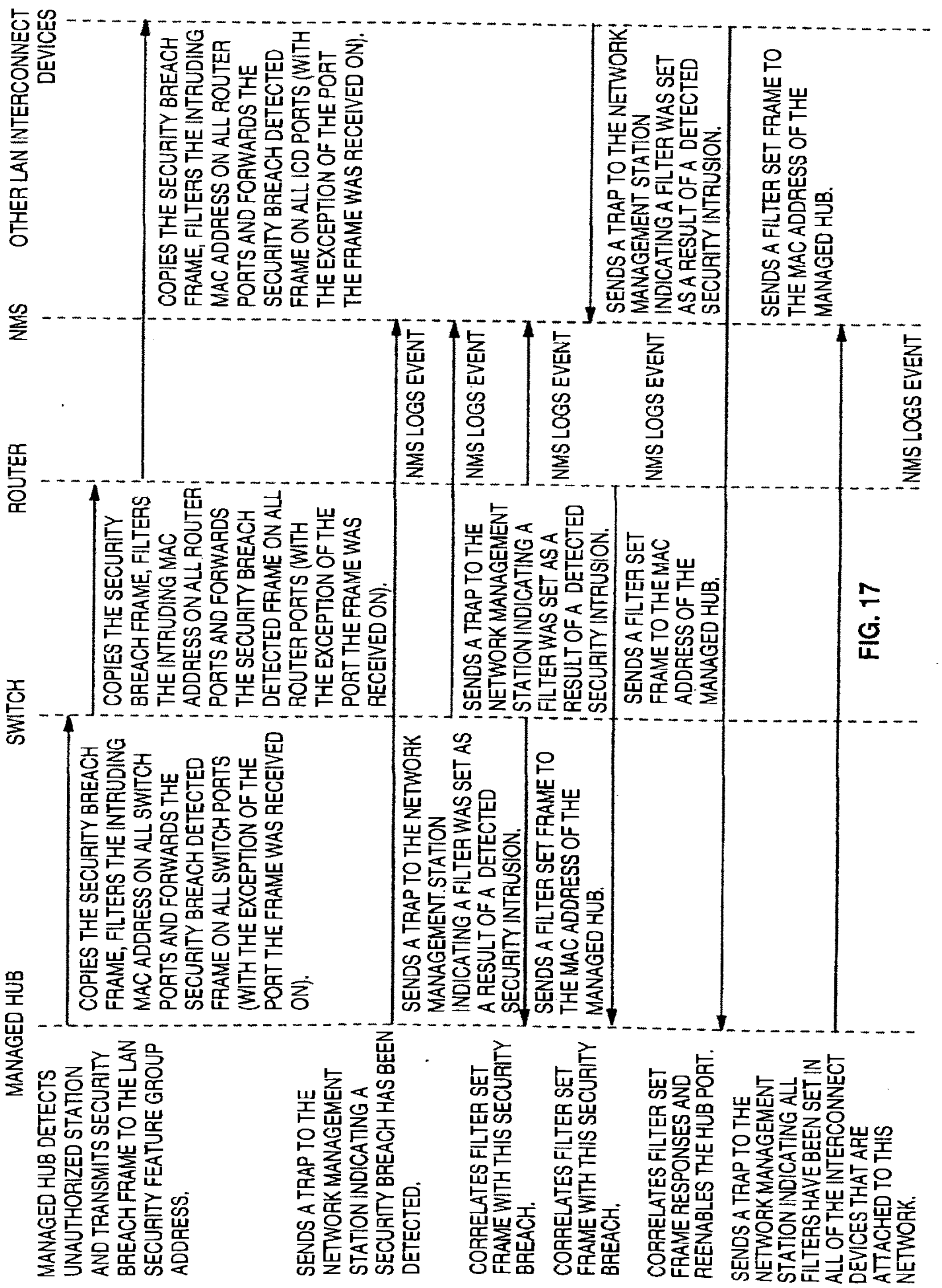

MANAGED HUB
SWITCH
ROUTER
NMS
OTHER LAN INTERCONNECT DEVICES
MANAGED HUB DETECTS UNAUTHORIZED STATION AND TRANSMITS SECURITY BREACH FRAME TO THE LAN SECURITY FEATURE GROUP ADDRESS.
COPIES THE SECURITY BREACH FRAME, FILTERS THE INTRUDING MAC ADDRESS ON ALL SWITCH PORTS AND FORWARDS THE SECURITY BREACH DETECTED FRAME ON ALL SWITCH PORTS (WITH THE EXCEPTION OF THE PORT THE FRAME WAS RECEIVED ON).
COPIES THE SECURITY BREACH FRAME, FILTERS THE INTRUDING MAC ADDRESS ON ALL ROUTER PORTS AND FORWARDS THE SECURITY BREACH DETECTED FRAME ON ALL ROUTER PORTS (WITH THE EXCEPTION OF THE PORT THE FRAME WAS RECEIVED ON).
COPIES THE SECURITY BREACH FRAME, FILTERS THE INTRUDING MAC ADDRESS ON ALL ROUTER PORTS AND FORWARDS THE SECURITY BREACH DETECTED FRAME ON ALL ICD PORTS (WITH THE EXCEPTION OF THE PORT THE FRAME WAS RECEIVED ON).
SENDS A TRAP TO THE NETWORK MANAGEMENT STATION INDICATING A SECURITY BREACH HAS BEEN DETECTED.
NMS LOGS EVENT
SENDS A TRAP TO THE NETWORK MANAGEMENT STATION INDICATING A FILTER WAS SET AS A RESULT OF A DETECTED SECURITY INTRUSION.
NMS LOGS EVENT
SENDS A TRAP TO THE NETWORK MANAGEMENT STATION INDICATING A FILTER WAS SET AS A RESULT OF A DETECTED SECURITY INTRUSION.
NMS LOGS EVENT
SENDS A TRAP TO THE NETWORK MANAGEMENT STATION INDICATING A FILTER WAS SET AS A RESULT OF A DETECTED SECURITY INTRUSION.
NMS LOGS EVENT
CORRELATES FILTER SET FRAME WITH THIS SECURITY BREACH.
SENDS A FILTER SET FRAME TO THE MAC ADDRESS OF THE MANAGED HUB.
CORRELATES FILTER SET FRAME WITH THIS SECURITY BREACH.
SENDS A FILTER SET FRAME TO THE MAC ADDRESS OF THE MANAGED HUB.
CORRELATES FILTER SET FRAME RESPONSES AND REENABLES THE HUB PORT.
SENDS A FILTER SET FRAME TO THE MAC ADDRESS OF THE MANAGED HUB.
SENDS A TRAP TO THE NETWORK MANAGEMENT STATION INDICATING ALL FILTERS HAVE BEEN SET IN ALL OF THE INTERCONNECT DEVICES THAT ARE ATTACHED TO THIS NETWORK.
NMS LOGS EVENT

FIG. 17

IW 7112490

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

March 11, 2008

THIS IS TO CERTIFY THAT ANNEXED IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF THE FILE WRAPPER AND CONTENTS OF:

APPLICATION NUMBER: *09/504,783*
FILING DATE: *February 15, 2000*
PATENT NUMBER: *6,502,135*
ISSUE DATE: *December 31, 2002*

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

[illegible]

M. TARVER
Certifying Officer

UNITED STATES PATENT AND TRADEMARK OFFICE DEPARTMENT OF COMMERCE

PART (4) OF (7) PART(S)

MANAGED NETWORK DEVICE SECURITY METHOD AND APPARATUS

Reference to Related Application

This application is related to the following application having the same assignee and inventorship and containing common disclosure, and is believed to have an identical effective filing date: "System and Method for Detecting and Preventing Security Intrusions in Campus LAN Networks", Ser. No. 08/780804.

BACKGROUND OF THE INVENTION

This invention relates in general to computer network security systems and in particular to systems and methods for detecting and preventing intrusion into a campus local area network by an unauthorized user.

As local area networks (LANS) continue to proliferate, and the number of personal computers (PCs) connected to LANs continue to grow at a rapid pace, network security becomes an ever increasing problem for network administrators. As the trend of deploying distributed LANs continues, this provides multiple access points to an enterprise's network. Each of these distributed access points, if not controlled, is a potential security risk to the network.

To further illustrate the demand for improved network security, an IDC report on network management, "LAN Management: The Pivotal Role of Intelligent Hubs", published in 1993, highlighted the importance of network security to LAN administrators. When asked the importance of improving management of specific LAN devices, 75% of the respondents stated network security is very important. When further asked about the growing importance of network security over the next three years, many respondents indicated that it would increase in importance.

More recently, a request for proposal from the U.S. Federal Reserve specified a requirement that a LAN hub must detect an unauthorized station at the port level and disable the port within a 10-second period. Although this requirement will stop an intruder, there is an inherent weakness in this solution in that it only isolates the security intrusion to the port of entry. The rest of the campus network is unaware of an attempted break-in. The detection of the unauthorized station and the disabling of the port is the first reaction to a security intrusion, but many significant enhancements can be made to provide a network-wide security mechanism. Where the above solution stops at the hub/port level, this invention provides significant enhancements to solving the problem of network security by presenting a system wide solution to detecting and preventing security intrusions in a campus LAN environment.

In today's environment, network administrators focus their attention on router management, hub management, server management, and switch management, with the goals of ensuring network up time and managing growth (capacity planning). Security is often an afterthought and at best administrators get security as a by-product of employing other device functions. For example, network administrators may set filters at router, switch, or bridge ports for performance improvements and implicitly realize some level of security as a side effect since the filters control the flow of frames to LAN segments.

The problem with using filters is that their primary focus is on performance improvements, by restricting the flow of certain types of network traffic to specified LAN segments. The filters do not indicate how many times the filter has actually been used and do not indicate a list of the media access control (MAC) addresses that have been filtered. Therefore, filters do not provide an adequate detection mechanism against break-in attempts.

Another security technique that is commonly employed in hubs is intrusion control. There are token ring and Ethernet managed hubs that allow a network administrator to define, by MAC address, one or more authorized users per hub port. If an unauthorized MAC address is detected at the hub port, then the port is automatically disabled. The problem with this solution is that prevention stops at the hub and no further action is taken once the security intrusion has been detected. This solution does not provide a network-centric, system-wide solution. It only provides a piecemeal solution for a particular type of network hardware namely, the token ring and Ethernet managed hubs. The result is a fragmented solution, where security may exist for some work groups that have managed hubs installed, but not for the entire campus network. At best, the security detection/prevention is localized to the hub level and no solution exists for a network-wide solution.

Other attempts to control LAN access have been done with software program products. For example, IBM Corporation's Lan Network Management (LNM) products LNM for OS2 and LNM for AIX both provide functions called access control to token ring LANs. There are several problems with these solutions. One problem with both of these solutions is that it takes a long time to detect that an unauthorized station has inserted into the ring. An intruder could have ample time to compromise the integrity of a LAN segment before LNM could take an appropriate action. Another problem with the LNM products is that once an unauthorized MAC address has been detected, LNM issues a remove ring station MAC frame. Although this MAC frame removes the station from the ring, it does not prevent the station from reinserting into the ring and potentially causing more damage. Because these products do not provide foolproof solutions, and significant security exposure still exists, they do not provide a viable solution to the problem of network security for campus LAN environments.

Thus, there is a need for a mechanism in the managed devices of a computer network that enables a comprehensive solution and that not only provides for detection of security intrusions, but also provides the proactive actions needed to stop the proliferation of security intrusions over the domain of an entire campus network.

SUMMARY OF THE INVENTION

It is, therefore, an object of the invention to provide an apparatus and method in a managed device for detecting and preventing security intrusions in a computer network.

It is another object of the invention to provide an apparatus and method in a managed hub for detecting and preventing security intrusions in a computer network.

Overall, this invention can be described in terms of the following procedures or phases: discovery, detection, prevention, hub enable, and security clear. During each of these phases, a series of frames are transmitted between the interconnect devices on a campus network. These frames are addressed to a group address (multicast address). This well known group address needs to be defined and reserved for the LAN security functions that are described herein. This group address will be referred to as LAN security feature group address throughout the rest of this description.

The campus LAN security feature relies on managed hubs discovering the interconnect devices in the campus LAN segment that support this LAN security feature. The term "LAN interconnect device" is used throughout this description to refer to LAN switches (token ring and Ethernet 10/100 Mbps), LAN bridges and routers. The managed hub maintains a list of authorized MAC addresses for each port in the managed hub. If the managed hub detects an unauthorized station connecting to the LAN, the hub disables the port and then transmits a security breach detected frame to the LAN security feature group address. Each of the LAN interconnect devices on the campus LAN segment copies the LAN security feature group address and performs the following steps: 1) set up filters to filter the intruding MAC address; 2) forward the LAN security feature group address to other segments attached to the LAN interconnect device; and 3) send an acknowledgement back to the managed hub indicating that the intruding address has been filtered at the LAN interconnect device. Once the managed hub receives acknowledgements from all of the interconnect devices in the campus LAN, the port where the security intrusion was detected is re-enabled for use. Another part of the invention provides a network management station with the capability to override any security filter that was set in the above process.

The following is a brief description of each phase in the preferred embodiment of the invention:

1. Discovery

In this phase, the managed hub determines the interconnect devices in the campus network that are capable of supporting the LAN security feature. The managed hub periodically sends a discovery frame to the LAN security feature group address. The managed hub then uses the responses to build and maintain a table of interconnect devices in the network that support the security feature.

2. Detection

In the detection phase, the managed hub compares the MAC addresses on each port against a list of authorized MAC addresses. If an unauthorized MAC address is detected, then the managed hub disables the port and notifies the other interconnect devices in the campus network by transmitting a security breach detected frame to the LAN security feature group address.

3. Prevention

The prevention phase is initiated when a LAN interconnect device receives the security breach detected frame. Once this frame is received, the LAN interconnect device sets up a filter to prevent frames with the intruding MAC address from flowing through this network device. The LAN interconnect device then forwards the security breach detected frame to the other LAN segments attached to the interconnect device. The LAN interconnect device also transmits a filter set frame back to the managed hub.

4. Hub Enable

The hub enable phase takes place when the managed hub has received all acknowledgements from the LAN interconnect devices in the campus network. When the acknowledgements have been received, the managed hub re-enables the port where the security intrusion occurred.

5. Security Clear Condition

In this phase, a network management station can remove a filter from a LAN interconnect device that was previously set in the prevention step.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be described with respect to a preferred embodiment thereof which is further illustrated and described in the drawings.

FIG. 1 is a block diagram of a campus network in which the present invention can be implemented.

FIG. 2 is a component block diagram for an SNMP managed device.

FIG. 3 is a component block diagram for a network management station.

FIGS. 4A–4C show general frame formats for Ethernet and token ring frames.

FIGS. 5A–5E show the information contained in the Ethernet and token ring frame data fields to represent the different frame types that are implemented in the preferred embodiment.

FIG. 6 illustrates the structure of the Interconnect Device List (ICD).

FIG. 7 illustrates the structure of the Breach List.

FIG. 8 illustrates the structure of the Intrusion List.

FIG. 9 is a flow chart of the processing that occurs in the managed hub to initiate the discovery phase of the invention.

FIG. 10 is a flow chart of the processing that occurs in the interconnect device during the discovery phase of the invention.

FIG. 11 is a flow chart of the processing that occurs in the managed hub during the discovery phase of the invention in response to the receipt of a discovery response frame.

FIG. 12 is a flow chart of the processing that occurs in the managed hub during the detection phase of the invention.

FIG. 13 is a flow chart of the processing that occurs in an interconnect device during the prevention phase of this invention.

FIG. 14 is a flow chart of the processing that occurs in the managed hub during the hub enable phase of the invention.

FIG. 15 is a flow chart of the processing that occurs in the interconnect devices in response to the receipt of a security clear condition frame.

FIG. 16 is an example of the implementation of the invention in a campus LAN environment.

FIG. 17 is an example of the data flows corresponding to the example implementation in a campus LAN environment.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The preferred embodiment of this invention uses the SNMP network management protocol, since SNMP is the most prevalent network management protocol in the industry and is the most widely deployed in campus networks. It should be noted that the concepts in this invention related to network management could also be applied to other network management protocols such as CMIP or SNA.

FIG. 1 illustrates a typical campus network environment in which the present invention can be implemented. As shown in the figure, the campus network 10 contains interconnect devices, such as router 12, router 14, token ring switch 16, bridge 18, managed hubs 20, 22, 24, network management station 26, workstation 28 and file server 30.

The managed hubs and interconnect devices depicted in FIG. 1 are considered SNMP managed devices. The typical component block diagram for an SNMP managed device is illustrated in FIG. 2. A typical managed device is an embedded system that includes a system bus 50, random access memory (RAM) 52, NVRAM 54 to store configuration information, FLASH EPROM 56 to store the operational and boot-up code, a processor or CPU 58 to execute the code instructions, and a media access control (MAC) chip 66 that connects the device to the network 10. FIG. 2 also shows operational code 60, TCP/IP protocol stack 62 and SNMP agent code 64. In most instances, the operational code and the frame processing code execute in FLASH memory **56** or in RAM **52**. The code that implements several phases in this invention is included as a part of the operational code (microcode or firmware) of the managed device. The MAC chip **66** copies the frames corresponding to the different phases into RAM **52** and notifies the processor **58**, usually via an interrupt, that a frame is ready for processing. The operational code **60** handles the interrupt and processes the frame.

FIG. 3 illustrates the typical component block diagram for a network management station such as that indicated by reference numeral **26** in FIG. 1. The network management station includes a processor **70**, with a system bus **90** to which RAM **72**, direct access storage device (DASD) **74**, other peripherals **76**, display monitor **78**, keyboard **80**, mouse **82** and network interface card **84** are connected.

FIGS. 4A–4C show the general frame formats for Ethernet and token ring frames. The LAN security feature group address is placed in the destination address (DA) field of the discovery request, security breach detected and security clear condition (optionally) frames as discussed more fully below. The data field portion of each frame is used to pass the additional information related to this security feature.

The following describes the information that is included in the data fields of the Ethernet and token ring frame types to represent the different frames that are specific to the preferred embodiment of the invention.

The discovery request frame shown in FIG. 5A is sent to the LAN security feature group address and the data field includes a one byte field which indicates that the frame type (frame type identifier x '01') is a discovery request frame. The time stamp field is the system time value when the discovery request frame is transmitted. It is used to correlate the discovery response frame with the discovery request frame.

The discovery response frame shown in FIG. 5B is sent to the individual MAC address of the managed hub that initiated the request. The data field in this frame includes a one byte field which indicates that the frame type is a discovery response frame (frame type identifier x '02'), and also contains the MAC address of the LAN interconnect device sending the frame, a description of the LAN interconnect device (e.g., IBM 8272 Model 108 Token Ring Switch), and a time stamp that is used to correlate the discovery response frame with the discovery request frame.

The security breach detected frame shown in FIG. 5C is sent to the LAN security feature group address and the data field includes a one byte field which indicates that the frame type is a security breach detected frame (frame type identifier x '03') and contains the MAC address that was detected as the security intruder. Other fields of this frame contain the module number and port number where the security breach was detected and the system time when the security breach was detected. When the time stamp value is used in combination with the intruding MAC address and module and port numbers, it forms an intrusion identifier as will be referred to subsequently. Following the time stamp are device field length indicating the length of the field that follows and address fields. The address field contains the list of addresses that have processed and forwarded the security breach detected frame. It starts with the originating MAC address of the managed hub. Each successive interconnect device that receives the frame, appends its MAC address to the end of this field and updates the device field length before it forwards the frame. It provides an audit trail or path that the security breach detected frame followed throughout the network. A network management station can monitor the progress of the security breach detected frame through information in the trap frames that it receives.

The filter set frame shown in FIG. 5D is sent to the individual MAC address of the managed hub that initiated the security intrusion condition. The data field includes a one byte field which indicates that the frame type is a filt er set frame (frame type identifier x '04') and contains the MAC address of the LAN interconnect device sending the frame. Other fields in this frame are the MAC address of the detected intrusion, the module and port number of the managed hub where the security intrusion was detected, and the time stamp representing the system time when the security breach was detected.

The security clear condition frame shown in FIG. 5E can be sent to the LAN security feature group address or to the individual MAC address of a LAN interconnect device. The data field includes a one byte field which indicates that the frame type is a security clear condition frame (frame type identifier x '05') and contains the intruding MAC address to remove as a filter.

Trap frames are sent to the network management station at various times depending upon the phase of the invention that is being performed. All trap frames have the same basic format with the information in each trap frame varying according to the phase.

In the discovery phase, traps are sent as a result of the managed hub deleting an interconnect device from the list of devices that are in the security domain of interconnect devices. The discovery trap frame contains the trap identifier (x '01'), the MAC address of the interconnect device and device description. This trap indicates that an interconnect device was removed from a managed hub interconnect device list because it did not respond to the managed hub with a discovery response frame within the allotted time period of the discovery window.

Traps sent in the detection phase indicate that the managed hub detected an intrusion on one of the hub ports. Information in this trap frame includes trap identifier (x '02'), the MAC address of the intruding device, the module and port number of the detected intrusion, and the time when the security intrusion was detected.

Traps sent in the prevention phase indicate that the interconnect device has completed the processing of a received security breach detected frame. This trap frame contains the trap identifier (x '03'), the MAC address of the intruding device, the module and port number of the detected intrusion, the time when the security breach was detected and a variable length address field. This last field contains a list of MAC addresses for all the devices that have processed the security breach detected frame. This information provides to the network management station the path that the security breach detected frame followed through the network.

Traps sent in the hub enable phase indicate that the managed hub has reenabled a hub port as a result of receiving filter set frames from all of the interconnect devices in the discovered security domain, i.e., all the discovered interconnect devices. This trap frame contains the trap identifier (x '04'), the MAC address of the intruding device, the module and port number of the detected intrusion, and the time when the security breach was detected.

For token ring networks, the information in the trap frames can be included in frames addressed to the functional address of the LAN manager. The LAN management frame format and defined functional address are specified in the IBM Token Ring Network Architecture (SC30-3374-02) publication.

For managed hubs, the authorized address list (AAL) controls which MAC addresses are allowed to connect to specified ports. Each entry in the AAL consists of two fields: port number and authorized address. The port number identifies a specific port on the hub; the authorized address field specifies the address or addresses that are allowed to connect to the port.

The AAL can be built by the network administrator as part of the configuration of the managed hub. The network administrator identifies the addresses that are allowed to connect to specific ports on the hub. After the initial configuration, the AAL can be updated in several ways. The network management station can add or delete entries in the AAL by sending SNMP management frames. Since most managed hubs provide a Telnet interface into the device to change configuration parameters, a Telnet session could be used to add or delete entries in the AAL. Also, since most managed hubs provide for the attachment of a local console over an RS232 serial port connection which can be used to change configuration parameters, a local console session can be used to add or delete entries in the AAL.

Alternatively, the AAL can be built dynamically through a learning process. Most managed hubs provide a mechanism in the hardware to capture the addresses of the stations that are attached to the ports of a hub. These learned addresses can be provided to the network management station as those stations authorized to access the hub. These learned addresses are then used as the AAL for the managed hub.

The discovery phase is initiated by each managed hub in the campus network. Its purpose is to determine the LAN interconnect devices in the campus LAN that support the LAN security feature. Each managed hub periodically transmits a discovery frame (FIG. 5A) to the LAN security feature group address. The managed hub then uses the information in the response frame (FIG. 5B) to build and maintain a list of all of the devices that support the LAN security feature. This list is referred to as the Interconnect Device List (ICD). The addresses in this list are used in the hub enable phase to correlate the reception of the filter set frame (FIG. 5D) with entries in the list. The managed hubs typically store these ICD lists in management information base (MIB) tables where they can be retrieved, upon request, from a network management station.

The discovery phase can also be used to provide an integrity check on the ICD list of devices supporting the LAN security feature. By periodically transmitting the discovery frame (FIG. 5A) to the LAN security feature group address, checks can then be made to ensure that all of the devices are still in the ICD security list. If any discrepancies are detected, e.g., if a station is removed from the list or added to the list, then an SNMP trap is sent to the network management station. This notification alerts the network administrator that a potential security exposure exists in the campus network. FIG. 6 illustrates the structure of the ICD list along with the information stored in the list for each discovered interconnect device. Other lists that are built and maintained in the detection and prevention phases are the Breach List shown in FIG. 7 and the Intrusion List shown in FIG. 8. Their use will be explained below in the description of the detection and prevention phases.

The detection phase operates at the managed hub level. Each port on the managed hub can be configured to hold one or more MAC addresses of users that are authorized to access the network. The managed hubs can be 10 or 100 Mbps Ethernet or token ring hubs. Current hub chipsets provide the capability to determine the last source MAC address that is seen on a port. When a station attempts to connect to a network, either by inserting into the token ring or by establishing a link state with an Ethernet hub, the last source address seen on the port is compared to the authorized list of MAC addresses that has been defined for this port. If the address is authorized then normal network operations occur. If the address is not authorized, then the managed hub performs the following actions:

1. disables the port;
2. sends an SNMP trap frame to the network management station;
3. sends an alert frame to the functional address of the LAN Manager (token ring); and
4. transmits a security breach detected frame (FIG. 5C) to the LAN security feature group address.

Additional variables in the SNMP trap provide information about the point of intrusion: e.g. the module id (in the case of stackable hubs), the port number, the network number (in cases where hubs have multiple backplanes), and a time stamp (sysUpTime) of when the intrusion was detected. SysUpTime is an SNMP MIB variable that represents the time (units of 0.01s) since the network management portion of the system was last re-initialized.

Some managed hubs support multiple backplanes or networks. In this case, the security breach detected frame is transmitted on all of the active backplanes/networks within the hub.

The well known group address needs to be defined and reserved for LAN security functions. The security breach detected frame (FIG. 5C) containing the MAC address of the station that intruded into the network is sent to the LAN security feature group address.

The prevention phase spans the network. Each interconnect device in the campus network is configured to copy frames addressed to the LAN security feature group address. Upon a security intrusion, the network interconnect devices copy the security breach detected frame (FIG. 5C) and perform the following functions:

1. set filters based on the intruder's MAC address.
2. transmit a security breach detected frame (FIG. 5C) to the LAN security feature group address.
3. send an SNMP trap frame to the network management station.
4. send an alert frame to the functional address of the LAN manager (token ring).
5. transmit filter set frame (FIG. 5D) to the MAC address of the hub that initiated the security breach process.

Setting filters by the network interconnect device prevents intrusion attempts with this MAC address originating elsewhere in the campus network from flowing through this interconnect device. This protects an enterprise's data on this segment of the network from any attacks via the intruder's MAC address.

The interconnect device extracts the intrusion identifier information from the security breach detected frame. If this is the first time the interconnect device has received a security breach detected frame with this intrusion identifier, the interconnect device adds this information to the Intrusion List, then checks to ensure the filter has been set for the intruding MAC address and resets, if required. The interconnect device then transmits the security breach detected frame on all ports except the port on which the security breach detected frame was received.

Sending the trap frame indicates that the filter has been set as a result of receiving the security breach detected frame. Likewise, sending the alert frame indicates that the filter has been set as a result of receiving the security breach detected frame.

The hub enable phase operates at the network level. The hub that initiates the security breach process receives the filter set frames from the interconnect devices in the campus network. The hub then waits to receive responses back from all of the interconnect devices that were determined in the discovery phase to be in the campus network. When all the interconnect devices in the network have responded to the hub with the filter set frame, the hub then re-enables the port for use and then sends a TRAP frame back to the network management station indicating that all filters have been set for the intruding MAC address. The network management station can optionally forward this information to a network management application such as IBM Corporation's NetView/390 product via an alert.

The security clear condition phase of this invention provides the capability for a network administrator to manually override, if necessary, one of the filters that has been set in the prevention phase. The network management station could globally clear, i.e., remove a filter from all LAN interconnect devices by transmitting the security clear condition frame (FIG. 5E) to the LAN security feature group address. The network management station could selectively clear, i.e., remove a filter from a LAN interconnect device by transmitting the security clear condition frame to the MAC address of the specific LAN interconnect device.

FIGS. 9–15 are flow charts that illustrate the processing that occurs in the managed hub and in the interconnect devices during each phase of the invention. The code to implement the discovery phase of this invention runs within the managed hub and interconnect device as event driven threads within the real time OS embedded system. The flows in FIG. 9 depict the processing that occurs in the managed hub to initiate each discovery phase. This task manages the initialization and update of the Interconnect Device List and timing of the next iteration of the discovery phase. The following briefly describes each logic block in the figure.

Step **100**: Entry to this task can be caused by a power on and/or reset. This would be one of many tasks that would run in response to this event.

Step **101**: There are two lists, a period, a window, and two flags that are used by the managed hub in this invention. The ICD (Interconnect Device) List contains information on the devices found during the discovery phase. The Breach List contains information on intrusions recognized by the hub and in the process of being secured. The period is the time between discovery phases. The window is the time between when a discovery phase is initiated and when an Interconnect Device must respond before being assumed inaccessible due to network or device outage. One flag is an indication that initialization has completed. The other flag is an indication that the security feature is enabled. The lists, the period, the window and the enabled flag may be cleared or loaded from persistent memory. The initialized flag is set to True.

Step **102**: Test for whether the security feature is enabled.

Step **103**: Each managed hub maintains a MIB variable that is called SysUpTime. This is used as a time stamp for sẽcurity feature frames.

Step **104**: The discovery frame is built with the data field containing the type of the frame—Request.

Step **105**: The frame is sent to the LAN security feature group address.

Step **106**: The discovery phase is initiated periodically as an integrity check on the security feature coverage within the network. The period is adjustable to reflect variable path lengths or round-trip-times between a managed hub and interconnect devices. The period can be set via SNMP. The longer t he period, the less the integrity of the network coverage. The shorter the period, the higher the traffic rate required for the security feature.

Step **107**: Set a pointer to the he ad of the list of ICD (Interconnect Device) List items. The pointer may point to an item or nothing if there are not items in the list. (The ICD List is a list of the interconnect devices that responded in a previous discovery phase). This part of the task is to update the Interconnect Device List by updating items as appropriate or deleting them as necessary.

Step **108**: Does the pointer point to an item in the list or does it point beyond the end of the list?

Step **109**: Each ICD List item has a time stamp from the last discovery response frame received from the device.

Step **110**: Is the time for the item in the ICD List later than current time?

Step **111**: If yes, the managed hub has reset or rolled over its SysUpTime since the last response from the ICD. Set the time in the ICD List item to current time.

Step **112**: Is the difference between the current time and the last response time from the item greater than the discovery window?

Step **113**: Assume the device is inaccessible due to network or device outage and purge the item from the ICD List. Also, decrement the outstanding filter set count on all the Breach List items.

Step **114**: If there is a network management station (NMS) that is receiving traps from the managed hub and the traps are enabled, send a trap indicating that the interconnect device is no longer accessible. If there is an LNM for OS/2 station available and traps are enabled, send a trap to the LNM for OS/2 station.

Step **115**: Move the ICD List pointer to the next item or to the end of the list if no more entries exist. This is for stepping through the entire list of ICD items.

Step **116**: End the task and return to the embedded system OS.

Step **117**: Enter this task due to a timer driven interrupt (set in step **106**).

The flows in FIG. **10** depict the processing that occurs in the interconnect devices during each iteration of the discovery phase. This task responds to the receipt of a discovery request frame by sending a discovery response frame. The following briefly describes each logic block in the figure.

Step **143**: The task is initiated by the receipt of a discovery request frame.

Step **144**: A check is made for whether the security feature is enabled. This determines if any additional processing is required.

Step **145**: The source MAC address and time stamp are extracted for building the response.

Step **146**: The discovery response frame is built using the information from the discovery request frame that was just received.

Step **147**: The frame is sent to the originating managed hub.

Step **148**: The task ends, returning control to the embedded OS.

The flows in FIG. **11** depict the processing that occurs in the managed hub in response to the receipt of a discovery response frame. This task maintains the state of this iteration of the discovery phase. The following briefly describes each logic block in the figure.

Step 130: The task is initiated in the managed hub by the receipt of a discovery response frame.

Step 131: The interconnect device information is extracted from the frame.

Step 132: The Interconnect Device List is searched for an item with a MAC address matching the source address of the discovery response frame.

Step 133: Has a match been found?

Step 134: If a match is found, update the last response time in the ICD List item with the time stamp that was extracted from the discovery response frame.

Step 135: If there is no match, assume that the device is not in the list because of either network/device outages or the device has just started utilizing the security feature. It is necessary to determine if the discovery window is still large enough. The round-trip-time is calculated, and multiplied by 2 to derive a potential discovery window. If this is larger than the current discovery window, the discovery window needs to be changed.

Step 136: Change the discovery window.

Step 137: Create a new Interconnect Device List item using the source address from the discovery response frame, the device description from the frame, and the time stamp from the frame. Add it to the list.

Step 138: Optionally send a trap to the network management station(s) and if this is a token ring, to the LAN manager functional address.

Step 139: The task ends, returning control to the embeded OS.

The code to implement the detection phase of this invention runs as a separate task independent from the other tasks in the managed hub. The flows in FIG. 12 depict the processing that occurs during the dispatch of the detection phase task. This task simply checks all the ports in the hub to ensure that the station attached to the port has been authorized to establish a connection on this port. The AAL (Authorized Address List) defines which MAC addresses are allowed to connect to specific ports on the hub. The following briefly describes each logic block in the figure.

Step 200: This is the entry point for the detection phase task. Processing starts at port number 1 in the hub and continues until all of the ports in the hub have been processed.

Step 210: This step checks if a station is attached to the port in the hub. If a station is attached, then an address exists for the port. If an address is detected for the port (i.e., a station is attached to the port), then processing continues with step 220. if there is no address detected for this port (i.e., no station is attached), then processing continues with step 230.

Step 220: A check is made here to ensure that the address that has been detected on this port is in the list of authorized addresses. If the address detected on the port is authorized, then continue processing at step 230. If the address detected on the port is not in the authorized list, then processing continues at step 250.

Step 230: A check is made here to see if all of the ports in the hub have been processed. If all of the ports have been processed, then processing resumes at step 200 with the processing of port number 1. if this was not the last port and there are more ports to process, then processing continues at step 240.

Step 240: In this step, the next port in the hub is set up to be processed. Processing then continues at step 210.

Step 250: In this step a check is made to see if the port is already disabled. If the port is already disabled, then the port/network is already secure from intruders on this port. if the port is already disabled, then processing continues at step 230. If the port is enabled, processing then continues at step 260.

Step 260: In this step, the port is disabled. Processing then continues at step 265.

Step 265: In this step, an entry is added to the Breach List containing the following: MAC address that was detected as the intruder, the module and port number where the intrusion was detected, the time (sysUpTime) when the security breach was detected, and the outstanding filter set count which is set to the number of entries in the ICD list. Processing then continues at step 270.

Step 270: In this step, the security breach detected frame is transmitted on all network segments of the hub. The info field of the security breach detected frame includes the following: MAC Address of the intruder, module number, port number, time stamp (sysUpTime), the device field length initialized to 6 (bytes), the 6 byte MAC address of the managed hub. Processing then continues at step 280.

Step 280: In this step, a trap frame is optionally sent to the network management station. The trap frame includes the following information:

(a) trap identifier x '02';

This indicates that the managed hub detected in intrusion on one of the hub ports.

(b) MAC address of the intruding device;

(c) module number of the detected intrusion;

(d) port number of the detected intrusion;

(e) time when the security breach was detected;

Processing then continues at step 290.

Step 290: In this step, a check is made to see if this invention has been implemented in a token ring network. The token ring architecture defines a special functional address that is used by LAN management stations. Functional addresses are only used in token ring environments. If the invention is implemented in a token ring network, processing then continues at step 295. If the invention is implemented in a non-token ring network, processing then continues at step 230.

Step 295: in this step, a frame is sent to the functional address of the LAN manager with the information from step 280. Processing then continues at step 230.

FIG. 13 depicts the flows for the prevention phase of the invention. The prevention phase is implemented in the interconnect devices of the network. The following briefly describe each logic block in the figure.

Step 300: The processing is initiated when the interconnect device receives a frame from the network. The interconnect device copies the frame and saves the port number that the frame was received on. Processing then continues at step 302.

Step 302: In this step, the frame that wa s copied in step 300 is interrogated and a check is made to determine if the destination address of the frame is equal to the LAN security feature group address. if the received frame is addressed to the LAN security feature group address, then processing continues at step 306. Otherwise, the frame is of some other type and the processing continues with step 304.

Step 304: This step is encountered for all frame types other than the LAN security feature. The normal frame processing code of the interconnect device runs here.

Step 306: In this step, the intrusion identifier information is copied from the frame. The intrusion identifier consists of the following information:

(a) MAC address of the intruder;

(b) module number;

(c) port number;

(d) time stamp;

Processing then continues at step **308**.

Step **308**: In this step, a check is made to determine if the intrusion identifier is already in the Intrusion List of this interconnect device. If yes, processing then continues at step **316**. If no, processing then continues at step **312**.

Step **312**: In this step, the intrusion identifier information is added to the Intrusion List. Processing then continues at step **316**.

Step **316**: In this step, the current port of the interconnect device is set to port number **1**. Processing then continues at step **318**.

Step **318**: In this step, a check is made to determine if the intruding MAC address is already filtered on the current port. If yes, processing then continues at step **322**. If no, processing then continues at step **320**.

Step **320**: In this step, a filter is set for the intruding MAC address on the current port. Processing then continues at step **322**.

Step **322**: In this step a check is made to determine if the filter processing has been applied to all of the ports in the interconnect device. If all of the ports have been processed, processing then continues at step **326**. If there are more ports to process, processing then continues at step **324**.

Step **324**: In this step, the current port is set to the next port in the interconnect device. Processing then continues at step **318**.

Step **326**: In this step, the security breach detected frame is propagated throughout the network. The interconnect device transmits the security breach detected frame on all ports other than the port the original frame was received on. (Reference step **300** where it is determined which port the frame was received on). Before transmitting the security breach detected frame, the ICD appends its MAC address to the addresses field of the frame and increments the device field length field of the frame by 6. This provides the audit trail or the path information for the security breach detected frame. Processing then continues at step **332**.

Step **332**: In this step, the interconnect device transmits the filter set frame to the originator of the security breach detected frame. The originator is determined by extracting the source address from the frame that was copied in step **306**. Processing then continues at step **334**.

Step **334**: In this step, a trap frame is sent to the network management station. The trap frame includes the following information:

(a) trap identifier x '03';

This indicates that the interconnect device has completed the processing of a received security breach detected frame.

(b) MAC address of the intruding device;

(c) module number of the detected intrusion;

(d) port number of the detected intrusion;

(e) time when the security breach was detected;

(f) addresses field;

This is a variable length field that contains a list of all of the devices that have processed the security breach detected frame. This information provides to the network management station the path that the security breach detected frame followed throughout the network.

Processing then continues at step **336**.

Step **336**: In this step, a check is made to see if this invention has been implemented in a token ring network. The token ring architecture defines a special functional address that is used for LAN management stations. Functional addresses are only used in token ring environments. If the invention is implemented in a token ring network, processing then continues at step **338**. If the invention is implemented in a non-token ring network, processing then continues at step **340**.

Step **338**: In this step, a frame containing the same information in the trap frame in step **334** is sent to the functional address of the LAN manager. Processing then continues at step **340**.

Step **340**: In this step, processing resumes again at step **300**.

The code to implement the hub enable phase of this invention runs within the managed hub as event driven threads within the real time OS embedded system. The flows in FIG. **14** depict the processing that occurs in the managed hub in response to receipt of each filter set frame. The task maintains the necessary lists of interconnect devices and breaches to complete the hub enable phase for each breach. The following briefly describes each logic block in the figure.

Step **400**: The task is initiated in the managed hub by the receipt of a filter set frame.

Step **401**: Get the source address of the frame for finding the associated ICD List item.

Step **402**: The Interconnect Device List is scanned for an item with the same MAC address as the source address of the frame.

Step **403**: Was a match found? If not, assume that the interconnect device is no longer accessible.

Step **404**: If a match is found, decrement the outstanding breach response count in ICD List item by 1. This provides an up-to-date count of outstanding responses for each ICD.

Step **405**: Extract intrusion identifier information from the frame.

Step **406**: Scan the Breach List for an item with a matching intrusion identifier.

Step **407**: Match found?

Step **408**: If a match is found, decrement the outstanding filter set count by 1 in the matching Breach List item.

Step **409**: Have all interconnect devices responded? Are all filters set?

Step **410**: Since the intruder is now being filtered and has been removed from the network, remove the Breach List item.

Step **411**: If there is a listening network management station(s), send a trap. If this is a token ring, send an alert to the LAN manager functional address.

Step **412**: Optionally reenable the port. This is a policy decision. It may also reflect the likelihood of the intruder still attempting to intrude via this same port.

Step **413**: End the task and return control to the embedded OS.

The code to implement the security clear condition phase of this invention runs within the interconnect devices as event driven threads within the real time OS embedded system. The flows in FIG. **15** define the processing that occurs in the interconnect devices in response to receipt of each security clear condition frame. The task updates the Intruder List of breaches and completes the security clear condition phase for each breach. The following briefly describes each logic block in the figure.

Step **500**: The task is initiated in the interconnect device by the receipt of a security clear condition frame from a network management station.

Step **501**: Extract the intruder MAC address from the security clear condition frame.

Step **502**: Search the Intrusion List for a matching MAC address.

Step 503: Is there a match?

Step 504: If there is a match, remove the item from the Intrusion List.

Step 505: Remove filter for the intruding MAC address.

Step 506: End the task and return control to the embedded OS.

Two examples are given below to illustrate the actions that are performed by the managed hub and interconnect devices in an implementation of this invention in an operational campus environment. Referring again to FIG. 1, there is depicted a workstation 28, attached to an Ethernet hub 24, that is attempting to gain unauthorized access to a file server 30 that is located on a token ring segment. The security intrusion is detected by the managed Ethernet hub 24, since the MAC address of the workstation 28 is not authorized for this port in the hub. The managed hub 24 then disables the port and transmits the security breach detected frame to the LAN interconnect device 14 on this segment, which, in turn, forwards the security breach detected frame to LAN interconnect devices 12, 16 that are attached to subnet 3 and subnet 4, respectively. LAN interconnect device 12, in turn, forwards the security breach detected frame to LAN interconnect device 18. The LAN interconnect devices 12, 14, 16, 18 set filters on all ports in the device to prevent frames with the intruding MAC address from flowing through the interconnect device.

More specifically, the managed hub 24 disables the port and transmits the security breach detected frame to router 14. The managed hub 24 also sends a trap frame to the management station 26. Router 14 applies the intruder's MAC address as a filter on all of its ports and forwards the security breach detected frame on all of its ports, except the port the security breach detected frame was received on. Router 14 then sends a trap to the network management station 26 and sends a filter set frame back to the managed hub 24. Router 12 and the token ring switch 16 also receive the security breach detected frame and perform the same processing operations as defined above for router 14. The bridge 18 receives the security breach detected frame and performs the same processing operations as done by router 14. The managed hub 24 now correlates all of the received filter set frames with the interconnect devices 12, 14, 16, 18 that were discovered via the discovery request/response frames and reenables the port. The managed hub 24 then sends a trap to the management station 26 to indicate that the intruder's port has been reenabled.

As a practical example of the implementation of this invention in a campus LAN environment, FIG. 16 depicts a university setting in which there is a managed hub on each floor of the buildings in a campus network. The network infrastructure consists of a pair of Ethernet switches attached to a campus backbone. Each Ethernet switch is also attached to a plurality of Ethernet managed hubs (one on each floor in each building). The figure shows a student dormitory that is attached to the same network that runs the university administration applications. There are obvious security concerns about students accessing the proprietary administrative information (i.e., grades, transcripts, payroll, accounts receivable/payable, etc.).

An intruder trying to access the network via one of the managed hub ports in the dormitory is stopped at the port of entry to the network and further access to the campus network is prevented by having the intruder's MAC address filtered on all LAN interconnect devices. The symbols containing a "B" in FIG. 16 indicate the points in the campus network where frames with the intruding MAC address are blocked from access to LAN segments by the setting of

filters. The data flows corresponding to the example are shown in FIG. 17 and are self-explanatory.

For simplicity, this invention has used the term managed hub to refer to traditional token ring and Ethernet port concentration devices (e.g., IBM 8238, IBM 8224, IBM 8225, IBM 8250, IBM 8260). In reality, the functions of the managed hub can be extended to LAN switches (both token ring and Ethernet) where dedicated stations could be attached directly to the switch port. LAN switches would have to add the functionality of authorizing a set of MAC addresses that could attach to a switch port and detecting any unauthorized accesses to the switch port.

To describe the key aspects of this LAN security invention, it was easiest to illustrate with an implementation using managed hubs. In reality, many large enterprises use a combination of both managed hubs and unmanaged hubs throughout their networks. This invention is readily extendible and the security detection mechanism can easily be integrated into the function of a LAN bridge. The bridge would keep the list of authorized addresses for a given LAN segment where access to the LAN is via low cost unmanaged concentrators. The bridge would then detect any new addresses on the LAN segment and compare the addresses against the authorized list. If an unauthorized address was detected, the bridge would then set up filters for the intruding MAC address, and transmit the security breach detected frame to the other interconnect devices attached to the campus network. In this case, the intruder would be isolated to the LAN segment where the intrusion was first detected. This example shows that the composite function of the managed hub could be integrated into a LAN bridge and the bridge could control the security access for a large segment consisting of unmanaged concentrators.

Another special use of this invention involves the tasks of a network administrator. A key day-to-day task for most network administrators falls into the category of moves, adds, and changes to network configuration. In this invention, the network management station has complete awareness of all of the authorized users throughout the campus network. In the event that a security breach is detected, in the special case where an authorized user is trying to gain access through an unauthorized port, the network management station could detect this situation and automatically take the appropriate actions (i.e., remove filters from the interconnect devices since this is an authorized user). This type of action would assist administrators that work in dynamic environments where there are frequent moves, adds and changes.

The preferred embodiment of the invention has relied upon the detection of unauthorized MAC addresses by the managed hub. It can easily be modified to apply to the network layer (layer 3) or higher layers, in the Open System Interconnection (OSI) protocol stack and work with such well known network protocols as TCP/IP, IPX, HTTP, AppleTalk, DECnet and NETBIOS among others.

Currently, many LAN switches have custom application specific integrated circuits (ASICs) that are designed to detect or recognize frame patterns in hardware. These LAN switches use this frame type recognition capability primarily for frame forwarding based on the IP address and for placing switch ports in a virtual LAN (VLAN). In order to provide security protection at the network layer, it will be clear to one skilled in the art that the authorized address list (AAL) described herein can be extended to include IP addresses. The so-modified AAL, coupled with the LAN switch capability to detect IP addresses in a frame will enable implementation of the detection and prevention phases to support

IP addresses. In the detection phase, the ASIC-based LAN switch can be used to obtain the IP address that is connected to a port. The detected IP address would then be compared to the authorized IP addresses in the AAL. If an unauthorized IP address is detected, the invention works as previously described with the disabling of the port and the transmission of the security breach detected frame. In the prevention phase, the interconnect devices are notified of intruding IP addresses and then apply filters for the intruding IP address.

The present invention can also be modified to operate at the application layer (layer 7) of the OSI protocol stack. Currently, several commercially available LAN switches, such as the model 8273 and model 8274 LAN switches available from IBM Corporation, provide a capability for a user-defined policy for creating a VLAN. This user-defined policy enables one to specify an offset into a frame and a value (pattern) to be used to identify the frame. Once the user-defined policy has been defined, the switch ASIC detects all frames matching the specified pattern and places them into a specific VLAN. Since the custom ASIC recognizes the user-defined pattern, it can be programmed to recognize portions of a frame that identify a specific application. This application pattern can then be used as the detection criteria in the invention and thus provide application layer security.

The present invention can be modified further to provide additional security by encryption of the data fields in the frames that are used to implement the inventive concepts described above. One of the most widely known and recognized encryption algorithms is the Data Encryption Standard (DES). The implementation of DES or other encryption algorithm to encrypt the data fields of frames described in this invention can ensure the privacy and integrity of the communication between managed hubs, interconnect devices and network management stations. Security protocols such as Secure Sockets Layer (SSL) utilizing public key encryption techniques are becoming standardized and can be used to further enhance the invention described herein.

While the invention has been particularly shown and described with reference to the particular embodiments thereof, it will be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention.

Having thus described our invention, what we claim and desire to secure as Letters Patent is as follows:

1. A method for providing security against intrusion in a managed device of a computer network having at least one interconnect device, said method comprising the steps of:

discovering each of said interconnect devices that is enabled to provide network security;

detecting an unauthorized address on a first port of said managed device and disabling said first port;

notifying each of said security-enabled interconnect devices that the unauthorized address has been detected on said first port; and

reenabling said first port after each of said security-enabled interconnect devices has notified said managed device that a filter has been set to prevent frames with the unauthorized address from flowing through said each security-enabled interconnect device.

2. The method for providing security against intrusion of claim 1 wherein said managed device is a managed hub.

3. The method for providing security against intrusion of claim 1 wherein said managed device is a switch.

4. The method for providing security against intrusion of claim 1 wherein said computer network includes a local area network.

5. The method for providing security against intrusion of claim 1 further comprising the steps of building and maintaining an authorized address list of addresses that are allowed to connect to each port in said managed device.

6. The method for providing security against intrusion of claim 5 wherein each entry in said authorized address list includes a port number and an authorized address.

7. The method for providing security against intrusion of claim 1 wherein said discovering step includes the steps of:

transmitting a discovery request frame, said discovery request frame having a security feature group address;

receiving a discovery response frame from each of said security-enabled interconnect devices;

building and maintaining an interconnect device list of said security-enabled interconnect devices that transmitted said discovery response frame back to said managed device.

8. The method for providing security against intrusion of claim 7 wherein each entry in said interconnect device list includes an address of the security-enabled interconnect device that sent the discovery response frame and a time stamp extracted from said discovery response frame.

9. The method for providing security against intrusion of claim 6 wherein said detecting step includes the steps of:

comparing, for each port, a source address of a station attempting to connect to said port with the authorized address list of addresses for said port and determining whether said source address is on said authorized address list.

10. The method for providing security against intrusion of claim 7 wherein following said disabling step said method further includes:

sending a trap frame to a network management station indicating that an intrusion has been detected on said first port; and

transmitting a security breach detected frame having said security feature group address to said security-enabled interconnect devices that have entries in said interconnect device list.

11. The method for providing security against intrusion of claim 10 wherein said security breach detected frame includes a source address of an unauthorized station, the port number at which the intrusion occurred, and a time stamp representing the time at which the unauthorized station was detected.

12. The method for providing security against intrusion of claim 11 wherein each of said security-enabled interconnect devices transmits a filter set frame to said managed device that includes the address of said each security-enabled interconnect device sending said filter set frame, the source address of said unauthorized station, the port number at which the intrusion occurred, and a time stamp representing the time at which the unauthorized station was detected.

13. The method for providing security against intrusion of claim 1 wherein following said reenabling step said managed device sends a trap frame to a network management station indicating that said filtering step has been completed.

14. An apparatus for providing security against intrusion in a managed device of a computer network having at least one interconnect device, said apparatus comprising:

means for discovering each of said interconnect devices that is enabled to provide network security;

means for detecting an unauthorized address on a first port of said managed device and means for disabling said first port;

means for notifying each of said security-enabled interconnect devices that the unauthorized address has been detected on said first port; and means for reenabling said first port after each of said security-enabled interconnect devices has notified said managed device that a filter has been set to prevent frames having the unauthorized address from flowing through said each security-enabled interconnect device.

15. The apparatus for providing security against intrusion of claim 14 wherein said managed device is a managed hub.

16. The apparatus for providing security against intrusion of claim 14 wherein said managed device is a switch.

17. The apparatus for providing security against intrusion of claim 14 further comprising means for building and maintaining an authorized address list of addresses that are allowed to connect to each port in said managed device.

18. The apparatus for providing security against intrusion of claim 17 wherein each entry in said authorized address list includes a port number and an authorized address.

19. The apparatus for providing security against intrusion of claim 14 wherein said means for discovering includes:

means for transmitting a discovery request frame, said discovery request frame having a security feature group address;

means for receiving a discovery response frame from each of said security-enabled interconnect devices;

means for building and maintaining an interconnect device list of said security-enabled interconnect devices that transmitted said discovery response frame back to said managed device.

20. The apparatus for providing security against intrusion of claim 19 wherein each entry in said interconnect device list includes an address of the security-enabled interconnect device that sent the discovery response frame and a time stamp extracted from said discovery response frame.

21. The apparatus for providing security against intrusion of claim 18 wherein said means for detecting includes:

means for comparing, for each port, a source address of a station attempting to connect to said port with the authorized address list of addresses for said port and means for determining whether said source address is on said authorized address list.

22. The apparatus for providing security against intrusion of claim 19 further including:

means for sending a trap frame to a network management station indicating that an intrusion has been detected on said first port; and

means for transmitting a security breach detected frame having said security feature group address to said security-enabled interconnect devices that have entries in said interconnect device list.

23. The apparatus for providing security against intrusion of claim 22 wherein said security breach detected frame includes a source address of an unauthorized station, the port number at which the intrusion occurred, and a time stamp representing the time at which the unauthorized station was detected.

24. The apparatus for providing security against intrusion of claim 23 wherein each of said security-enabled interconnect devices transmits a filter set frame to said managed device that includes the address of said each security-enabled interconnect device sending said filter set frame, the source address of said unauthorized station, the port number at which the intrusion occurred, and a time stamp representing the time at which the unauthorized station was detected.

25. The apparatus for providing security against intrusion of claim 14 wherein said managed device further comprises means for sending a trap frame to a network management station indicating that said filter has been set at each of said security-enabled interconnect devices.

26. A method for providing security against intrusion in a managed hub of a computer network having at least one interconnect device, said method comprising the steps of:

building and maintaining an authorized address list of addresses that are allowed to connect to each port;

discovering each interconnect device that is enabled to provide network security;

detecting an unauthorized address on a first port and disabling said first port;

notifying each security-enabled interconnect device that the unauthorized address has been detected on said first port; and

reenabling said first port after each security-enabled interconnect device has notified said managed hub that a filter has been set to prevent frames with the unauthorized address from flowing through each security-enabled interconnect device.

27. The method for providing security against intrusion of claim 26 wherein said discovering step includes the steps of:

transmitting a discovery request frame, said discovery request frame having a security feature group address;

receiving a discovery response frame from each security-enabled interconnect device;

building and maintaining an interconnect device list of each security-enabled interconnect device that transmitted said discovery response frame back to said managed hub.

28. The method for providing security against intrusion of claim 27 wherein said detecting step includes the steps of:

comparing, for each port, a source address of a station attempting to connect to said port with an authorized address list of addresses for said port and determining whether said source address is on said authorized address list.

29. The method for providing security against intrusion of claim 27 wherein following said disabling step said method further includes:

sending a trap frame to a network management station indicating that an intrusion has been detected on said first port; and

transmitting a security breach detected frame having said security feature group address to each security-enabled interconnect device that has an entry in said interconnect device list.

30. The method for providing security against intrusion of claim 26 wherein following said reenabling step said managed hub sends a trap frame to a network management station indicating that said filtering step has been completed.

31. An apparatus for providing security against intrusion in a managed hub of a computer network having at least one interconnect device, said apparatus comprising:

means for building and maintaining an authorized address list of addresses that are allowed to connect to each port;

means for discovering each interconnect device that is enabled to provide network security;

means for detecting an unauthorized address on a first port and means for disabling said first port;

means for notifying each security-enabled interconnect device that the unauthorized address has been detected on said first port; and

means for reenabling said first port after each security-enabled interconnect device has notified said managed hub that a filter has been set to prevent frames with the unauthorized address from flowing through each security-enabled interconnect device.

32. The apparatus for providing security against intrusion of claim 31 wherein said means for discovering includes:

means for transmitting a discovery request frame, said discovery request frame having a security feature group address;

means for receiving a discovery response frame from each security-enabled interconnect device;

means for building and maintaining an interconnect device list of each security-enabled interconnect device that transmitted said discovery response frame back to said managed hub.

33. The apparatus for providing security against intrusion of claim 32 wherein said means for detecting includes:

means for comparing, for each port, a source address of a station attempting to connect to said port with an authorized address list of addresses for said port and

means for determining whether said source address is on said authorized address list.

34. The apparatus for providing security against intrusion of claim 32 further including:

means for sending a trap frame to a network management station indicating that an intrusion has been detected on said first port; and

means for transmitting a security breach detected frame having said security feature group address to each security-enabled interconnect device that has an entry in said interconnect device list.

35. The apparatus for providing security against intrusion of claim 31 wherein said managed hub further comprises means for sending a trap frame to a network management station indicating that said filter has been set at each security-enabled interconnect device.

* * * * *

United States Patent [19]
Klaus

[11] **Patent Number: 5,892,903**
[45] **Date of Patent: Apr. 6, 1999**

[54] **METHOD AND APPARATUS FOR DETECTING AND IDENTIFYING SECURITY VULNERABILITIES IN AN OPEN NETWORK COMPUTER COMMUNICATION SYSTEM**

[75] Inventor: **Christopher W. Klaus**, Atlanta, Ga.

[73] Assignee: **Internet Security Systems, Inc.**, Atlanta, Ga.

[21] Appl. No.: **710,162**

[22] Filed: **Sep. 12, 1996**

[51] **Int. Cl.**^{6} **G06F 11/00**
[52] **U.S. Cl.** **395/187.01**; 395/200.57
[58] **Field of Search** 395/187.01, 186, 395/188.01, 200.59, 200.57, 183.04, 200.67, 200.68

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,223,380	9/1980	Antonaccio et al.	364/200
5,204,966	4/1993	Wittenberg et al.	395/188.01
5,309,562	5/1994	Li	395/200
5,311,593	5/1994	Carmi	380/23
5,347,450	9/1994	Nugent	395/200
5,371,852	12/1994	Attanasio et al.	395/200
5,515,508	5/1996	Pettus et al.	395/200.01
5,557,742	9/1996	Smaha et al.	395/186
5,623,601	4/1997	Vu	395/187.01

OTHER PUBLICATIONS

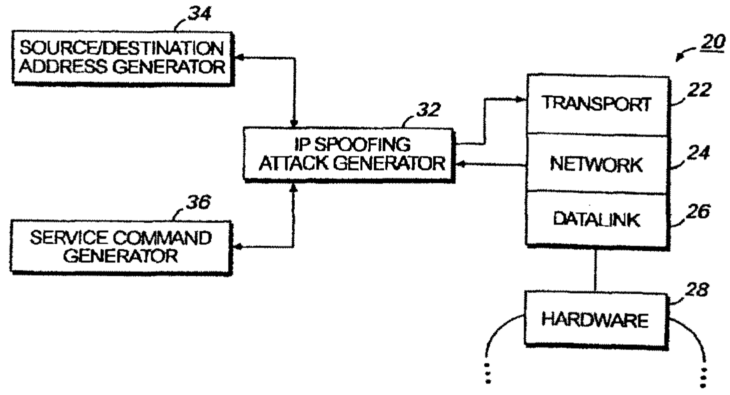
Guha et al., "Network Security via Reverse Engineering of TCP Code: Vulnerability Analysis and Proposeed Solutions", IEEE, pp. 603–610, Mar. 1996.
Garg et al., "High Level Communication Primitives for Concurrent Systems", IEEE, pp. 92–99, 1988.
Hastings et al., "TCP/IP Spoofing Fundamentals", IEEE, pp. 218–224, May 1996.
Snapp, "Signature Analysis and Communication Issues in a Distributed Intrusion Detection System", Master Thesis; University of California, Davis, CA, pp. 1–40, 1991.
Guha et al., "Network Security via Reverse Engineering of TCP Code: Vulnerability Analysis and Proposed Solutions", IEEE, pp. 40–48, Jul. 1997.
Djahandari et al., "An MBone Proxy for an Application Gateway Firewall", IEEE, pp. 72–81, Nov. 1997.
Kim et al., "Implementing a Secure rlogin Environment: A Case Study of Using a Secure Network Layer Protocol", Department of Computer Science, University of Arizona, pp. 1–9, Jun. 1995.
Satyanarayanan, "Integrating Security in a Large Distributed System", Acm Transactions on Computer Systems, vol. 7, No. 3, pp. 47–280, Aug. 1989.

Primary Examiner—Albert Decady
Assistant Examiner—Scott T. Badesman
Attorney, Agent, or Firm—Morris, Manning & Martin, L.L.P.

[57] **ABSTRACT**

A system and method is disclosed for detecting security vulnerabilities in a computer network. The system includes an IP spoofing attack detector, a stealth port service map generator, a source port verifier, source routing verifier, an RPC service detector and a Socks configuration verifier. Each of these verifiers may be operated separately or as a group to detect security vulnerabilities on a network. Each verifier may be programmed to exhaustively test all ports of all computers on a network to detect susceptibility to IP spoofing attacks, access to services with little or no authorization checks or misconfigured routers or Socks servers. The detected vulnerabilities or the location of services having little or no authorization checks may be stored in a table for reference by a network administrator. The service map generated by the stealth service map generator may be used to identify all service ports on a network to facilitate the operation of the other verifiers which send service command messages to service ports to detect their accessibility. A graphic user interface (GUI) may be used to provide input and control by a user to the security verifiers and to present options and display information to the user.

41 Claims, 8 Drawing Sheets


34
SOURCE/DESTINATION ADDRESS GENERATOR
32
IP SPOOFING ATTACK GENERATOR
36
SERVICE COMMAND GENERATOR
20
TRANSPORT
22
NETWORK
24
DATALINK
26
HARDWARE
28

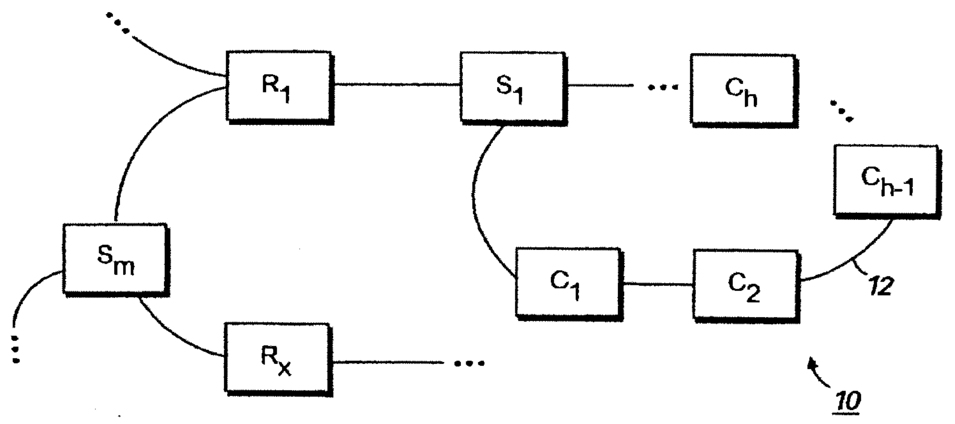

R1
S1
Ch
Ch-1
Sm
C1
C2
Rx
12
10

FIG.1

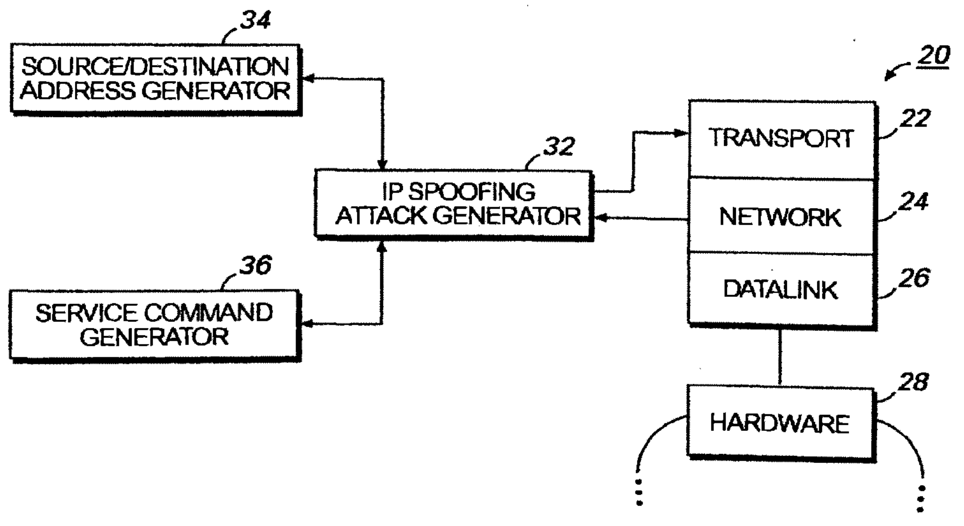
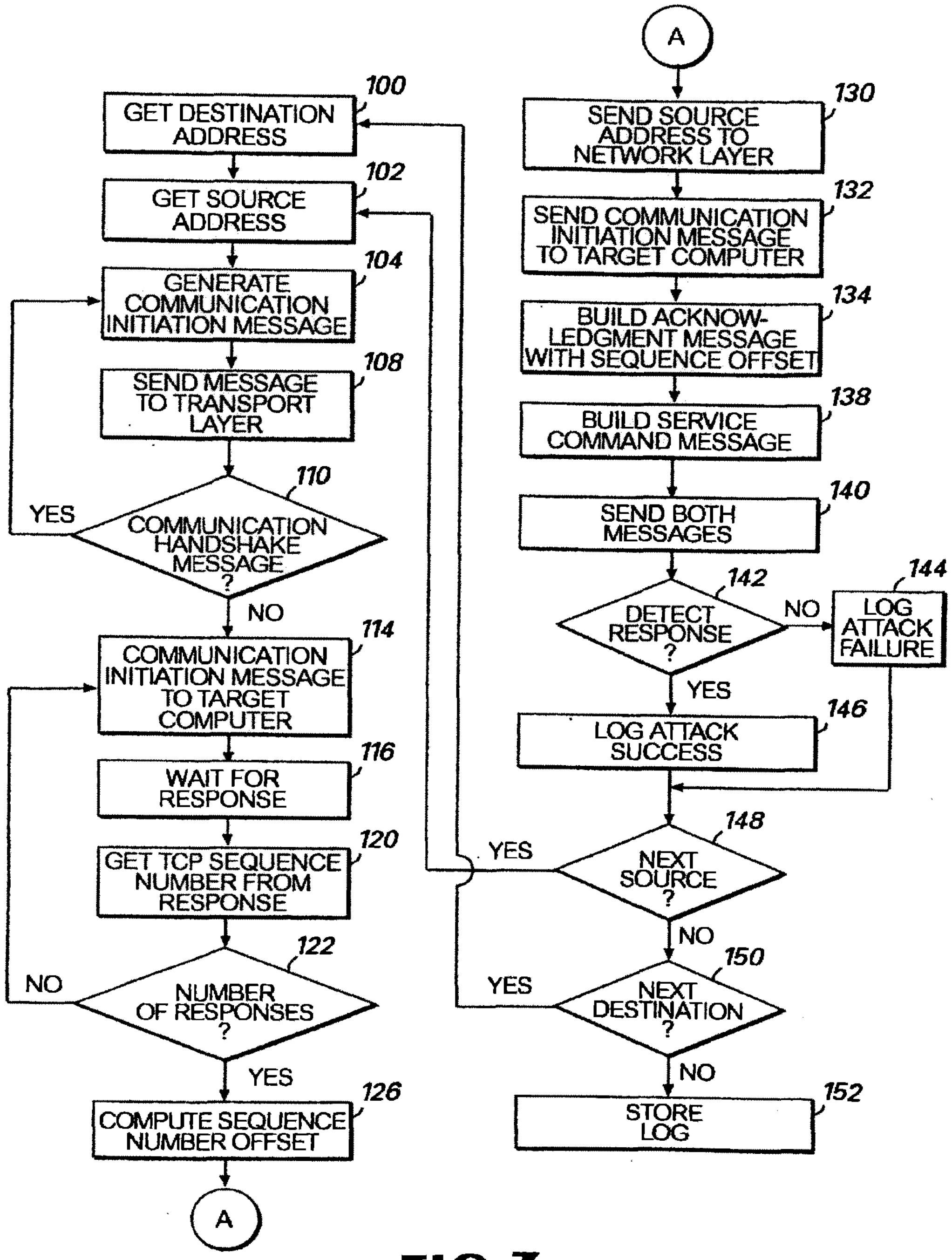

34
SOURCE/DESTINATION ADDRESS GENERATOR
32
IP SPOOFING ATTACK GENERATOR
36
SERVICE COMMAND GENERATOR
20
TRANSPORT
22
NETWORK
24
DATALINK
26
HARDWARE
28

FIG.2



GET DESTINATION ADDRESS 100
GET SOURCE ADDRESS 102
GENERATE COMMUNICATION INITIATION MESSAGE 104
SEND MESSAGE TO TRANSPORT LAYER 108
COMMUNICATION HANDSHAKE MESSAGE ? 110
YES
NO
COMMUNICATION INITIATION MESSAGE TO TARGET COMPUTER 114
WAIT FOR RESPONSE 116
GET TCP SEQUENCE NUMBER FROM RESPONSE 120
NUMBER OF RESPONSES ? 122
NO
YES
COMPUTE SEQUENCE NUMBER OFFSET 126
A
A
SEND SOURCE ADDRESS TO NETWORK LAYER 130
SEND COMMUNICATION INITIATION MESSAGE TO TARGET COMPUTER 132
BUILD ACKNOW-LEDGMENT MESSAGE WITH SEQUENCE OFFSET 134
BUILD SERVICE COMMAND MESSAGE 138
SEND BOTH MESSAGES 140
DETECT RESPONSE ? 142
NO
LOG ATTACK FAILURE 144
YES
LOG ATTACK SUCCESS 146
NEXT SOURCE ? 148
YES
NO
NEXT DESTINATION ? 150
YES
NO
STORE LOG 152

FIG.3

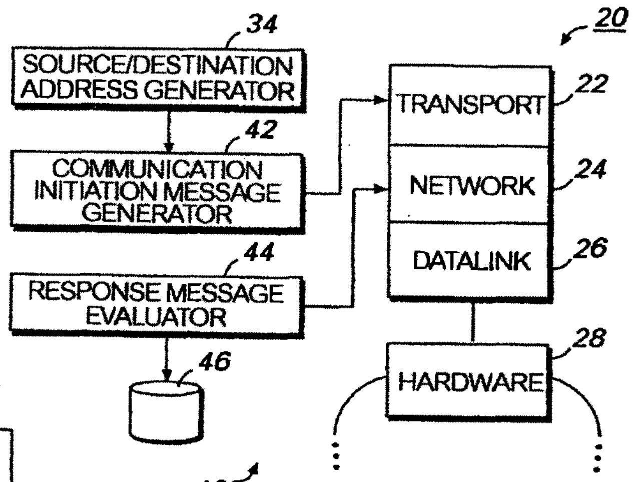

34
SOURCE/DESTINATION ADDRESS GENERATOR
42
COMMUNICATION INITIATION MESSAGE GENERATOR
44
RESPONSE MESSAGE EVALUATOR
46
20
TRANSPORT
22
NETWORK
24
DATALINK
26
HARDWARE
28
40

FIG.4

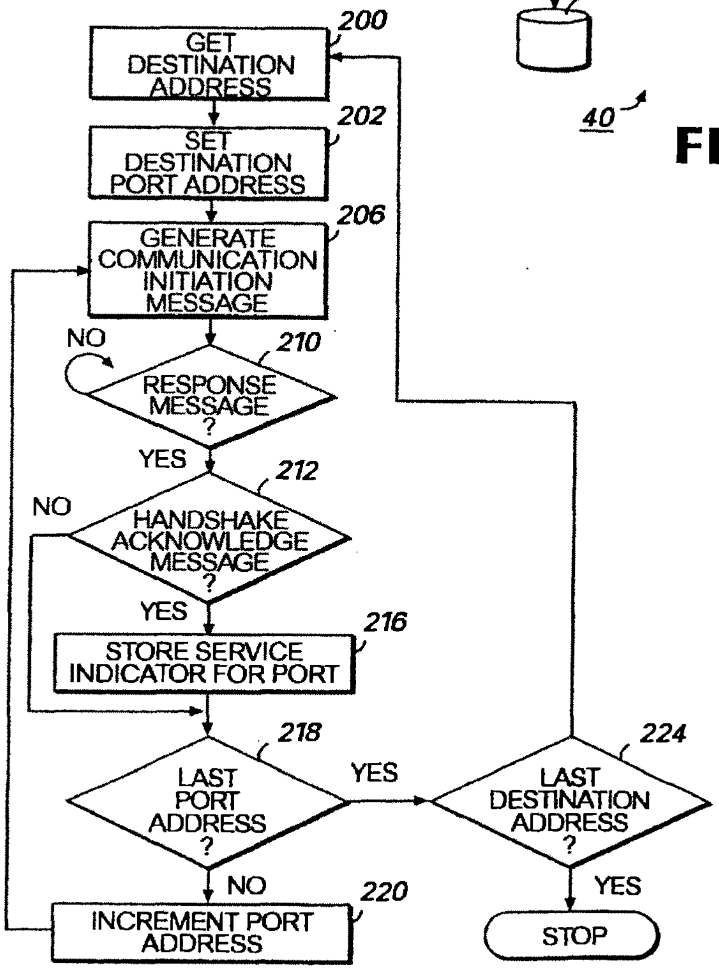

GET DESTINATION ADDRESS
200
SET DESTINATION PORT ADDRESS
202
GENERATE COMMUNICATION INITIATION MESSAGE
206
NO
RESPONSE MESSAGE ?
210
YES
NO
HANDSHAKE ACKNOWLEDGE MESSAGE ?
212
YES
STORE SERVICE INDICATOR FOR PORT
216
LAST PORT ADDRESS ?
218
YES
LAST DESTINATION ADDRESS ?
224
NO
INCREMENT PORT ADDRESS
220
YES
STOP

FIG.5

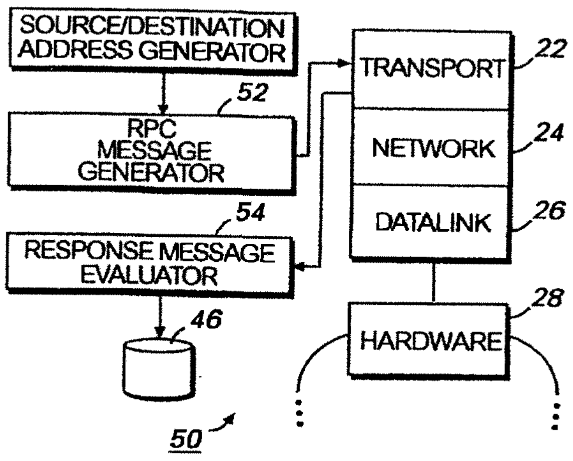

SOURCE/DESTINATION ADDRESS GENERATOR
RPC MESSAGE GENERATOR
52
RESPONSE MESSAGE EVALUATOR
54
46
50
TRANSPORT
22
NETWORK
24
DATALINK
26
HARDWARE
28

FIG.6

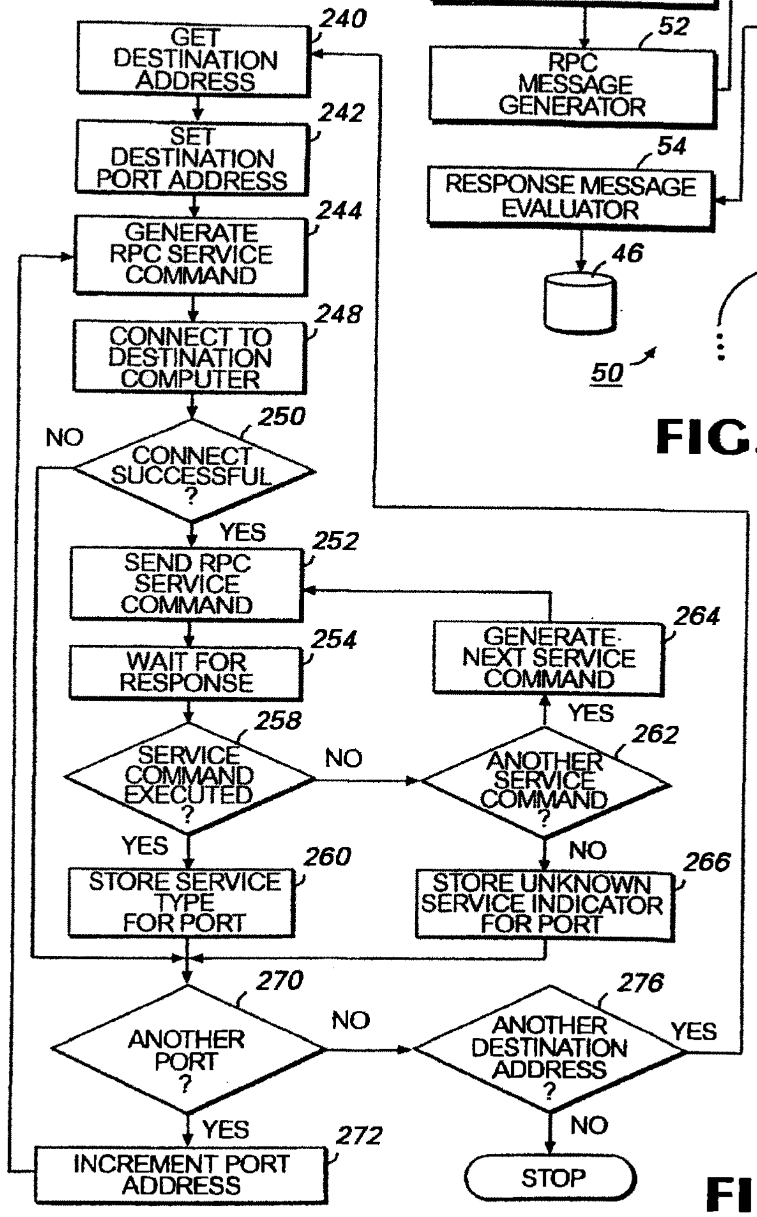

GET DESTINATION ADDRESS
240
SET DESTINATION PORT ADDRESS
242
GENERATE RPC SERVICE COMMAND
244
CONNECT TO DESTINATION COMPUTER
248
CONNECT SUCCESSFUL ?
250
NO
YES
SEND RPC SERVICE COMMAND
252
WAIT FOR RESPONSE
254
SERVICE COMMAND EXECUTED ?
258
NO
YES
GENERATE NEXT SERVICE COMMAND
264
YES
ANOTHER SERVICE COMMAND ?
262
NO
STORE SERVICE TYPE FOR PORT
260
STORE UNKNOWN SERVICE INDICATOR FOR PORT
266
ANOTHER PORT ?
270
NO
YES
ANOTHER DESTINATION ADDRESS ?
276
YES
NO
INCREMENT PORT ADDRESS
272
STOP

FIG.7

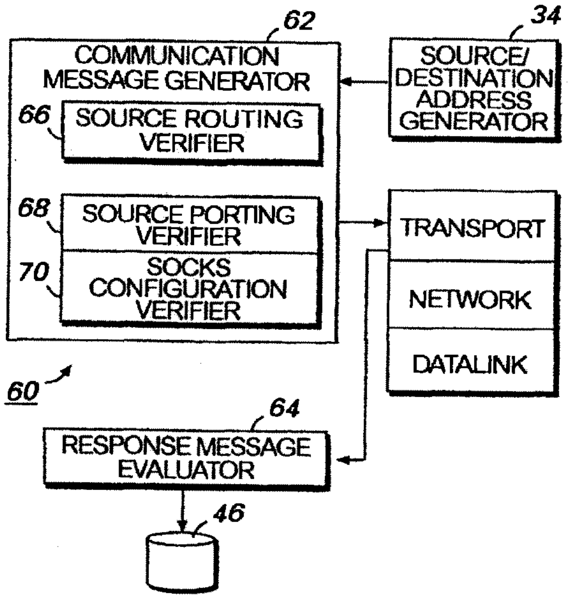

62
COMMUNICATION MESSAGE GENERATOR
34
SOURCE/ DESTINATION ADDRESS GENERATOR
66
SOURCE ROUTING VERIFIER
68
SOURCE PORTING VERIFIER
70
SOCKS CONFIGURATION VERIFIER
TRANSPORT
NETWORK
DATALINK
60
64
RESPONSE MESSAGE EVALUATOR
46

FIG.8

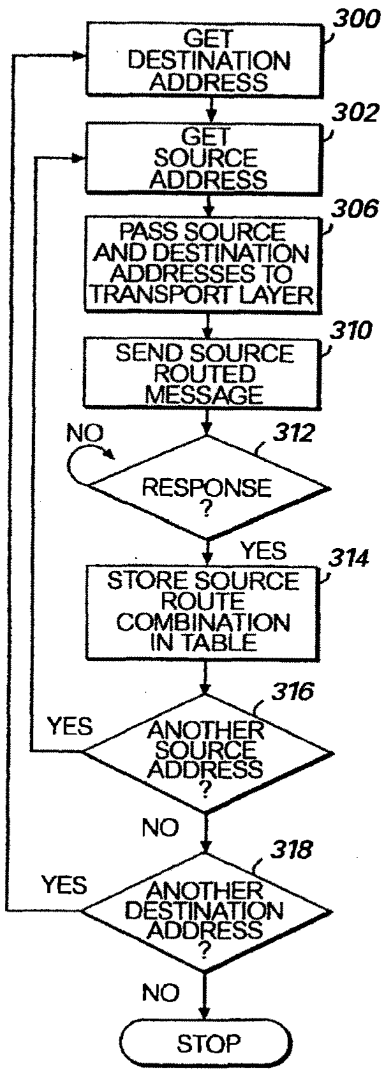

300
GET DESTINATION ADDRESS
302
GET SOURCE ADDRESS
306
PASS SOURCE AND DESTINATION ADDRESSES TO TRANSPORT LAYER
310
SEND SOURCE ROUTED MESSAGE
NO
312
RESPONSE ?
YES
314
STORE SOURCE ROUTE COMBINATION IN TABLE
316
YES
ANOTHER SOURCE ADDRESS ?
NO
318
YES
ANOTHER DESTINATION ADDRESS ?
NO
STOP

FIG.9

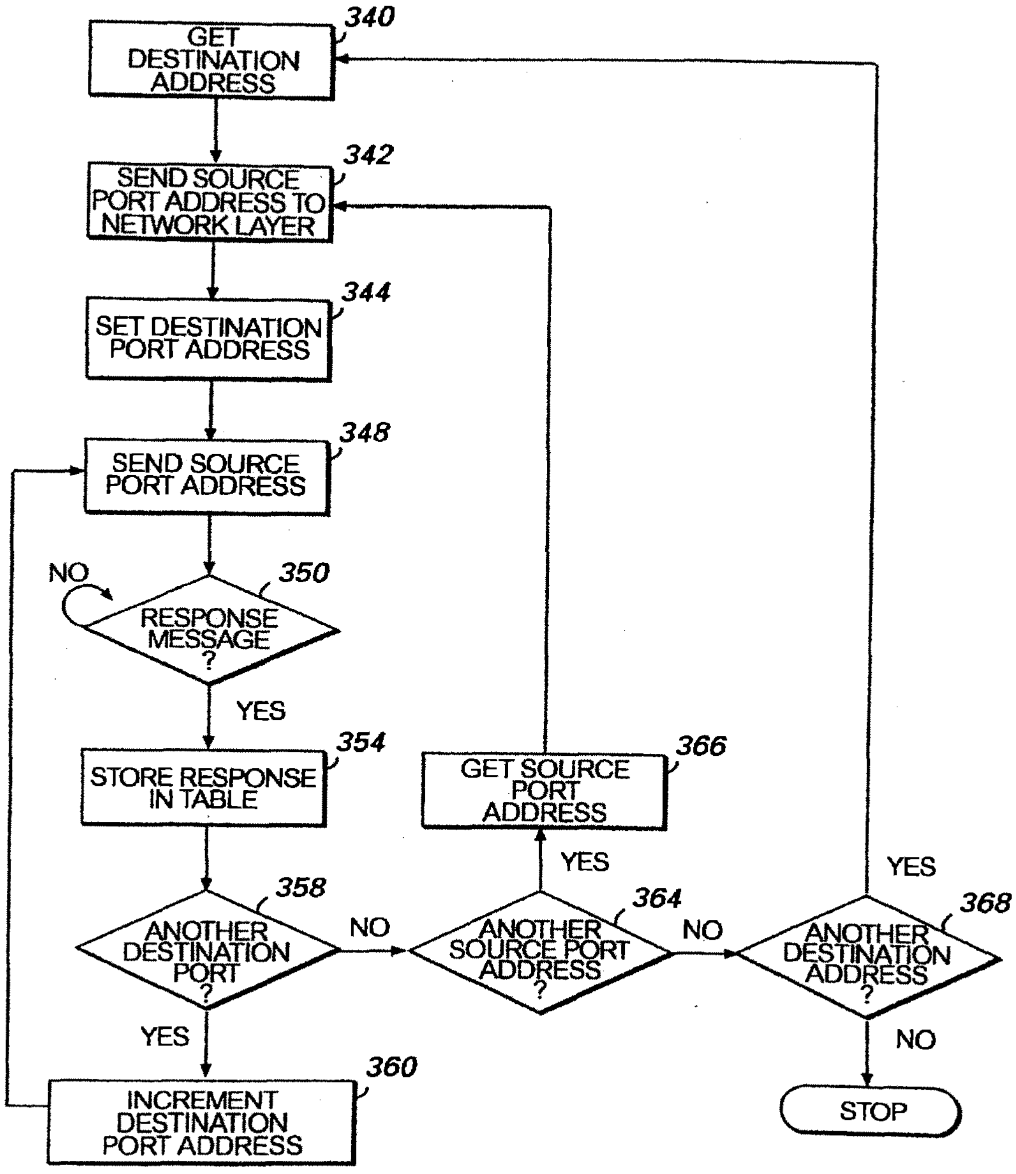

340
GET DESTINATION ADDRESS
342
SEND SOURCE PORT ADDRESS TO NETWORK LAYER
344
SET DESTINATION PORT ADDRESS
348
SEND SOURCE PORT ADDRESS
NO
350
RESPONSE MESSAGE ?
YES
354
STORE RESPONSE IN TABLE
358
ANOTHER DESTINATION PORT ?
NO
YES
360
INCREMENT DESTINATION PORT ADDRESS
364
ANOTHER SOURCE PORT ADDRESS ?
YES
366
GET SOURCE PORT ADDRESS
NO
368
ANOTHER DESTINATION ADDRESS ?
YES
NO
STOP

FIG.10

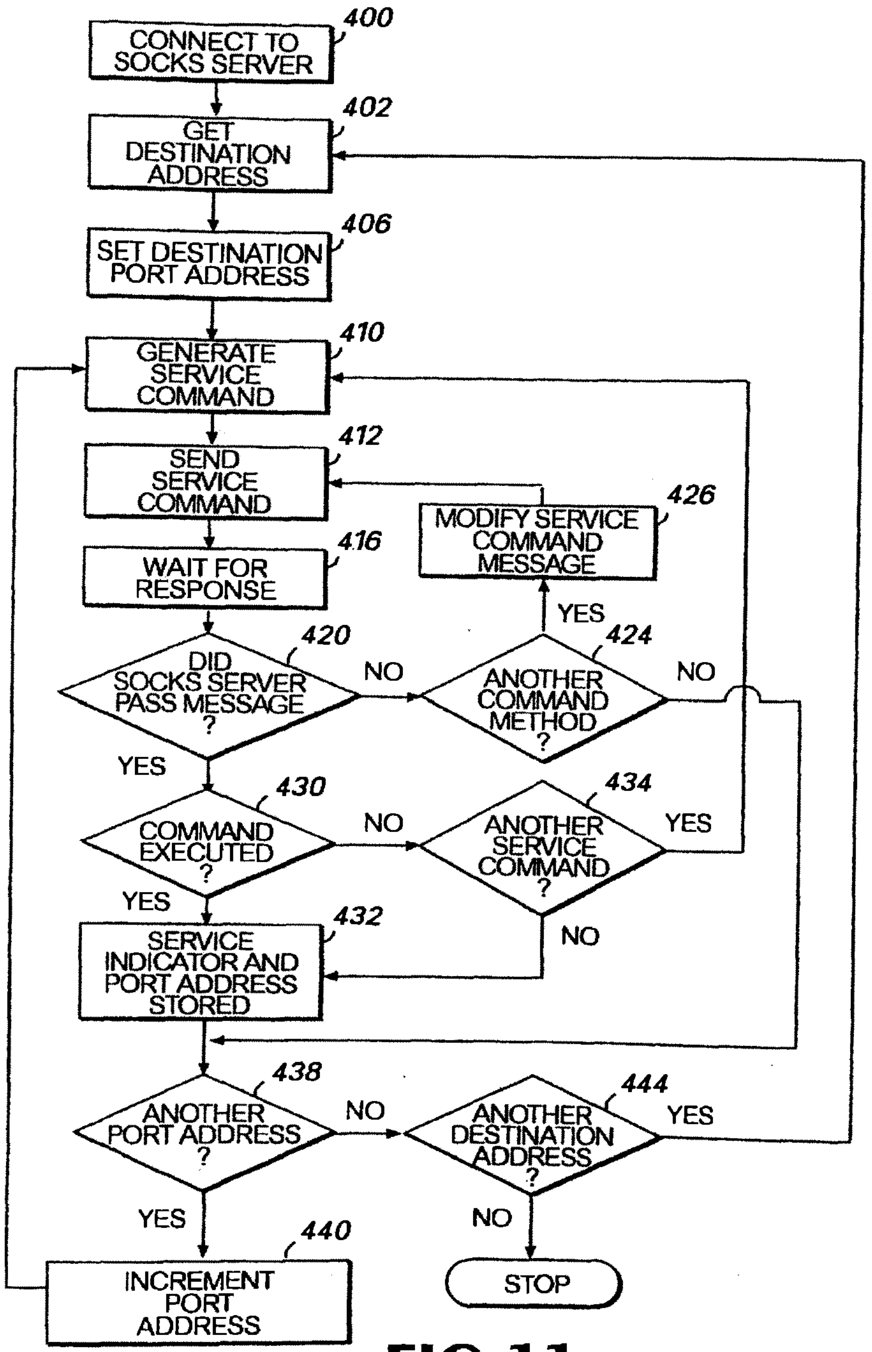

400
CONNECT TO SOCKS SERVER
402
GET DESTINATION ADDRESS
406
SET DESTINATION PORT ADDRESS
410
GENERATE SERVICE COMMAND
412
SEND SERVICE COMMAND
416
WAIT FOR RESPONSE
426
MODIFY SERVICE COMMAND MESSAGE
YES
420
DID SOCKS SERVER PASS MESSAGE ?
NO
424
ANOTHER COMMAND METHOD ?
NO
YES
430
COMMAND EXECUTED ?
NO
434
ANOTHER SERVICE COMMAND ?
YES
YES
NO
432
SERVICE INDICATOR AND PORT ADDRESS STORED
438
ANOTHER PORT ADDRESS ?
NO
444
ANOTHER DESTINATION ADDRESS ?
YES
YES
NO
440
INCREMENT PORT ADDRESS
STOP

FIG. 11

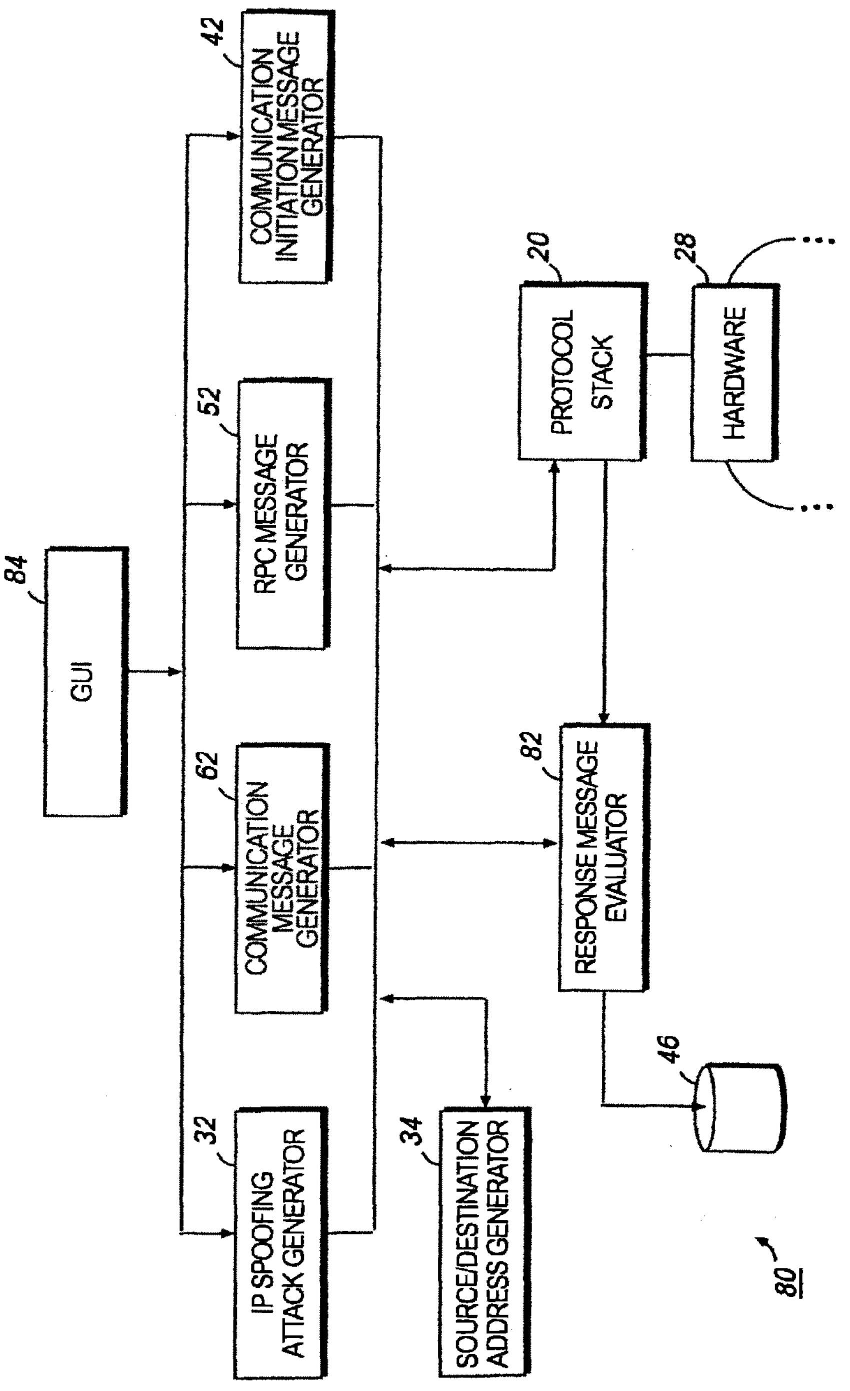

GUI
84
COMMUNICATION INITIATION MESSAGE GENERATOR
42
RPC MESSAGE GENERATOR
52
COMMUNICATION MESSAGE GENERATOR
62
IP SPOOFING ATTACK GENERATOR
32
SOURCE/DESTINATION ADDRESS GENERATOR
34
PROTOCOL STACK
20
HARDWARE
28
RESPONSE MESSAGE EVALUATOR
82
46
80

FIG.12

METHOD AND APPARATUS FOR DETECTING AND IDENTIFYING SECURITY VULNERABILITIES IN AN OPEN NETWORK COMPUTER COMMUNICATION SYSTEM

FIELD OF THE INVENTION

This invention relates to network communications for computers, and, more particularly, to computer communications over open networks.

BACKGROUND OF THE INVENTION

Many business and scientific organizations in the United States which use more than one computer in their operations couple the computers together through a network. The network permits the computers to be islands of processing which may share resources or data through communication over the network. The data which may be communicated over the network may take the form of programs developed on a user's computer, data files created on a user's computer, electronic mail messages and other data messages and files which may be generated or modified by a user at a user's computer. Typically, the user's computer includes an operating system for controlling the resources of the user's computer, including its central processing unit ("CPU"), memory (both volatile and non-volatile memory) and computer peripherals such as printers, modems and other known computer peripheral devices. The user typically executes application programs and system services to generate data files or programs.

Most computers are coupled to a network through a network communication printed circuit card which is typically resident within each computer system. This communication card typically includes processors, programs and memory to provide the electrical signals for transmission of data and implement the protocol which standardizes the messages transmitted through a network. To communicate data from a user's application program or operating system service, a protocol stack is typically implemented between the communication card for the network and the operating system services and application programs.

The typical protocol stack used on most open networks is a Transport Control Protocol/Internet Protocol ("TCP/IP"). This protocol stack includes a transport layer which divides a data stream from an application program or service into segments and which adds a header with a sequence number for each segment. The TCP segments generated by the transport layer are passed to the Internet Protocol ("IP") layer. The IP layer creates a packet having a packet header and a data portion. The data portion contains the TCP segment and the packet header contains a source address identifying the computer sending a message and a destination address identifying the computer for which the message is intended. The IP layer also determines the physical address of the destination computer or an intermediate computer, in some cases, which is intended to receive the transmitted message. The packet and the physical addresses are passed to a datalink layer. The datalink layer typically is part of the program implemented by a processor on the communication card and it encapsulates the packet from the IP layer in a datalink frame which is then transmitted by the hardware of the communication card. This datalink frame is typically called a packet. For purposes of this specification, the word "message" includes the data entities packet and datalink frame.

At the destination computer, the communication card implements the electrical specification of a hardware communication standard, such as Ethernet, and captures a data message from a source computer. The datalink layer at the destination computer discards the datalink header and passes the encapsulated packet to the IP layer at the destination computer. The IP layer at the destination computer verifies that the packet was properly transmitted, usually by verifying a checksum for the packet. The IP layer then passes the encapsulated TCP segment to the transport layer at the destination computer. The transport layer verifies the checksum of the TCP message segment and the sequence number for the TCP packet. If the checksum and TCP sequence number are correct, data from the segment is passed to an application program or service at the destination computer.

Segregation of communication functions in the various layers of the protocol stack and the segregation of the protocol stack from the communication card and application programs, modularizes the functions required to implement communication over a computer network. This modularization of functions simplifies computer communication operation and maintenance. It also does not require a user to have knowledge of how the protocol stack and communication card communicate in order to send data messages to other computers over the network.

All of the computers coupled to a network may have approximately the same resources available at each machine. The type of network is sometimes called a peer to peer network. Another type of network environment is one in which one computer controls shared databases and other computer resources with other computers over the network. The computer controlling access to the shared resources is typically called a server and the computers utilizing the shared resources are called clients.

In both the client/server and peer to peer environments, a server or computer may be used as a gateway to other networks or computers. Another device which a message may encountered as it moves along a network is a router. A router examines destination addresses of messages it receives and routes them in an efficient manner to the specified destination computer. For example, a server on a first network may be coupled to a router which is coupled to a plurality of servers including a server on a second network and a server for a third network. In this type of environment, the computer on the first network may communicate with a computer on the third network by generating data messages which have the destination address for a computer on the third network. The message circulates through the first network and is eventually provided to the server of the first network. The server of the first network then passes the message to the router which determines that the message is addressed for the third network. Accordingly, it sends the message to the server of the third network. The communication facilities at the server for the third network recognize the destination address as existing on the third network and pass the message to a computer on the third network where it eventually would be passed to the destination computer.

While this type of communication effectively and efficiently couples all of the computers from all of the networks together without requiring a message to pass through each computer on the network, a message typically passes through a number of computers, routers, servers or gateways prior to reaching the destination computer. As a result, the data messages from one computer to another computer may be intercepted and data obtained from the message as the message is passed on to another computer. The type of network wherein this type of accessible communication is provided is typically called an open network. One of the more popularly known open networks is the Internet where literally millions of servers and computers are coupled through a TCP/IP communication protocol.

While the open network architecture of the Internet permits a user on a network to have access to information on many different computers, it also provides access to messages generated by a user's computer and to the resources of the user's computer. In fact, there are persons who attempt to use knowledge regarding the operations of the protocol stack and operating systems in an effort to gain access to computers without authorization. These persons are typically called "hackers". Hackers present a significant security risk to any computer coupled to a network where a user for one computer may attempt to gain unauthorized access to resources on another computer of the network. For example, an employee may attempt to gain access to private and confidential employee records on a computer used by the human resources department of an employer.

In an effort to control access to a network and, hence, limit unauthorized access to computer resources available on that network, a number of computer communication security devices and techniques have been developed. One type of device which is used to control the transfer of data is typically called a "firewall". Firewalls are routers which use a set of rules to determine whether a data message should be permitted to pass into or out of a network before determining an efficient route for the message if the rules permit further transmission of the message. In this specification the term "routers" includes firewalls and routers.

In the TCP/IP protocol, a communication connection is established through a three handshake open network protocol. The first handshake or data message is from a source computer and is typically called a "synchronization" or "sync" message. In response to a sync message, the destination computer transmits a synchronization-acknowledgment ("sync-ack") message. The source computer then transmits an acknowledgment ("ack") message and a communication connection between the source and destination computer is established. To limit access to computers on a network, routers may be provided as a gateway to the network and programmed to detect and block sync messages being transmitted from a computer external to the network to a destination computer on the network. That is, computers on the network may send out sync messages through the router to initiate communication with other computers, but computers outside the router and its network cannot send sync messages through the router to initiate communication with computers on the network. In this way, a hacker cannot attempt to initiate communication with a computer on the network.

Hackers, however, have developed other ways which may be helpful in bypassing the screening function of a router. For example, one computer, such as a server on the network, may be permitted to receive sync messages from a computer outside the network. In an effort to get a message to another computer on a network, a hacker may attempt to use source routing to send a message from the server to another computer on the network. Source routing is a technique by which a source computer may specify an intermediate computer on the path for a message to be transmitted to a destination computer. In this way, the hacker may be able to establish a communication connection with a server through a router and thereafter send a message to another computer on the network by specifying the server as an intermediate computer for the message to the other computer.

In an effort to prevent source routing techniques from being used by hackers, some routers may be configured to

intercept and discard all source routed messages to a network. For a router configured with source routing blocking, the router may have a set of rules for inbound messages, a set of rules for outbound messages and a set of rules for source routing messages. When a message which originated from outside the network is received by such a router, the router determines if it is a source routed message. If it is, the router blocks the message if the source routing blocking rule is activated. If blocking is not activated, it allows the source routed message through to the network. If the message is not a source routed message, the router evaluates the parameters of the message in view of the rules for receiving messages from sources external to the network. One such rule is the external sync message filter discussed above. Other rules may also be implemented in such a router. However, a router vulnerability exists where the rules used by the router are only compared to messages that are not source routed and the source routed blocking rule is not activated. In this situation, the router permits source routed messages through without comparing them to the filtering rules. In such a case, a computer external of the network may be able to bypass the external sync message filter and establish a communication connection with a computer on the network by using source routed messages.

What is needed is a system and method for verifying that the source routing blocking feature of a router has been activated.

Networks may also be coupled to external computers through a specialized communication filter typically known as a "Socks" proxy server. A Socks proxy server is interposed between a network and external computers. For an external computer to establish communication with a computer on a network coupled to a Socks server, the external computer first establishes a communication connection with the Socks server and the Socks server establishes a communication connection with the destination computer. Thereafter, the Socks server relays messages between the external computer and a computer on the network only if they comply with the filter rules configured for the Socks server. Typically, Socks servers are used to interface e-mail, File Transfer Protocol ("FTP") and Telnet communication services between computers on a network and computers external of the network and to block access to most other ports on a network. The interrogation and evaluation of messages through a Socks server is dependent upon the network administrator for proper configuration. Known methods for verifying the configuration of the Socks server is to view the configuration files of the Socks server to verify the rules are properly set. However, this method does not ascertain the rules actually being implemented by the Socks server.

What is needed is a method and system for determining the rules being implemented by a Socks server without reviewing the configuration files for a Socks server.

Another entry port for hackers are commonly known services which provide information to external users without requiring authorization checks such as passwords. Most implementations of the UNIX operating system, for example, include Remote Procedure Call (RPC) services which may not be protected by authorization checks. The ports on which RPC services are located may be determined by querying a UNIX operating system service known as "portmapper". In an effort to obtain knowledge regarding accessible services on a computer, a hacker may make an inquiry of the portmapper service at its port in order to obtain information regarding the RPC services available for entry on the computer. Although the portmapper service may be reconfigured to include an authorization check that still does not provide an authorization check for the RPC services themselves.

What is needed is a system and method for detecting and reporting to a network administrator those ports which are coupled to RPC services which have little or no authorization checks.

As discussed above, the transport layer of the protocol stack provides a sequence number for each data segment to be transmitted. In the TCP/IP protocol, the sequence number is called a TCP sequence number which is placed in the TCP header generated by the transport layer. The sequence number for the data segment is typically incremented at predefined time units, for example, each second, and for each communication connection or attempted communication connection. For example, in attempting to establish communication with another computer on a TCP/IP network, the source computer generates a sync message with a TCP sequence number. The destination computer responds with a sync/ack message where the ack value in the message is the sequence number from the received sync message and the sequence number for the destination computer is a number generated by the destination computer. This sequence number typically has the value of the last TCP sequence number generated by the destination computer plus the addition of a preferred offset value for each predefined time unit and communication connection that has occurred since the last TCP sequence number was generated. The ack message from the source computer to the destination computer which completes the communication connection must include the TCP sequence number received from the destination computer in the sync/ack message.

One known way which hackers attempt to access a computer on a network is to emulate the communication of messages from another computer on the network. A hacker emulates another computer on the network by first blocking a communication port on the computer being emulated by repeatedly sending sync messages to a port on the computer. This causes the communication program for the port to fill its communication buffer with half-open communication connections. When the buffer is full, no more sync messages are accepted until the oldest attempted half-open communication connection times out. Typically, the time out period is ten minutes or longer. In order to obtain a sequence number, the hacker's computer sends a number of sync messages to the computer which is the target of the attack which responds with a plurality of sync/ack messages containing TCP sequence numbers to the hacker's computer. The TCP sequence numbers from the sync/ack messages may be compared to statistically determine the offset used by the target computer to generate TCP sequence numbers. The hacker then uses the emulated computer's blocked port address as the source computer address for a sync message originated by the hacker's computer. In response, the target computer replies with a sync/ack message which is addressed to the blocked computer port of the emulated computer. Thus, the hacker's computer does not receive the sync/ack message with the TCP sequence number required for a proper response. However, the hacker's computer then sends an ack message with the next computed sequence number derived from bombarding the target computer with sync messages. If the sequence number has been correctly computed so that it matches the sequence number in the sync/ack message sent by the target computer to the blocked computer port, a communication connection is established and the hacker is able to transmit a command to the service on the port of the target computer through which commu-

nication has been established. In a UNIX system, a hacker normally attacks the ports coupled to the rsh and rlogin services since the authorization check for these services is usually the source address. If the hacker is able to successfully emulate a computer on the network having an address authorized for the service on the target computer port, the command is executed by the service. The service command typically provided to the port of the target computer disrupts the target computer's operation so the hacker's computer has unencumbered access to the target computer's resources. These types of attacks which use predicted TCP sequence numbers are typically known as IP spoofing attacks.

Although the protocol stack for each computer uses different offset values to generate the initial TCP sequence number for establishing communication links, some machines generate initial sequence numbers which are more easily predicted than others. What is needed is a way of detecting which computers on a network are susceptible to attacks using predicted TCP sequence numbers.

SUMMARY OF THE INVENTION

The above-noted vulnerabilities of a computer network may be automatically detected by a computer program which implements the system and method of the present invention. One embodiment of the present invention includes an Internet protocol ("IP") spoofing attack generator for generating an IP spoofing attack directed to a target computer and a service command message generator for sending a command to be executed by a service coupled to a port on the target computer so that in response to the target computer being compromised by the IP spoofing attack the target computer generates a compromise indicator without altering or destroying the target computer's services and/or operations. Preferably, the target computer response is an electronic mail message or a Telnet initiation message. Preferably, the IP spoofing attack is directed against a port coupled to the rsh or rlogin services. Preferably, the embodiment includes a source/destination address generator which generates source and destination addresses for messages corresponding to an open network protocol. The destination addresses correspond to the target computer and the source addresses correspond to the emulated computer in the IP spoofing attack. The source/destination address generator generates the address for each computer on a network so that an IP spoofing attack from every computer on the network is directed against each of the other computers on the network. In this manner, those computers on the network which are most susceptible to an IP spoofing attack may be detected and modification of the TCP sequence number generator in the protocol stack may be adjusted to make an IP spoofing attack less likely to succeed.

Another embodiment of the present invention for detecting security vulnerabilities in the configuration rules of a router includes a communication message generator for generating and sending communication messages to computers coupled through an open network to a router and a response message detector for detecting responses from computers on the network generated in response to the communication messages. This embodiment of the present invention detects the vulnerability of the router to pass communication messages to computers on the network. Depending on the type of communication or service command message to which a computer responds, the inventive system may determine rules not implemented by a router. In one preferred embodiment, the communication message generator includes a Socks configuration verifier which establishes a communication connection with a Socks server and attempts to send service command messages for different services with source addresses for computers on the network. The responses of the destination computer are examined to determine the types of messages which the Socks server passes to computers on the network from computers external to the network. This system may be used to verify the rules actually implemented by a Socks server.

In another embodiment, the communication message generator includes a source porting verifier which sets the source port address in a header for a generated communication message to a predetermined value to see if the router passes externally generated messages having the specified source port address to the network. Preferably, the predetermined value is the default source port identifier for a service having a known required predetermined source port address such as an FTP service. In this manner, the system of the present invention detects whether a computer external of the network can establish a communication connection with a computer on the network by using a predetermined source port identifier to avoid other rules in a router.

In another embodiment of the present invention, the communication message generator includes a source routing verifier which generates source-routed communication messages to determine whether the router has a source router message blocking rule activated. This embodiment may be used to determine whether the rules that the router applies to communication messages originated by computers external to the network may be bypassed by using source routed messages.

In another embodiment of the present invention, an RPC message generator generates RPC service command messages which are sent to ports of computers on a network to detect the ports coupled to RPC services having little or no authorization checks. These ports and the coupled services, if determined, may be stored and provided to a network administrator for installing more rigorous authorization checks.

In another embodiment of the present system, a communication initiation message generator for generating communication initiation messages for a three handshake protocol and a response message evaluator are used to determine which of the ports on each computer in a network have a service coupled thereto. This inventive system operates by sending sync messages to each port on every computer on the network and building a table of service identifiers which identify those ports which responded with a message indicating the presence of a service. Preferably, the communication initiation message is a sync message for TCP/IP networks and the messages indicating a service is coupled to a port is a sync/ack message. In this manner, the inventive system may build a map of those ports of each computer on the network which have service coupled thereto without creating a log of any communication connections on any the computers on the network. Since communication connections are only established and logged when the originating computer sends the ack message, this embodiment generates a map of available services in a stealth manner. This embodiment of the inventive system may be coupled with one or more of the other embodiments which generate service command messages to eliminate ports from the attempts to detect vulnerable services. Such a system speeds the security analysis of a network.

These and other advantages and benefits of the present invention may be ascertained from reading of the detailed specification in conjunction with the drawings.

DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated and constitute a part of this specification, illustrate a number of

embodiments of the invention and, together with the general description given above, and the detailed description of the embodiments given below, serve to explain the principles of the invention.

FIG. 1 is a schematic of an open network system;

FIG. 2 is a block diagram of an embodiment of the present invention used to detect IP spoofing attack vulnerability;

FIG. 3 is a flow chart of the preferred process implemented by the embodiment depicted in FIG. 2;

FIG. 4 is a block diagram of an embodiment of the present invention used to map the ports of computers of a network which are coupled to services without generating communication connections;

FIG. 5 is a flow chart of the preferred process implemented by the embodiment depicted in FIG. 4;

FIG. 6 is a block diagram of an embodiment of the present invention used to detect Remote Procedure Call (RPC) services available on a network which have little or no authorization checks;

FIG. 7 is a flow chart of the preferred process implemented by the embodiment shown in FIG. 6;

FIG. 8 is a block diagram of an embodiment of the present invention used to verify the configuration of routers and/or Socks servers;

FIG. 9 is a flow chart of the preferred process implemented by the source routing verifier of FIG. 8;

FIG. 10 is a flow chart of the preferred process implemented by the source porting verifier of FIG. 8;

FIG. 11 is a flow chart of the preferred process implemented by the Socks server verifier of FIG. 8; and

FIG. 12 is a block diagram of a preferred embodiment of the present invention which incorporates the components of the systems shown in FIGS. 2, 4, 6 and 8.

DETAILED SPECIFICATION OF EMBODIMENTS OF THE INVENTION

An open network system in which a system made in accordance with the principles of the present invention may be used is shown in FIG. 1. An internetwork 10 may be comprised of a network 12 which in turn may be coupled to other servers, gateways and routers. Network 12 includes a plurality of computers C_1–C_n which are coupled through network 12 to a server S_1. This network in turn may be coupled to a router R_1 to provide further secured computer communication with other servers represented by S_m or other routers labeled R_x as shown in FIG. 1. Although the principles of the present invention are extensible to other protocols, the invention is preferably used on networks which utilize the TCP/IP protocol. The computer program implementing a system or method of the present invention may reside on any of the computers on the network 12 or any server or any router of internetwork 10.

Structure of a system embodiment made in accordance with the principles of the present invention is shown in FIG. 2. A computer executing a program implementing the system or method of the present invention would typically include the programs and communication hardware card which implement a protocol stack 20. Protocol stack 20 is comprised of transport layer 22, network layer 24 and datalink layer 26. These layers of protocol stack 20 operate in the well-known manner set forth above. The data frame prepared by datalink layer 26 is passed to communication hardware 28 for transmission to other computers in accordance with the source and destination information provided in the various headers generated by protocol stack 20.

In one embodiment of the present invention which detects a computer's vulnerability to IP spoofing, the system includes an IP spoofing attack generator 32, a source/destination address generator 34 and a service command generator 36. Source/destination address generator 34 identifies the internet and physical addresses of the computers on the network 12 to be tested. Source/destination address generator 34 verifies that each computer on network 12 is emulated in IP spoofing attacks on all of the other computers on network 12. In this manner, the inventive system exhaustibly tests all possible attack combinations on a network. Service command generator 36 generates commands for a service which may be coupled to a port which IP spoofing attack generator 32 is able to initiate a communications connection. Preferably, service command generator 36 generates commands for services which have little or no authorization checks. "Little" means that the authorization check verifies a computer address is on the network 12 or the like while "no" authorization check means the service executes any valid server command received on a port regardless of originating source. Preferably, service commands are generated for electronic mail, file transport protocol (FTP) and Telnet services. These commands preferably indicate that a target computer identified by a destination address has been compromised without altering the target computer's operational parameters such as changing system privileges for a user or deleting data files. Examples of such commands include a Telnet session initiation command such as telnet attack_computer_address where attack_computer_address is the address of the computer which performed the IP spoofing attack on the target computer. Another example of such a message is mail admin message where admin indicates the system or network administrator's mailbox and message indicates the contents of the message informing the administrator of the compromise. The service command received from command message generator 36 and the source and destination addresses received from source/destination address generator 34 are used by IP spoofing attack generator 32 to provide data and header content for messages sent to transport layer 22 and network layer 24 of protocol stack 20 which are used to implement the IP spoofing attack and detection.

The process implemented by IP spoofing attack generator 36 is shown in FIG. 3. That process begins by obtaining a destination address (Block 100) and a source address (Block 102) from source/destination address generator 34. Attack generator 32 then generates a communication initiation message for a three handshake protocol which is preferably a synchronization or sync message for the TCP/IP protocol (Block 104). The communication initiation message is sent to a port on the source address computer by placing the message in a TCP segment and passing it to the transport layer (Block 108). Transport layer 22, network layer 24 and datalink layer 26 all appropriately encapsulate the sync message for transmission to the computer at the source address which is the address of the computer to be emulated in the IP spoofing attack. The process awaits the reception of a handshake acknowledgment message from the computer at the source address (Block 110). The handshake acknowledgment message in the TCP/IP protocol is a sync/ack message. If a sync/ack message is received, another sync message is generated and sent to the same port address of the computer at the source address. This process continues until no sync/ack message is received from the computer at the source address within a predetermined time. These steps are performed to fill the communication buffer for a port on the source address computer with half-opened communication connections. This full buffer condition exists until the time period for completing a communication connection expires. In most computers, the expiration period is at least 10 minutes which is typically enough time to complete the attack. Because its buffer is full, this port on the computer at the source address no longer responds to communication initiation messages.

A sync message is then generated and transmitted to the computer at the destination address which now defines the target computer (Block 114). The process waits for a sync/ack message from the computer at the destination address (Block 116). When it is received, the process retrieves the TCP sequence number from the TCP segment header (Block 120) and checks to see if a predetermined number of TCP sequence numbers have been retrieved from the target computer at the destination address (Block 122). If the predetermined number of sequence numbers has not been received, a time period corresponding to the unit of time between changes in TCP sequence number modifications is delayed. This delay permits the computer at the destination address to modify the TCP sequence number which is used for initiating a communication session. Alternatively, the destination port address on the target computer may be changed to cause a sequence number increment as well. After this delay has expired or the destination port address changed, another sync message is generated and sent to the target computer (Block 114). When the predetermined number of TCP sequence numbers have been received, the TCP numbers are used to evaluate the offset between TCP sequence numbers or the pattern for generating the TCP numbers (Block 126). For example, if a predetermined offset amount is added to generate a new TCP sequence number for communication initiation, three TCP sequence numbers may be used to compute the difference between two adjacent TCP numbers. This difference should indicate the predetermined offset so that the next TCP sequence number which would be used by the target computer to respond to a new sync message is determined.

The IP spoofing attack process continues by setting the source address in the network layer 24 to the source address retrieved from source/destination address generator 34 (Block 130). Now messages generated by the computer implementing the system and method of the present invention generates messages which appear to be originated from the computer at the source address. A communication initiation message is then generated and transmitted to the computer at the destination address (Block 132). A period of time is delayed which corresponds to the normal response time for the target computer to send a sync/ack message. The process then prepares an ack message with the predicted TCP sequence number (Block 134). A service command is obtained from a service command generator 36 and placed in a TCP segment passed to transport layer 22 to build a service command message (Block 138). Both messages are then transmitted to the target computer to emulate an ack message and service command message from the emulated computer with the blocked port. If the predicted TCP sequence number for the ack message having the source address of the emulated computer matches the TCP sequence number sent by the target computer in the sync/ack message, the target computer establishes a communication connection which accepts messages having a source address of the emulated computer. Now the service command message sent from the computer implementing the process of FIG. 3 is accepted and executed by the service coupled to the port if the command is valid for the service. Preferably, the service command causes the computer at the destination address to log the attack at the computer which has been compromised and, most preferably, the command causes the target computer to send a compromise indicator to the computer implementing the process of FIG. 3, although another computer may receive the compromise indicator. The success or failure of the attack is logged (Block 142–146). Preferably, a Telnet session is established between the compromised target computer and the computer executing the program which implements the process of FIG. 3. Initiation of the Telnet session may be logged to record the success of the IP spoofing attack and additional information may be obtained during the Telnet session about the compromised computer to search for other security vulnerabilities of the target system.

The process then determines whether another source address exists on the network (Block 148), and if there is, an attack on the target computer is attempted using the computer at the new source address as the emulated computer. If all of the source addresses have been used, the process checks to see if another destination address is available (Block 150). If another source address is available, the process is repeated to evaluate attacks from each of the other computers on the network on the target computer defined by the new destination address. This process continues until each computer on the network has been used to attack all the other computers on the network. Once this has been done, the attack log may be stored in table 46. The log may be later displayed to identify those computers on the network that are susceptible to IP spoofing attacks or provide other information obtained from the target computers that were compromised (Block 152).

Another embodiment of the present invention is shown in FIG. 4. System 40 includes a communication initiation message generator 42 and a response message evaluator 44 for determining whether a service is coupled to a port responding to a communication initiation message. System 40 builds a topology table 46 of service ports for network 12 from the communication initiation responses without causing a communication connection which may be logged by the computer having the ports which are being interrogated. Communication initiation message generator 42 is coupled to transport layer 22 of protocol stack 20 so communication initiation messages may be provided to transport layer 22 for transmission to the ports of the other computers coupled to network 12. Preferably, the communication initiation messages are sync messages used in the three handshake protocol of a TCP/IP network. Response evaluator 44 is also coupled to transport layer 22 to receive the response messages to the communication initiation messages sent by a computer executing a program implementing the process shown in FIG. 5. If the response message is the handshake acknowledgment message in the communication connection process, response evaluator 44 records the port address as a service access port for network 12 in table 46. In the three handshake protocol used to establish a communication connection on a TCP/IP network, a sync/ack message is the handshake acknowledgment message which indicates a service is present on a port.

The process implemented by system 40 of FIG. 4 is shown in FIG. 5. The process begins with communication initiation message generator 42 obtaining a destination address of a computer on network 12 from source/destination address generator 34 (Block 200) and the destination port address is set to the first port address on the destination computer (Block 202). Most computers in a TCP/IP protocol have port addresses in the range of 0–65,535. Preferably, each port address is tested by system 40. A communication initiation message is generated for the first port address of the computer at the destination address and passed to transport layer 22 (Block 206). After the communication initiation message is transmitted, response evaluator 44 waits for receipt of a response message from the port to which the communication initiation message was sent (Block 210). Response evaluator 44 then determines whether the message is a handshake acknowledgment message (Block 212). If it is, response evaluator 44 stores a service indicator, the destination address and port address in service topology table (Block 216). In a TCP/IP network, a sync/ack message indicates a service is coupled to the port while a reset message indicates no service is coupled to the port. The process then checks to see if the port address is the last possible port address on the computer (Block 218). If it is not, the port address is incremented (Block 220) and a new communication initiation message is sent to the next port address of the computer at the destination address (Block 206). The process continues until all of the port addresses on a computer have been tested to determine whether a service is coupled to each port. After each port has been checked for a service, the process determines whether another destination address is available (Block 224). If there is, another destination address is obtained (Block 200) and the process continues at the first port address for the next computer. The process terminates when all of the computers on network 12 have been checked.

Another embodiment of the present invention is shown in FIG. 6. In system 50, a RPC message generator 52 and response evaluator 54 are coupled to transport layer 22. RPC message generator 52 generates a data segment having a command for an RPC service which may not require an authorization check such as a password. Response message evaluator 54 determines from a message received in response to the RPC service command message whether an RPC service having little or no authorization check is available over the network. A record of this service may be provided to the system or network administrator.

The process implemented by system 50 is depicted in FIG. 7. The process begins by obtaining a destination address for a computer on the network 12 from source/destination address generator 34 (Block 240). The destination port address is initialized to the first port address on the computer at the destination address (Block 242) and a first RPC service command is generated by RPC message generator 52 (Block 244). Preferably, a CONNECT command which identifies the destination address and port address is issued to transport layer 22 (Block 248). Once a communication connection has been established, transport layer 22 notifies RPC message generator 52 (Block 250). RPC message generator 52 then passes the generated service command to transport layer 22 and a message containing the service command is transmitted to the port with which communication has been established (Block 252). Response message evaluator 54 then waits for a response (Block 254). If a response is detected which indicates the service command was executed (Block 258), the destination address, port address and type of RPC service is stored in topology table 46 (Block 260). If no communication connection was established with the port, no entry is made for the port. If communication is established but the port does not respond to the first service command, RPC message generator 52 determines if another RPC service command is available (Block 262) and, if there is, it generates a service command for another service (Block 264) and passes the command to transport layer 22 (Block 252). There are a number of known RPC commands for the UNIX operating system and RPC message generator 52 may generate a service command for each one to determine if it exists on a port being tested. If the process does not determine that an RPC service is coupled to the port, it identifies the service as a non-RPC service and stores an unknown or non-RPC service indicator in table 46 (Block 266). Response evaluator 54 evaluates any message received which was responsive to the next service command (Blocks 254, 258). After the process finishes its interrogation of a port for the type of service coupled to the port, the process determines whether another port exists (Block 270). If there are other ports to be interrogated, the port address is incremented (Block 272) and the process continues until all the ports on the computer at the destination address have been tested. The process then continues by determining whether another destination address for a computer on the network exists (Block 276) and, if it does, repeating the process for each port on that computer. When the process of FIG. 7 is completed, a topology map has been built which identifies the port and the RPC service coupled to each port for each computer on the network.

System 50 of FIG. 6 may be combined with system 40 of FIG. 4 such that once topology table 46 identifying those ports which are coupled to a service has been generated by response evaluator 44 of system 40, RPC message generator 52 need only attempt to identify which of the ports identified as being coupled to a service are coupled to an RPC service having little or no authorization check. Response evaluator 54 of system 50 message generator may then identify the RPC services for those ports which respond to service commands generated by RPC message generator 52.

An embodiment used to test the configuration of a router is shown in FIG. 8. System 60 includes a communication message generator 62 and a response evaluator 64. Preferably, communication message generator 62 includes a source routing verifier 66, a source porting verifier 68 and a Socks configuration verifier 70. Socks configuration verifier 70 and source routing verifier 66 execute in the application layer of a computer which is located outside network 12 and router RI which controls access to network 12. Source porting verifier 68 specifies a source port for data messages being sent to a computer on network 12 and, consequently, it communicates with transport layer 22 and network layer 24 of protocol stack 20 on the computer executing the program which implements system 60.

The process performed by the source routing verifier 66 is shown in FIG. 9. That process begins by obtaining a destination address for a computer on network 12 from source/destination address generator 34 (Block 300). The computer to which the message is to be ultimately delivered is defined by a destination address. The source address used to identify an intermediate source for a source routed message is also obtained from source/destination address generator 34 (Block 302). Source routing verifier 66 then passes the source and destination addresses to transport layer 22 (Block 306) to source route a message to a computer at the destination address on network 12 through the intermediate source identified by the source address (Block 310). If a response is detected by response message evaluator 64 to the source routed message (Block 312), a log indicating that the source routing blocking feature is not activated for the particular source/destination address combination is recorded in table 46 (Block 314). If another source address is available for another computer on the network (Block 316), it is obtained and another source routed message through the selected source address to the destination address is attempted. After attempts to source route messages to the destination address through all the source addresses for the other computers on the network have been attempted, the process determines if all destination addresses have been tested (Block 318). If another destination address is available, another destination address is obtained and the process is repeated using the addresses of the other computers on the network as source addresses for source routed messages to the next destination address. In this manner, a log of all the source routed combinations which are not being blocked by the router are recorded in table 46 so the router may be reconfigured.

FIG. 10 shows a process implemented by source porting verifier 68. The process begins by obtaining a destination address for a computer on the network from source/destination address generator 76 (Block 340). Preferably, a source port address which corresponds to the default FTP source port address, typically port address 20, is provided to network layer 24 (Block 342). Until it is changed, data messages from the computer executing the program which implements the process of FIG. 11 generates data messages having a source port address of 20. The destination port address is set to the first port address (Block 344) and a data message having a source port address of 20 is sent to the port of the computer at the destination address (BLOCK 348). Response evaluator 72 evaluates the responsive message received (Block 350), if any, to determine whether the port responded to the source ported data message. Each response is stored in table 46 (Block 354). The process determines if there is another destination port address (Block 358) and, if there is, the destination port address is incremented (Block 360). The process continues by checking the next destination port. If all the destination ports on the destination computer have been checked, the process determines if another source port address is to be tested (Block 364). If there is, the next source port address is obtained (Block 366) and the ports of the destination computer are tested with messages having the new source port address. Alternatively, all source port addresses may be exhaustively tested. If there are no more source port addresses to check, the process determines if another destination address exists on the network (Block 368). If it does, the next destination address is obtained (Block 340) and the process continues. Otherwise, the process stops.

A router may be configured with a rule which blocks data messages from computers external to network 12. However, another rule may permit messages with certain source port address values to pass through in order to support certain services such as FTP. FTP requires a source port address of 20. A hacker may attempt to get into a network by sending messages with a source port value which a router passes because it conforms to the rule for FTP messages. The process of FIG. 10 determines whether messages with predetermined source port addresses from computers external to the network are able to be received by computers on a network despite router configuration rules which would otherwise prevent the transmission of the messages.

As discussed above, Socks servers do not pass simply pass messages between computers on the network and those external to the network but instead require two separate communication connections. One communication connection is with an external computer and the other communication connection is with a computer on the network. In this manner, the Socks server may more thoroughly examine message in accordance with the rules configured for the server before passing the messages from one communication connection to another communication connection.

A preferred process implemented by the Socks configuration verifier of FIG. 8 is shown in FIG. 11. That process begins by having the computer executing the program which implements the process of FIG. 11 connect to the Socks server (Block 400). A destination address is then obtained from the source/destination address generator 34 and used to request that the Socks server connect to the computer on the network at the destination address (Block 402). The destination port address is set to the first port address value of the possible range of port address values (Block 406). A service command is then generated (Block 410) and a service command message addressed for the computer at the destination address is sent to the Socks server (Block 412). The process then waits for a response (Block 416). The response message is evaluated by response message generator 64 to determine if the response message indicates that the computer at the destination address received the service command (Block 420). If it did not, the process determines if another communication method is available (Block 424). If there is, the service command message is modified for another communication method (Block 426) and sent to the Socks server (Block 412). For example, if the message did not go through the Socks server, the service command message may be reformatted as a source routed message or a message with a predetermined source port value to see if the Socks server passes that type of message to the computer at the destination address. If no other communication format is available, the process continues by determining if another port address is available (Block 438).

If the message indicates that the computer on the network responded to the service command, the process determines whether the service command was executed (Block 430). If it was, the service and port address are stored in table 46 (Block 432). If the response message indicates that the service command was received but not executed, the process determines if another service command is available (Block 434). If there is, a new service command is generated (Block 410) and the process continues until all service commands have been attempted for the port address at the destination address computer. If no other service commands remain to be tried, an indicator is stored in table 46 which indicates communication was established with the port address but no service was executed (Block 432).

The process continues by determining if another port address remains for the computer at the destination address (Block 438). If one does, the port address is incremented (Block 440) and the testing for the new port address continues (Block 410). Otherwise, the process determines whether another destination address is available on the network (Block 444). If there is, it is obtained from source/destination address generator 34 (Block 402) and testing of the computer at the new destination address continues. Otherwise, the communication connection with the Socks server is terminated and the process stops.

A more preferred embodiment of the present invention is shown in FIG. 12. System 80 includes IP spoofing attack generator 32, communication initiation message generator 42, RPC message generator 52, communication message generator 62, source/destination address generator 34, topology table or log 46 and protocol stack 20 which operate in manner consistent with the description of the embodiments for those like numbered components discussed above. System 80 also includes response evaluator 82 which includes the functionality of response message evaluators 44, 54 and 64 as discussed above. A Graphic User Interface (GUI) 84 is also provided to accept input and control from a user and to display options and information to a user in a known manner. A user may use GUI 84 to activate each of the network verifiers 32, 42, 52 or 62 individually or selectively

identify a group of verifiers to automatically execute and build the information in table 46. GUI 84 also permits a user to enter information for execution of the verifiers such as defining or adding predetermined source port addresses, RPC services, addresses for computers added or deleted from a network or the like.

In operation, a user activates the program which implements an embodiment of the present invention such as system 80. As a result, GUI 84 may present options to the user such as modifying information for system operation, selection of one or more of the network verifiers or display of stored information. After the user makes a selection, system 80 then performs the requested option. For example, if the user selects the system information modification option, the user is permitted to change system information such as adding addresses for new computers on a network. GUI 84 then returns the user to the main option menu following completion of the input of data and the user may now select one or more network verifiers to run. GUI 84 then selectively activates the selected network verifiers which communicate with protocol stack 20 to communicate messages between the computer executing system 80 and a computer on the network being tested or a router or a Socks server coupled to the network. When the verification tests or scans are completed, the user may select the display option and either view or print the information The user may then use the displayed information to add authorization checks to services or new rules to a Socks server or router.

While the present invention has been illustrated by the description of a number of embodiments and while the embodiments have been described in considerable detail, it is not the intention of the applicant to restrict or any way limit the scope of the appended claims to such detail. Additional advantages and modifications will readily appear to those skilled in the art. The invention in its broader aspects is therefore not limited to the specific details, representative systems and methods, and illustrative examples shown and described. Accordingly, departures may be made from such details without departing from the spirit or scope of applicant's general inventive concept.

What is claimed is:

1. A system for detecting a security vulnerability in open network communications comprising:

an internet protocol (IP) spoofing attack generator for generating an IP spoofing attack on a target computer coupled to an open network to determine whether said target computer is vulnerable to an IP spoofing attack which emulates communication from another computer on said open network;

a service command message generator for generating a service command to be executed by a service coupled to a port on said target computer; and

said IP spoofing attack generator transmitting said service command to said target computer to generate a response in said target computer that provides a compromise indication without altering system operational parameters of said target computer.

2. The system of claim 1, wherein said generated service command is for one of an rsh and an rlogin service to determine whether authorization checks for said service exist.

3. The system of claim 2, wherein said generated service command causes said target computer to generate an electronic mail message indicative that said target computer has been compromised.

4. The system of claim 3, wherein said generated service command causes said target computer to initiate a Telnet

session with a computer which logs said Telnet session to indicate said target computer has been compromised.

5. The system of claim 1, further comprising:

a source/destination address generator which generates source and destination addresses for messages corresponding to an open network protocol used to communicate on said open network, said destination address corresponding to said target computer and said source address corresponding to said computer being emulated for said attack.

6. The system of claim 5, wherein said source/destination address generator generates source and destination address combinations which are used by said IP spoofing attack generator to test vulnerability of each computer in said open network to an IP spoofing attack which emulates communication from each of said other computers on said open network.

7. A system for generating a service topology map for each computer on an open network without completing a communication connection with any computer on the open network comprising:

a communication initiation message generator for generating communication initiation messages, said communication initiation messages being transmitted to ports on a computer on an open network; and

a response message evaluator for determining from response messages received from said ports receiving said communication initiation messages whether services exist on said ports receiving said communication initiation messages, said response messages not completing communication connections with said ports so that services coupled to said ports may be detected without completing communication connection with said ports.

8. The system of claim 7, further comprising:

a table for storing service indicators indicative of which ports responding to said communication initiation messages are coupled to services.

9. The system of claim 8, wherein said communication initiation message generator generates a communication initiation message for each port address on a computer on said open network.

10. The system of claim 9, wherein a source/destination address generator generates a destination address for each computer on an open network so that each port on each computer on said open network receives a communication initiation message and said table contains service indicators for each port of each computer on said open network which responds to said communication initiation messages.

11. The system of claim 7, wherein said communication initiation message generator generates sync messages for a TCP/IP protocol.

12. The system of claim 11, wherein said response message evaluator determines a service is coupled to a port receiving a communication initiation message in response to detecting a sync/ack message.

13. The system of claim 7, wherein said communication initiation message is the first message for a three handshake protocol to establish a communication connection.

14. A system for detecting vulnerability of ports coupled to remote procedure call (RPC) services on a computer of an open network comprising:

a remote procedure call (RPC) message generator for generating and sending RPC service commands to ports on a computer on an open network; and

a response message evaluator for evaluating response messages from said ports of said computer receiving said RPC service commands, said response messages indicating whether said RPC service commands were executed by an RPC service coupled to said ports of said computer receiving said RPC service commands without establishing a communication connection with said ports.

15. The system of claim 14, further comprising:

a table for storing port addresses and service indicators that indicate which particular RPC services are coupled to ports receiving said service commands.

16. A system for detecting vulnerabilities in routers comprising:

a communication message generator for generating and sending service commands from a computer external to an open network to ports on computers coupled to said open network through a router; and

a response message evaluator for evaluating response messages received from said ports on computers of said open network in response to said service commands sent from said communication message generator external to said open network whereby access to said computers on said open network through said router may be determined without referencing configuration files of said router.

17. The system of claim 16, wherein said communication message generator includes a source routing verifier for generating source routed messages with a destination address of a computer on said open network and an intermediate source address on said open network; and

said response message evaluator evaluating response messages received from said ports on computers of said open network in response to said service commands sent from said communication message generator external to said open network to detect a vulnerability in said router of permitting source routed messages to bypass rules configured for filtering inbound messages on said router.

18. The system of claim 17, wherein each source address for each computer on said open network is used as said intermediate source address with each destination address for each computer on said open network to test each possible intermediate source/destination address combination for source routed messages on said open network.

19. The system of claim 18, further comprising:

a table for storing indicators for each intermediate source address/destination address combination that is detected as being vulnerable to receiving source routed messages.

20. The system of claim 16, wherein said communication message generator includes a source porting verifier for generating service command messages with a source port address having a predetermined value; and

said response message evaluator evaluating response messages received from said ports on computers of said open network in response to said service command messages having said predetermined source port address values sent from said source porting verifier external to said open network to detect said router passing messages having said predetermined source port address values to ports coupled to services on said open network.

21. The system of claim 20, wherein service command messages having said predetermined source port address value are sent to each computer on said open network.

22. The system of claim 21, further comprising:

a table for storing service indicators for each computer address that is detected as being vulnerable to receiving source ported messages.

23. The system of claim 22, wherein said predetermined value corresponds to a default source port address for a file transfer protocol (FTP) message of a TCP/IP protocol.

24. The system of claim 16, further comprising:

a Socks configuration verifier for establishing a communication connection with a Socks server and for sending service command messages to computers on said open network coupled to said Socks server; and

said response message evaluator evaluating said messages received in response to said service command messages to determine whether said service command message was passed by said Socks server to one of said computers on said open network.

25. The system of claim 24 said response message evaluator determining whether said service command message was executed by said one computer on said open network.

26. The system of claim 25 said response message evaluator storing service indicators indicative of said services which executed said service command messages received at said port addresses.

27. A method for detecting a security vulnerability in an open network comprised of the steps of:

attempting an Internet Protocol (IP) spoofing attack against a target computer and open network;

generating a service command message; and

sending said service command message to said target computer following said IP spoofing attack to determine whether said target computer has been compromised, said service command message generating an indicator of the success of the IP spoofing attack without altering the operational parameters of the target computer.

28. The method of claim 27, wherein said generating service command message step generates one of an rsh and rlogin command.

29. The method of claim 28, wherein said generating step:

generates an electronic mail message indicative of the success of the IP spoofing attack in response to said service command message.

30. The method of claim 27, further comprising the step of:

initiating a Telnet session between said target computer and another computer to indicate the success of said IP spoofing attack in response to said service command message.

31. The method of claim 27, further comprising the steps of:

generating source addresses and destination addresses for said IP spoofing attack; and

attempting said IP spoofing attack against each said generated destination address by emulating communication from each of said source addresses.

32. A method for generating a service topology map of an open network comprising the steps of:

generating a communication command initiation message;

sending said communication command initiation message to a port on a computer on an open network;

receiving a message from said port in response to said communication initiation message being received at said port; and

evaluating said message received from said port to determine whether a service is coupled to said port without establishing a communication connection with said port.

33. The method of claim 32, further comprising the step of:

storing a service indicator to provide a reference that said port has a service coupled thereto which may be accessed from another computer.

34. A method for detecting availability of a service on a port of a computer on an open network comprising the steps of:

generating a service command message;

sending said generated service command message to a port of a computer on said open network;

receiving a message from said port in response to said port receiving said generated service command message; and

evaluating said message received from said port to determine whether a service coupled to said port executed said service command message, without establishing a communication connection with said ports.

35. The method of claim 34, further comprising the step of:

storing a service indicator indicative that said service coupled to said port executed said service command message.

36. The method of claim 35, wherein said generating step generates service command messages for different services; and

said evaluating step determines the type of service coupled to said port which executed said service command message.

37. The method of claim 36, wherein said generating step generates said service command messages for each port of a computer of said open network.

38. The method of claim 34, further comprising the steps of:

establishing a communication connection with a Socks server;

requesting said Socks server establish a communication connection with a computer on said open network; and

said evaluating step determining whether said Socks server is configured to stop said service command message from being sent to said port of said computer of said open network.

39. The method of claim 34, wherein said generating step generates remote procedure call (RPC) service command messages.

40. The method of claim 34, wherein said generating step generates service command messages having predetermined source port addresses.

41. The method of claim 34, wherein said generating step generates source routed service command messages.

* * * * *

US005805801A

United States Patent [19]

Holloway et al.

[11] **Patent Number: 5,805,801**

[45] **Date of Patent: Sep. 8, 1998**

[54] **SYSTEM AND METHOD FOR DETECTING AND PREVENTING SECURITY**

[75] Inventors: **Malcolm H. Holloway**, Durham; **Thomas Joseph Prorock**, Raleigh, both of N.C.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **780,804**

[22] Filed: **Jan. 9, 1997**

[51] **Int. Cl.**^{6} **G06F 11/00**

[52] **U.S. Cl.** **395/187.01**

[58] **Field of Search** 395/186, 187.01, 395/182.02, 200.55; 380/3, 25; 370/434, 488

[56] **References Cited**

U.S. PATENT DOCUMENTS

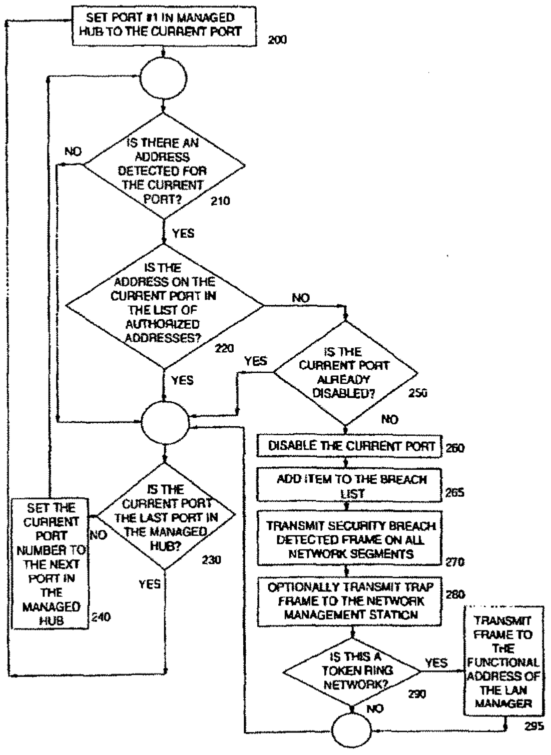
4,930,159	5/1990	Kravitz et al.	380/23
5,177,788	1/1993	Schanning et al.	380/23
5,305,385	4/1994	Schanning et al.	380/45
5,311,593	5/1994	Carmi	380/23
5,337,309	8/1994	Faulk, Jr.	
5,414,833	5/1995	Hershey et al.	380/4
5,440,723	8/1995	Arnold et al.	395/181
5,537,099	7/1996	Liang	340/825.07
5,606,668	2/1997	Shwed	395/200.11
5,610,981	3/1997	Mooney et al.	380/25
5,727,146	3/1998	Savoldi et al.	395/187.01

Primary Examiner—Albert Decady
Attorney, Agent, or Firm—John J. Timar

[57] **ABSTRACT**

A system and method for providing security against intrusion in a campus LAN network is provided. A managed hub discovers each interconnect device in the network that supports the security feature and maintains an interconnect device list of such devices, which may include token ring switches, Ethernet switches, bridges and routers. The managed hub detects an intrusion by an unauthorized address on one of its ports and notifies the interconnect devices of the intrusion by transmitting a security breach detected frame. The interconnect devices set a filter on their respective ports against the intruding unauthorized address. The interconnect devices send a filter set frame to the managed hub which reenables the port where the security intrusion occurred, after all filter set frames are received. A network management station sends a security clear condition frame to remove the filters.

59 Claims, 16 Drawing Sheets


SET PORT #1 IN MANAGED HUB TO THE CURRENT PORT
200
IS THERE AN ADDRESS DETECTED FOR THE CURRENT PORT?
210
NO
YES
IS THE ADDRESS ON THE CURRENT PORT IN THE LIST OF AUTHORIZED ADDRESSES?
220
NO
YES
IS THE CURRENT PORT ALREADY DISABLED?
250
YES
NO
DISABLE THE CURRENT PORT
260
ADD ITEM TO THE BREACH LIST
265
TRANSMIT SECURITY BREACH DETECTED FRAME ON ALL NETWORK SEGMENTS
270
OPTIONALLY TRANSMIT TRAP FRAME TO THE NETWORK MANAGEMENT STATION
280
IS THIS A TOKEN RING NETWORK?
290
YES
NO
TRANSMIT FRAME TO THE FUNCTIONAL ADDRESS OF THE LAN MANAGER
295
IS THE CURRENT PORT THE LAST PORT IN THE MANAGED HUB?
230
NO
YES
SET THE CURRENT PORT NUMBER TO THE NEXT PORT IN THE MANAGED HUB
240

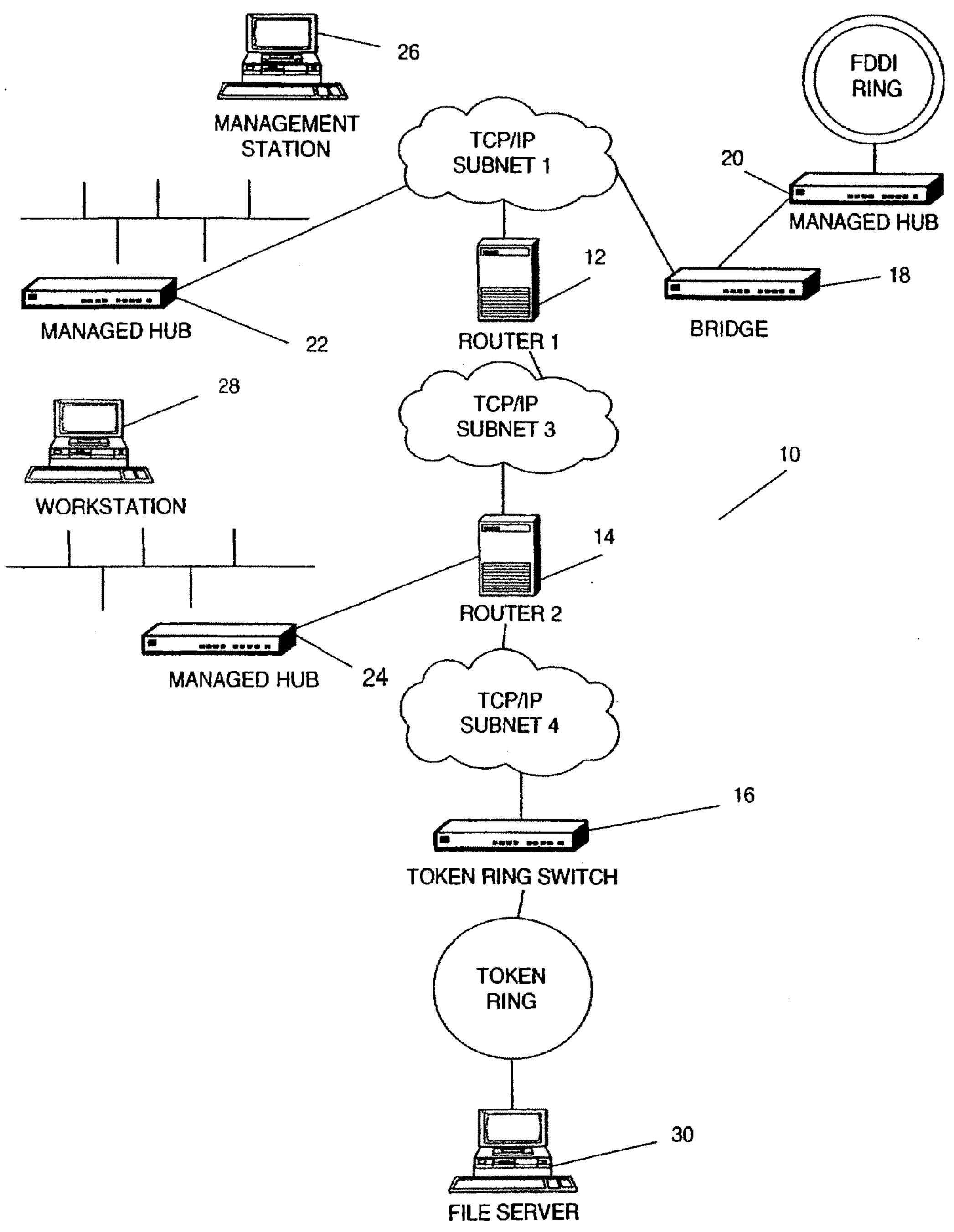

26
MANAGEMENT STATION
TCP/IP SUBNET 1
FDDI RING
20
MANAGED HUB
12
18
MANAGED HUB
22
ROUTER 1
BRIDGE
28
TCP/IP SUBNET 3
10
WORKSTATION
14
ROUTER 2
MANAGED HUB
24
TCP/IP SUBNET 4
16
TOKEN RING SWITCH
TOKEN RING
30
FILE SERVER

FIG. 1

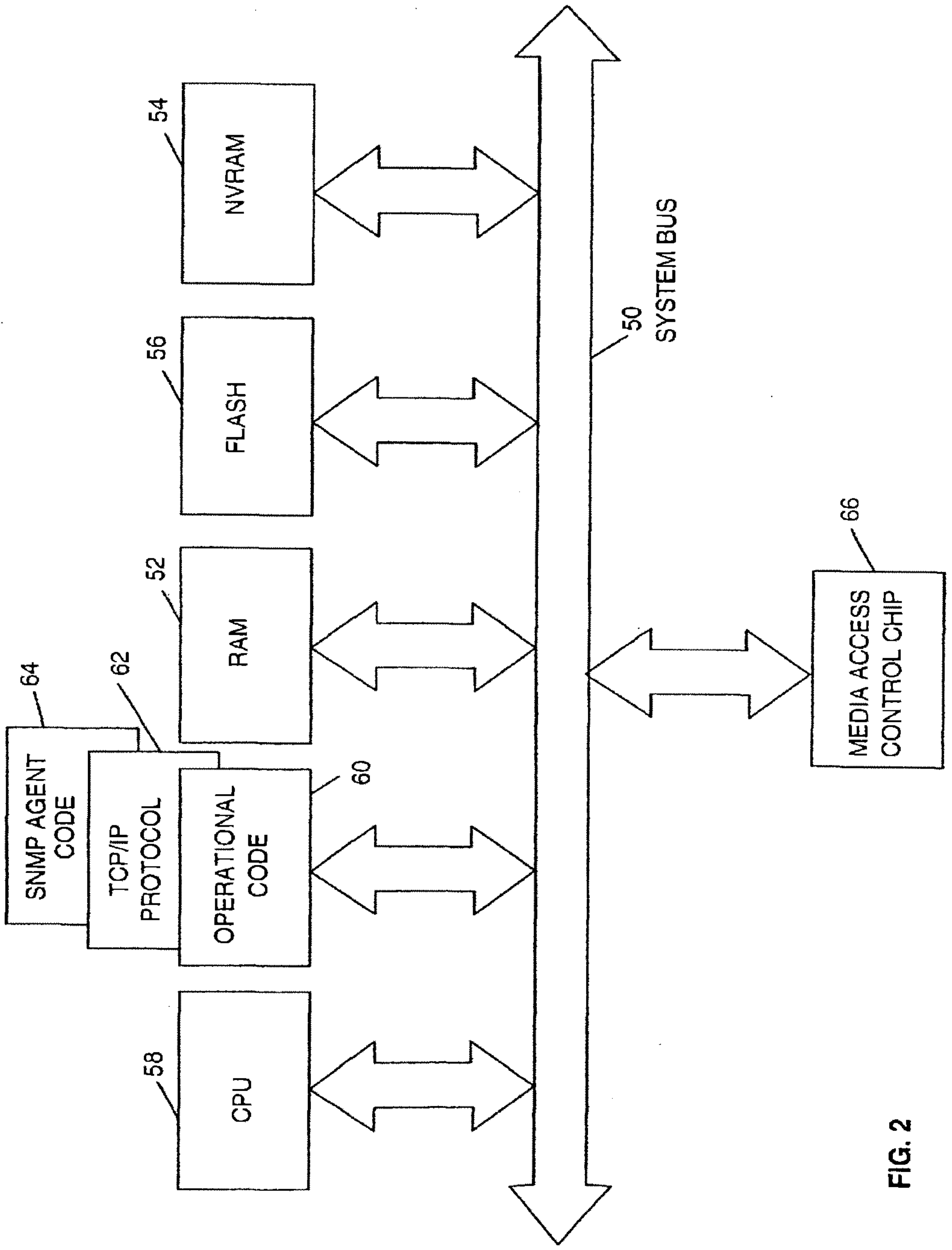
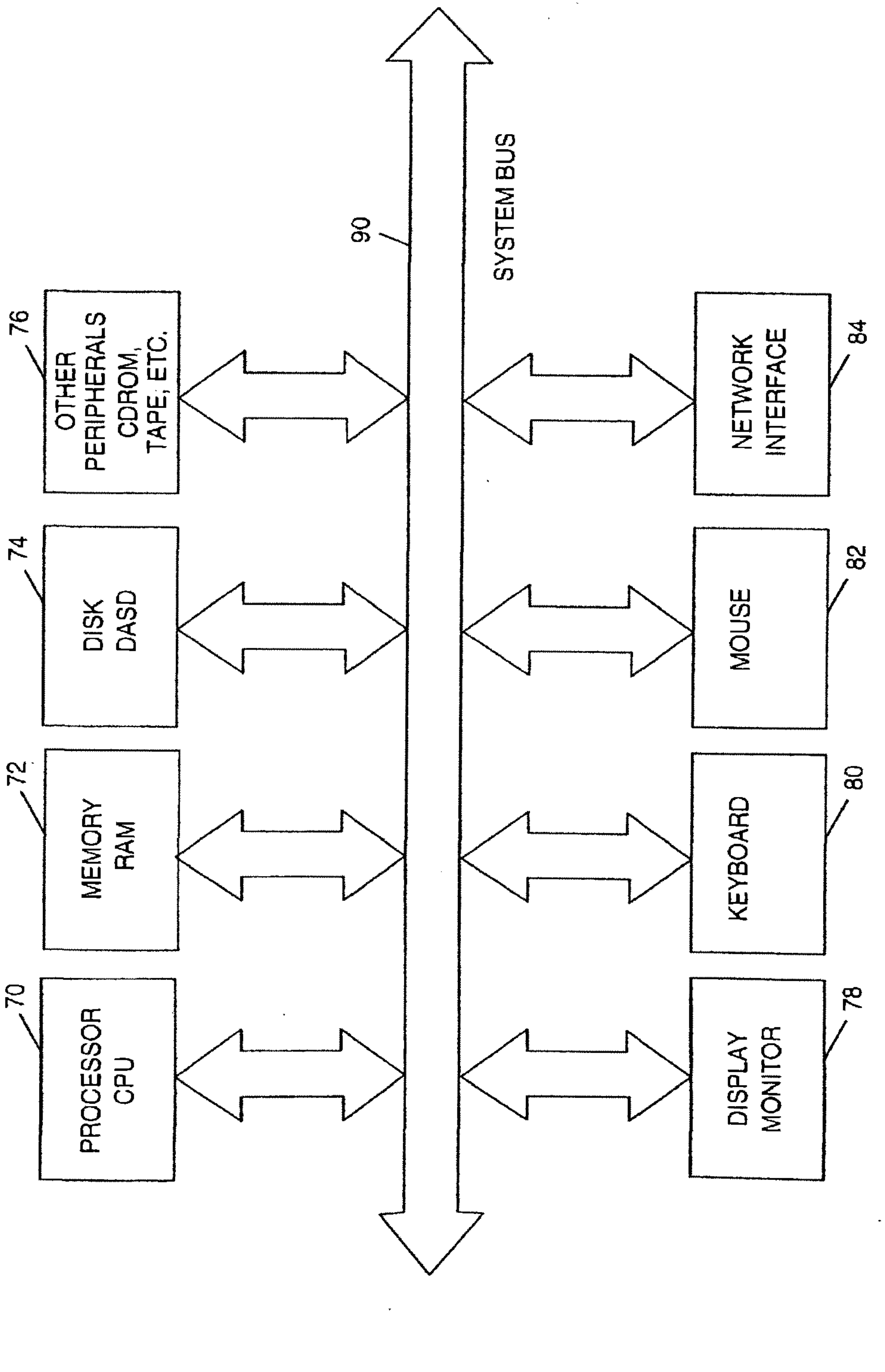

54
NVRAM
56
FLASH
52
RAM
64
SNMP AGENT CODE
62
TCP/IP PROTOCOL
60
OPERATIONAL CODE
58
CPU
50
SYSTEM BUS
66
MEDIA ACCESS CONTROL CHIP

FIG. 2



SYSTEM BUS
90
OTHER PERIPHERALS CDROM, TAPE, ETC.
76
DISK DASD
74
MEMORY RAM
72
PROCESSOR CPU
70
NETWORK INTERFACE
84
MOUSE
82
KEYBOARD
80
DISPLAY MONITOR
78

FIG. 3

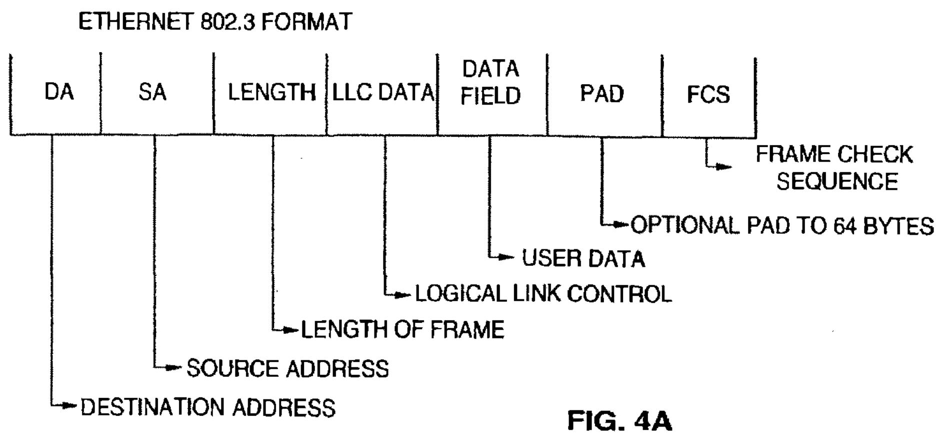


FIG. 4A

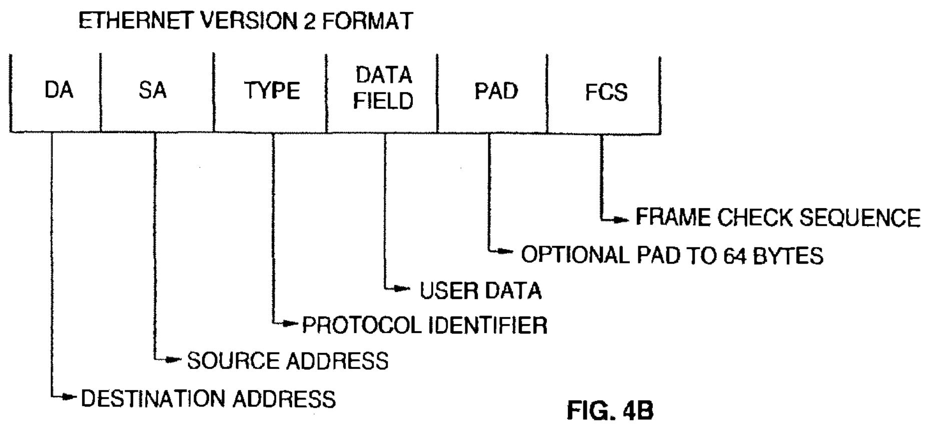


FIG. 4B

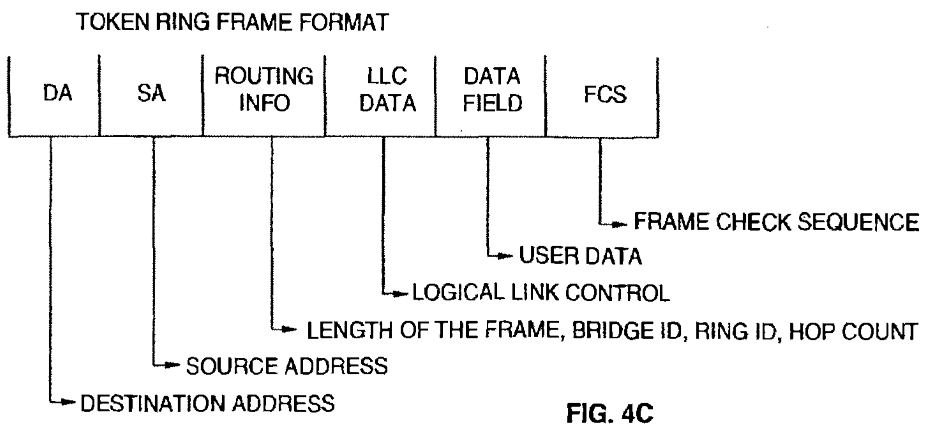


FIG. 4C

DISCOVERY REQUEST

FRAME TYPE IDENTIFIER	TIME STAMP	
1	4	BYTES

FIG. 5A

DISCOVERY RESPONSE

FRAME TYPE IDENTIFIER	INTERCONNECT DEVICE MAC ADDRESS	INTERCONNECT DEVICE DESCRIPTION	TIME STAMP	
1	6	50	4	BYTES

FIG. 5B

SECURITY BREACH DETECTED FRAME

	FRAME TYPE IDENTIFIER	INTRUDING MAC ADDRESS	MODULE NUMBER	PORT NUMBER	TIME STAMP	DEVICE FIELD LENGTH	ADDRESSES
BYTES	1	6	1	1	4	2	VARIABLE LENGTH

FIG. 5C

FILTER SET FRAME

FRAME TYPE IDENTIFIER	INTERCONNECT DEVICE MAC ADDRESS	INTRUDING MAC ADDRESS	MODULE NUMBER	PORT NUMBER	TIME STAMP	
1	6	6	1	1	4	BYTES

FIG. 5D

SECUITY CLEAR CONDITION

FRAME TYPE IDENTIFIER	INTRUDING MAC ADDRESS	
1	6	BYTES

FIG. 5E

INTERCONNECT DEVICE LIST ITEM			
MAC ADDRESS	DEVICE DESCRIPTION	LAST RESPONSE TIME	OUTSTANDING BREACH RESPONSE COUNT

MAC ADDRESS:	MAC ADDRESS OF THE INTERCONNECT DEVICE
DEVICE DESCRIPTION:	ASCII SELF DESCRIPTION PROVIDED BY THE INTERCONNECT DEVICE
LAST RESPONSE TIME:	TIME WHEN LAST RESPONSE RECEIVED FROM INTERCONNECT DEVICE
OUTSTANDING BREACH RESPONSE COUNT:	NUMBER OF SECURITY BREACH FRAMES THE INTERCONNECT DEVICE HAS NOT RESPONDED TO

FIG. 6

BREACH LIST ITEM				
MAC ADDRESS	BREACH TIME	BREACH PORT	BREACH MODULE	OUTSTANDING FILTER SET COUNT

MAC ADDRESS:	MAC ADDRESS OF THE INTRUDING DEVICE
BREACH TIME:	TIME WHEN INTRUSION OCCURED
BREACH PORT:	PORT IN MANAGED HUB WHEN INTRUSION OCCURRED
BREACH MODULE:	MODULE IN MANAGED HUB WHEN INTRUSION OCCURRED
OUTSTANDING FILTER SET COUNT:	NUMBER OF FILTER SET FRAMES NOT RECEIVED YET

FIG. 7

INTRUSION LIST ITEM			
MAC ADDRESS	BREACH TIME	BREACH PORT	BREACH MODULE

MAC ADDRESS:	MAC ADDRESS OF INTRUDING DEVICE
BREACH TIME:	TIME WHEN INTRUSION OCCURRED
BREACH PORT:	PORT IN MANAGED HUB WHEN INTRUSION OCCURRED
BREACH MODULE:	MODULE IN MANAGED HUB WHEN INTRUSION OCCURRED

FIG. 8

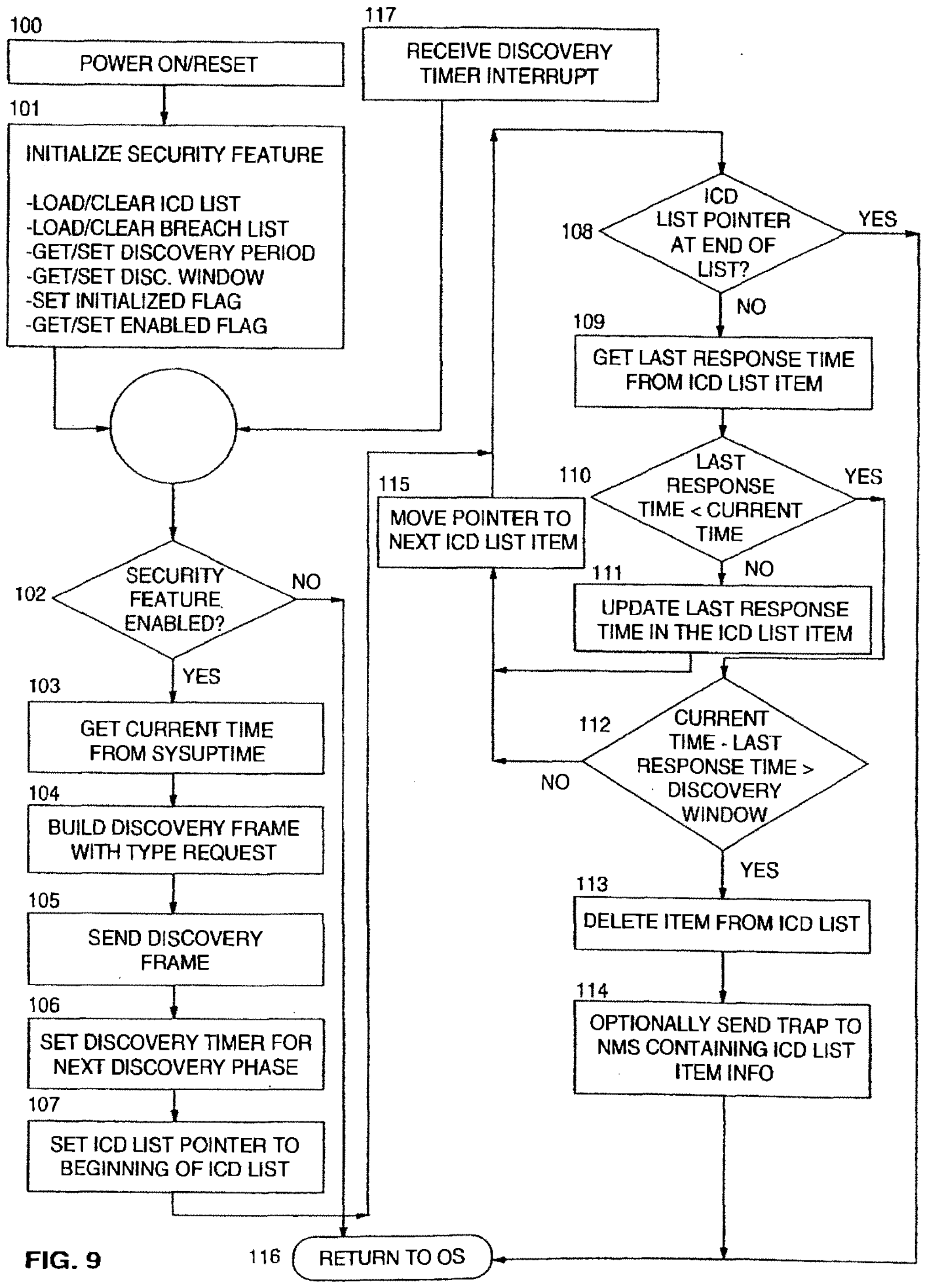

100
POWER ON/RESET
101
INITIALIZE SECURITY FEATURE
-LOAD/CLEAR ICD LIST
-LOAD/CLEAR BREACH LIST
-GET/SET DISCOVERY PERIOD
-GET/SET DISC. WINDOW
-SET INITIALIZED FLAG
-GET/SET ENABLED FLAG
117
RECEIVE DISCOVERY TIMER INTERRUPT
102
SECURITY FEATURE ENABLED?
NO
YES
103
GET CURRENT TIME FROM SYSUPTIME
104
BUILD DISCOVERY FRAME WITH TYPE REQUEST
105
SEND DISCOVERY FRAME
106
SET DISCOVERY TIMER FOR NEXT DISCOVERY PHASE
107
SET ICD LIST POINTER TO BEGINNING OF ICD LIST
108
ICD LIST POINTER AT END OF LIST?
YES
NO
109
GET LAST RESPONSE TIME FROM ICD LIST ITEM
110
LAST RESPONSE TIME < CURRENT TIME
YES
NO
115
MOVE POINTER TO NEXT ICD LIST ITEM
111
UPDATE LAST RESPONSE TIME IN THE ICD LIST ITEM
112
CURRENT TIME - LAST RESPONSE TIME > DISCOVERY WINDOW
NO
YES
113
DELETE ITEM FROM ICD LIST
114
OPTIONALLY SEND TRAP TO NMS CONTAINING ICD LIST ITEM INFO
116
RETURN TO OS

FIG. 9

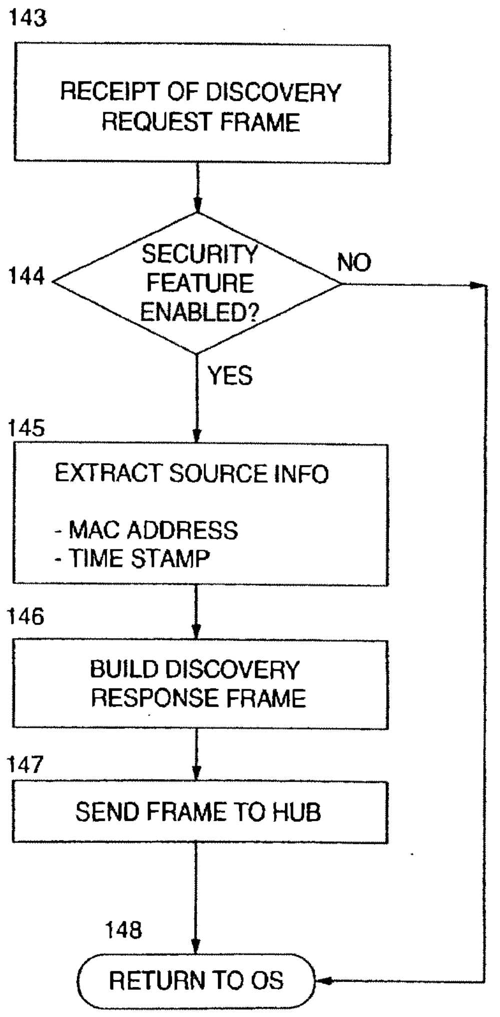
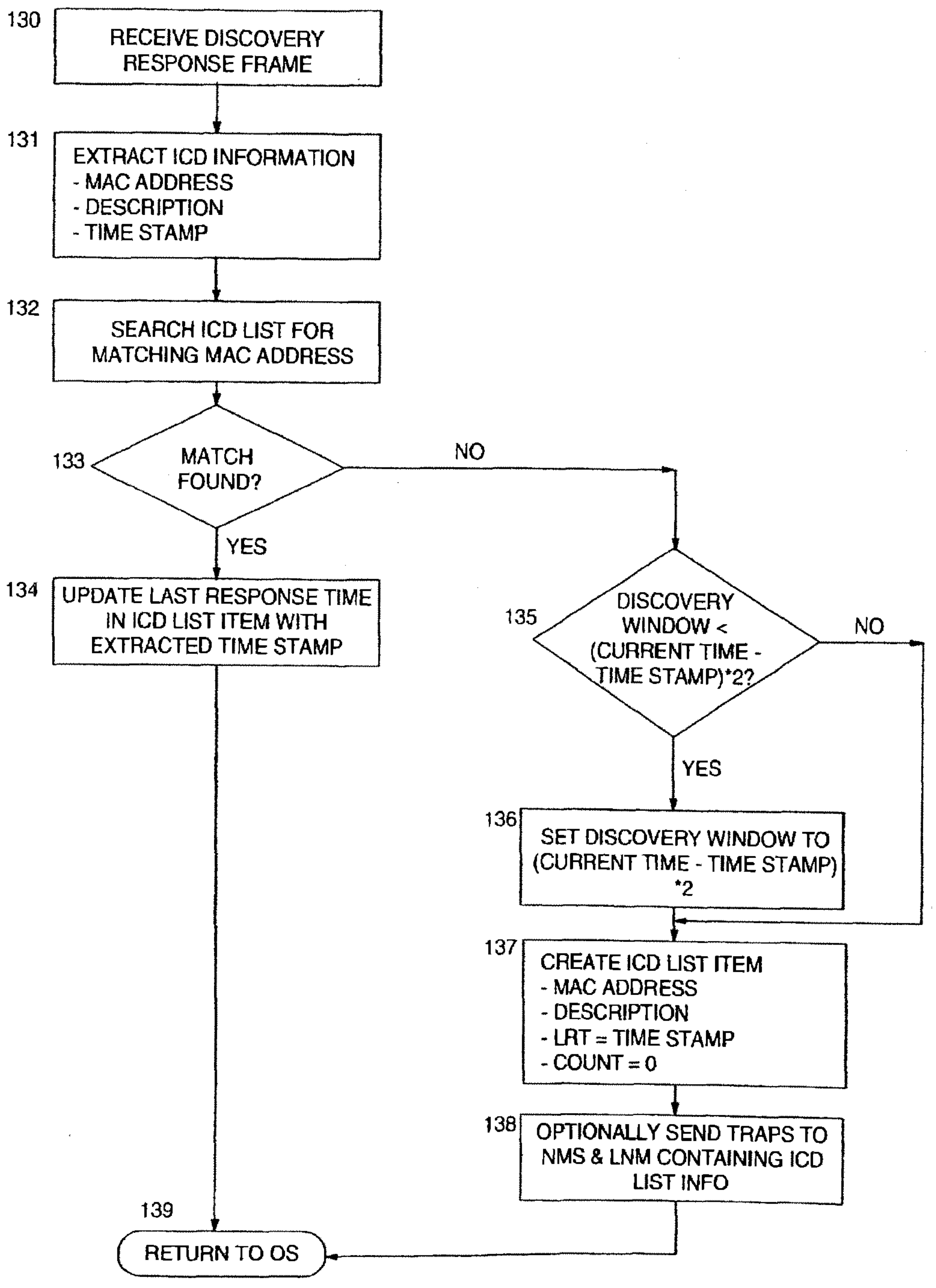

143
RECEIPT OF DISCOVERY REQUEST FRAME
144
SECURITY FEATURE ENABLED?
NO
YES
145
EXTRACT SOURCE INFO
- MAC ADDRESS
- TIME STAMP
146
BUILD DISCOVERY RESPONSE FRAME
147
SEND FRAME TO HUB
148
RETURN TO OS

FIG. 10



130 RECEIVE DISCOVERY RESPONSE FRAME
131 EXTRACT ICD INFORMATION - MAC ADDRESS - DESCRIPTION - TIME STAMP
132 SEARCH ICD LIST FOR MATCHING MAC ADDRESS
133 MATCH FOUND?
NO
YES
134 UPDATE LAST RESPONSE TIME IN ICD LIST ITEM WITH EXTRACTED TIME STAMP
135 DISCOVERY WINDOW < (CURRENT TIME - TIME STAMP)*2?
NO
YES
136 SET DISCOVERY WINDOW TO (CURRENT TIME - TIME STAMP) *2
137 CREATE ICD LIST ITEM - MAC ADDRESS - DESCRIPTION - LRT = TIME STAMP - COUNT = 0
138 OPTIONALLY SEND TRAPS TO NMS & LNM CONTAINING ICD LIST INFO
139 RETURN TO OS

FIG. 11

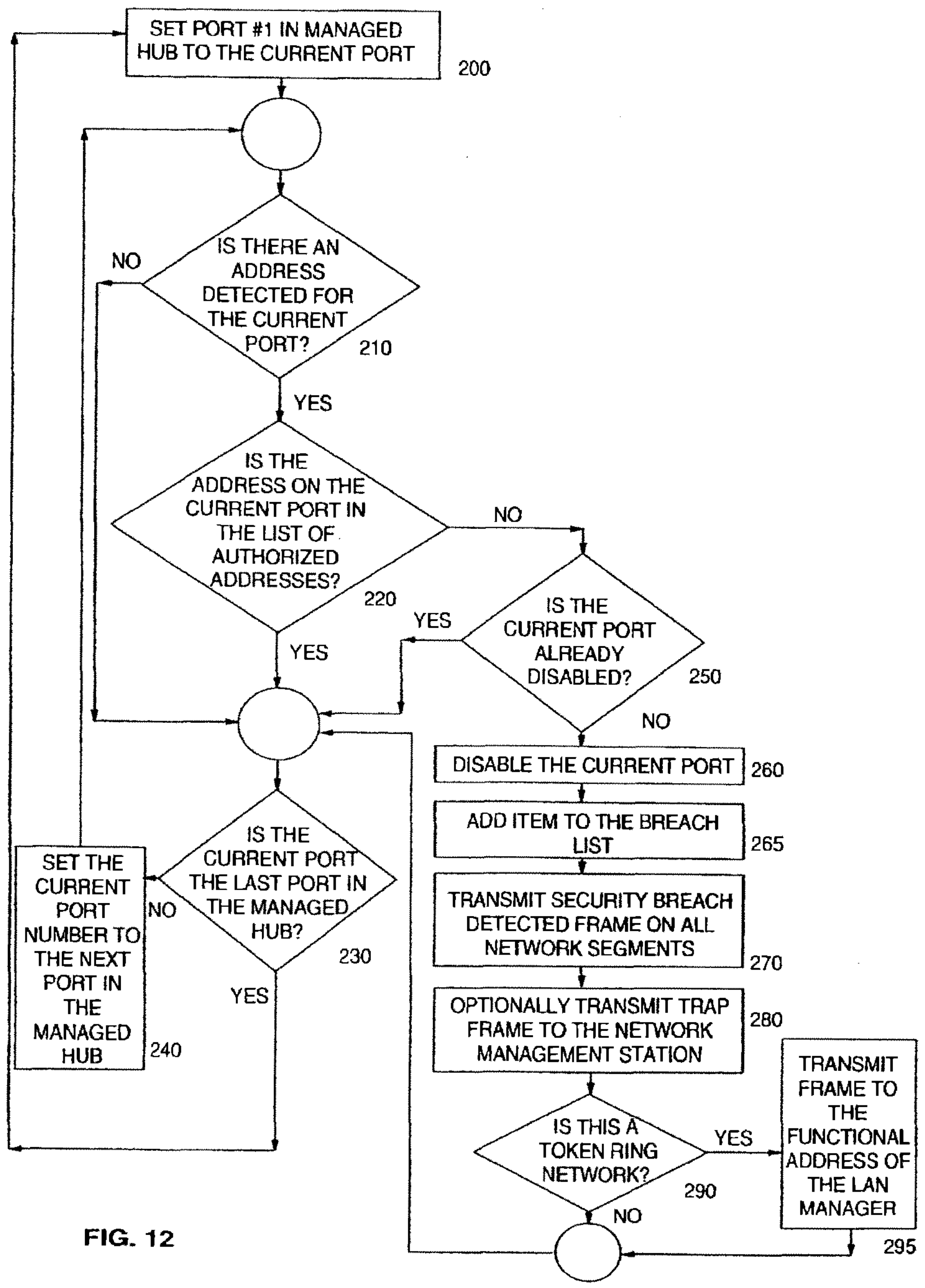

SET PORT #1 IN MANAGED HUB TO THE CURRENT PORT
200
IS THERE AN ADDRESS DETECTED FOR THE CURRENT PORT?
210
NO
YES
IS THE ADDRESS ON THE CURRENT PORT IN THE LIST OF AUTHORIZED ADDRESSES?
220
NO
YES
IS THE CURRENT PORT ALREADY DISABLED?
250
YES
NO
DISABLE THE CURRENT PORT
260
ADD ITEM TO THE BREACH LIST
265
TRANSMIT SECURITY BREACH DETECTED FRAME ON ALL NETWORK SEGMENTS
270
OPTIONALLY TRANSMIT TRAP FRAME TO THE NETWORK MANAGEMENT STATION
280
IS THIS A TOKEN RING NETWORK?
290
YES
NO
TRANSMIT FRAME TO THE FUNCTIONAL ADDRESS OF THE LAN MANAGER
295
IS THE CURRENT PORT THE LAST PORT IN THE MANAGED HUB?
230
NO
YES
SET THE CURRENT PORT NUMBER TO THE NEXT PORT IN THE MANAGED HUB
240

FIG. 12

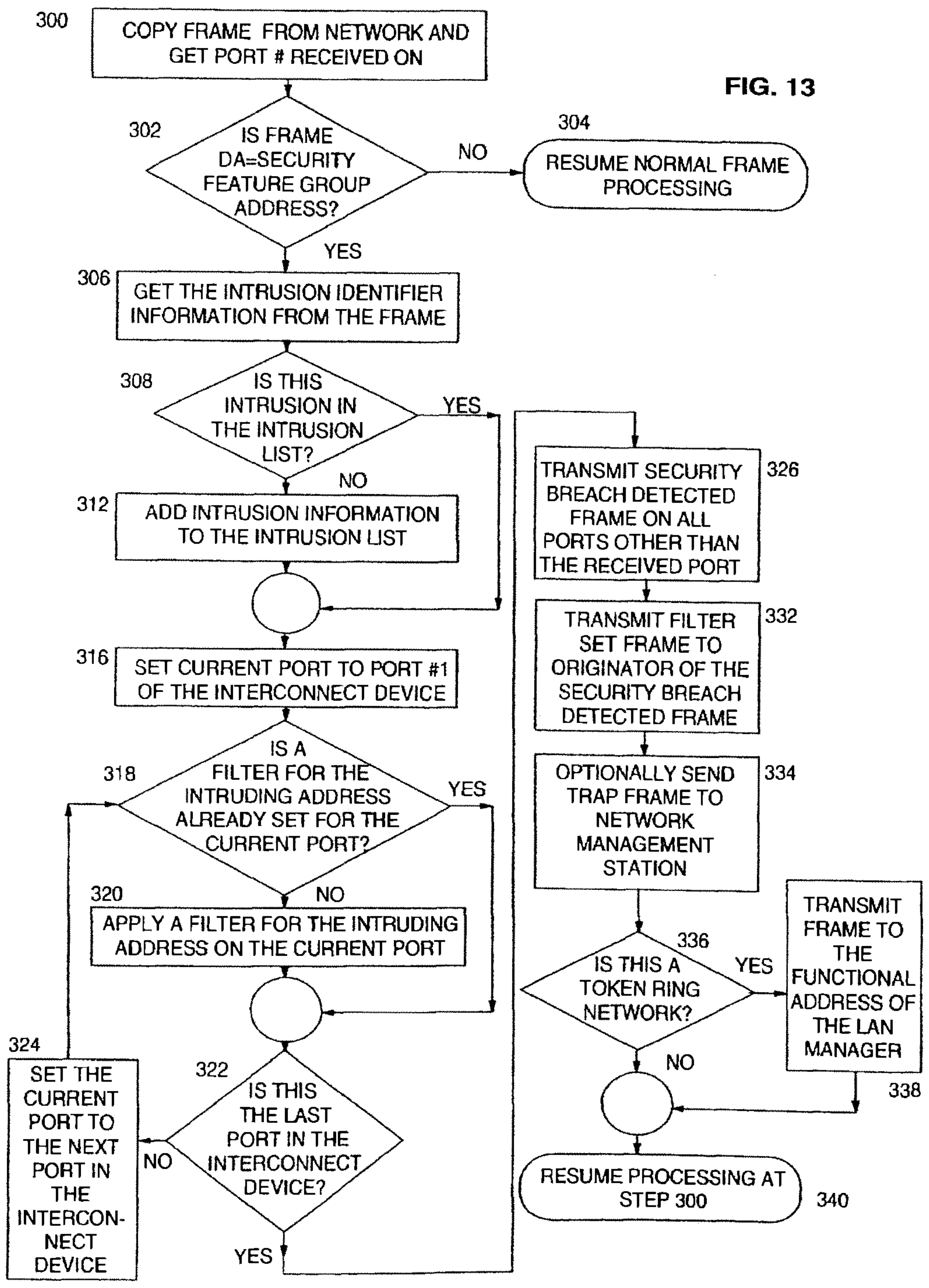
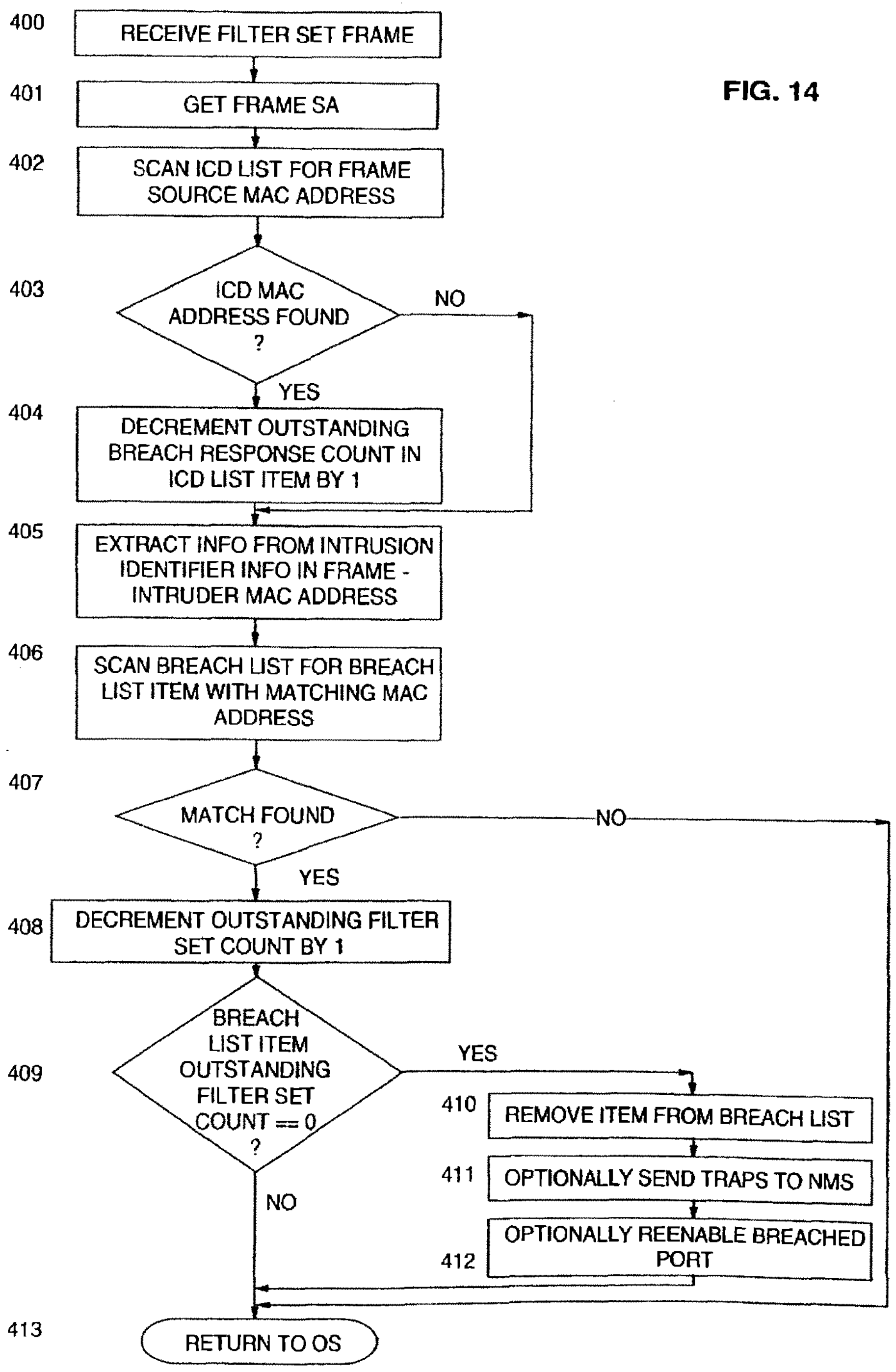

300
COPY FRAME FROM NETWORK AND GET PORT # RECEIVED ON
302
IS FRAME DA=SECURITY FEATURE GROUP ADDRESS?
NO
304
RESUME NORMAL FRAME PROCESSING
YES
306
GET THE INTRUSION IDENTIFIER INFORMATION FROM THE FRAME
308
IS THIS INTRUSION IN THE INTRUSION LIST?
YES
NO
312
ADD INTRUSION INFORMATION TO THE INTRUSION LIST
316
SET CURRENT PORT TO PORT #1 OF THE INTERCONNECT DEVICE
318
IS A FILTER FOR THE INTRUDING ADDRESS ALREADY SET FOR THE CURRENT PORT?
YES
320
NO
APPLY A FILTER FOR THE INTRUDING ADDRESS ON THE CURRENT PORT
324
SET THE CURRENT PORT TO THE NEXT PORT IN THE INTERCON-NECT DEVICE
322
IS THIS THE LAST PORT IN THE INTERCONNECT DEVICE?
NO
YES
TRANSMIT SECURITY BREACH DETECTED FRAME ON ALL PORTS OTHER THAN THE RECEIVED PORT
326
TRANSMIT FILTER SET FRAME TO ORIGINATOR OF THE SECURITY BREACH DETECTED FRAME
332
OPTIONALLY SEND TRAP FRAME TO NETWORK MANAGEMENT STATION
334
336
IS THIS A TOKEN RING NETWORK?
YES
TRANSMIT FRAME TO THE FUNCTIONAL ADDRESS OF THE LAN MANAGER
NO
338
RESUME PROCESSING AT STEP 300
340

FIG. 13



400 RECEIVE FILTER SET FRAME
401 GET FRAME SA
402 SCAN ICD LIST FOR FRAME SOURCE MAC ADDRESS
403 ICD MAC ADDRESS FOUND ?
NO
YES
404 DECREMENT OUTSTANDING BREACH RESPONSE COUNT IN ICD LIST ITEM BY 1
405 EXTRACT INFO FROM INTRUSION IDENTIFIER INFO IN FRAME - INTRUDER MAC ADDRESS
406 SCAN BREACH LIST FOR BREACH LIST ITEM WITH MATCHING MAC ADDRESS
407 MATCH FOUND ?
NO
YES
408 DECREMENT OUTSTANDING FILTER SET COUNT BY 1
409 BREACH LIST ITEM OUTSTANDING FILTER SET COUNT == 0 ?
YES
NO
410 REMOVE ITEM FROM BREACH LIST
411 OPTIONALLY SEND TRAPS TO NMS
412 OPTIONALLY REENABLE BREACHED PORT
413 RETURN TO OS

FIG. 14

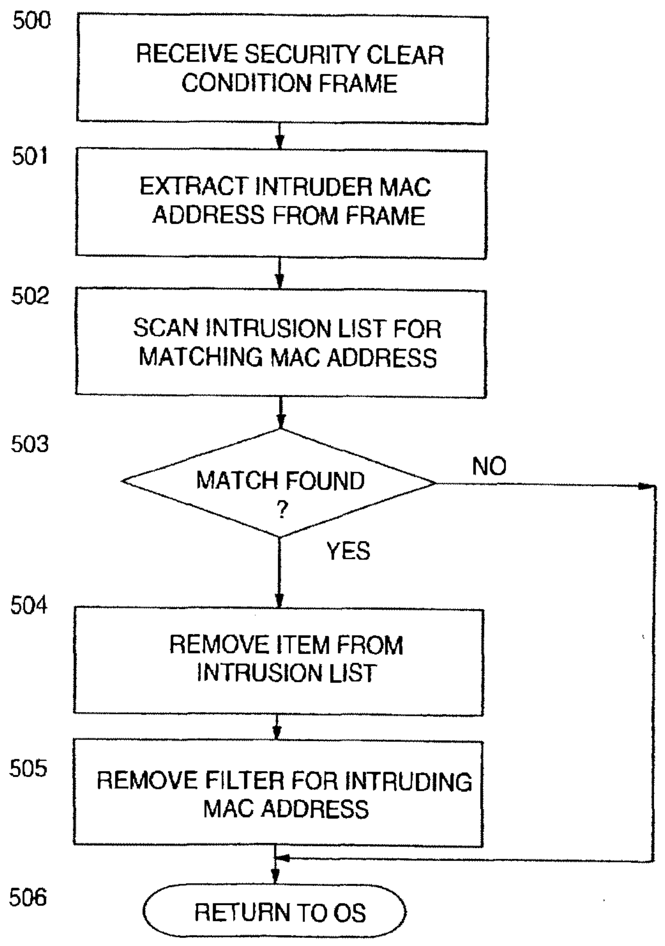

500
RECEIVE SECURITY CLEAR CONDITION FRAME
501
EXTRACT INTRUDER MAC ADDRESS FROM FRAME
502
SCAN INTRUSION LIST FOR MATCHING MAC ADDRESS
503
MATCH FOUND ?
NO
YES
504
REMOVE ITEM FROM INTRUSION LIST
505
REMOVE FILTER FOR INTRUDING MAC ADDRESS
506
RETURN TO OS

FIG. 15

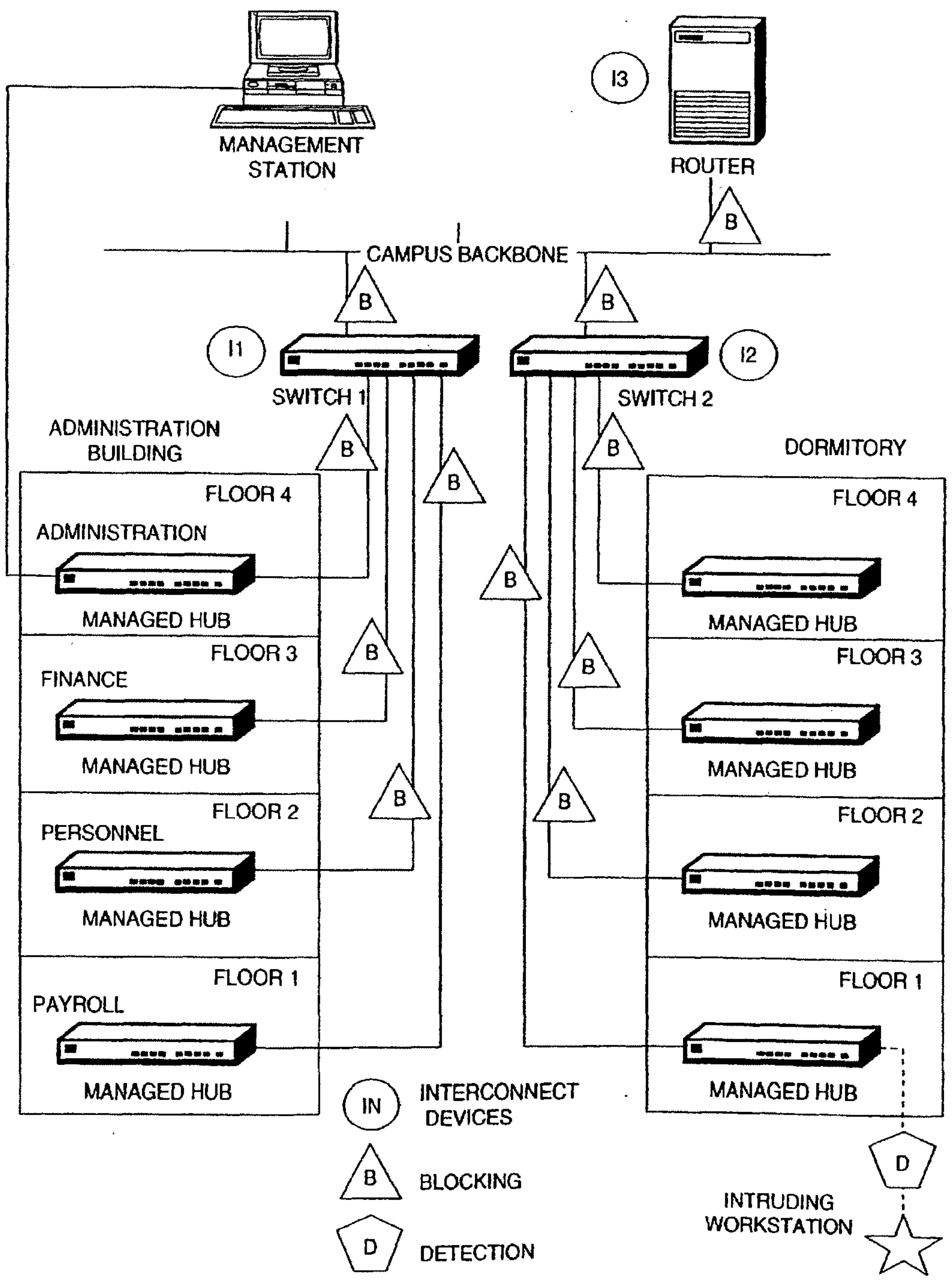

MANAGEMENT STATION
I3
ROUTER
B
CAMPUS BACKBONE
B
B
I1
SWITCH 1
SWITCH 2
I2
ADMINISTRATION BUILDING
DORMITORY
B
B
B
FLOOR 4
ADMINISTRATION
MANAGED HUB
FLOOR 3
FINANCE
MANAGED HUB
FLOOR 2
PERSONNEL
MANAGED HUB
FLOOR 1
PAYROLL
MANAGED HUB
B
B
B
B
B
FLOOR 4
MANAGED HUB
FLOOR 3
MANAGED HUB
FLOOR 2
MANAGED HUB
FLOOR 1
MANAGED HUB
IN
INTERCONNECT DEVICES
B
BLOCKING
D
DETECTION
D
INTRUDING WORKSTATION

FIG. 16

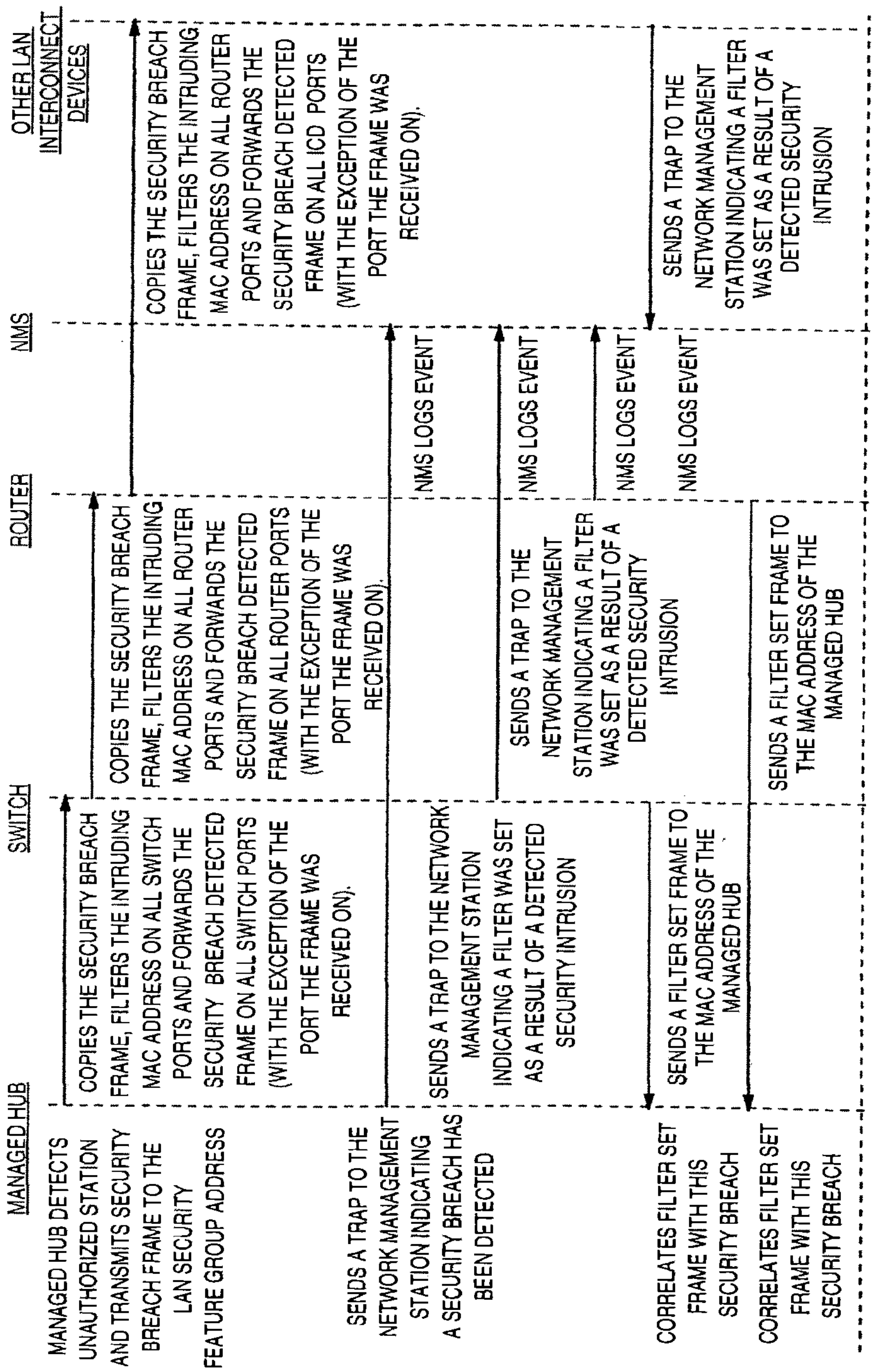

MANAGED HUB
SWITCH
ROUTER
NMS
OTHER LAN INTERCONNECT DEVICES
MANAGED HUB DETECTS UNAUTHORIZED STATION AND TRANSMITS SECURITY BREACH FRAME TO THE LAN SECURITY FEATURE GROUP ADDRESS
COPIES THE SECURITY BREACH FRAME, FILTERS THE INTRUDING MAC ADDRESS ON ALL SWITCH PORTS AND FORWARDS THE SECURITY BREACH DETECTED FRAME ON ALL SWITCH PORTS (WITH THE EXCEPTION OF THE PORT THE FRAME WAS RECEIVED ON).
COPIES THE SECURITY BREACH FRAME, FILTERS THE INTRUDING MAC ADDRESS ON ALL ROUTER PORTS AND FORWARDS THE SECURITY BREACH DETECTED FRAME ON ALL ROUTER PORTS (WITH THE EXCEPTION OF THE PORT THE FRAME WAS RECEIVED ON).
COPIES THE SECURITY BREACH FRAME, FILTERS THE INTRUDING MAC ADDRESS ON ALL ROUTER PORTS AND FORWARDS THE SECURITY BREACH DETECTED FRAME ON ALL ICD PORTS (WITH THE EXCEPTION OF THE PORT THE FRAME WAS RECEIVED ON).
SENDS A TRAP TO THE NETWORK MANAGEMENT STATION INDICATING A SECURITY BREACH HAS BEEN DETECTED
NMS LOGS EVENT
SENDS A TRAP TO THE NETWORK MANAGEMENT STATION INDICATING A FILTER WAS SET AS A RESULT OF A DETECTED SECURITY INTRUSION
NMS LOGS EVENT
SENDS A TRAP TO THE NETWORK MANAGEMENT STATION INDICATING A FILTER WAS SET AS A RESULT OF A DETECTED SECURITY INTRUSION
NMS LOGS EVENT
SENDS A TRAP TO THE NETWORK MANAGEMENT STATION INDICATING A FILTER WAS SET AS A RESULT OF A DETECTED SECURITY INTRUSION
NMS LOGS EVENT
CORRELATES FILTER SET FRAME WITH THIS SECURITY BREACH
SENDS A FILTER SET FRAME TO THE MAC ADDRESS OF THE MANAGED HUB
CORRELATES FILTER SET FRAME WITH THIS SECURITY BREACH
SENDS A FILTER SET FRAME TO THE MAC ADDRESS OF THE MANAGED HUB

FIG. 17A

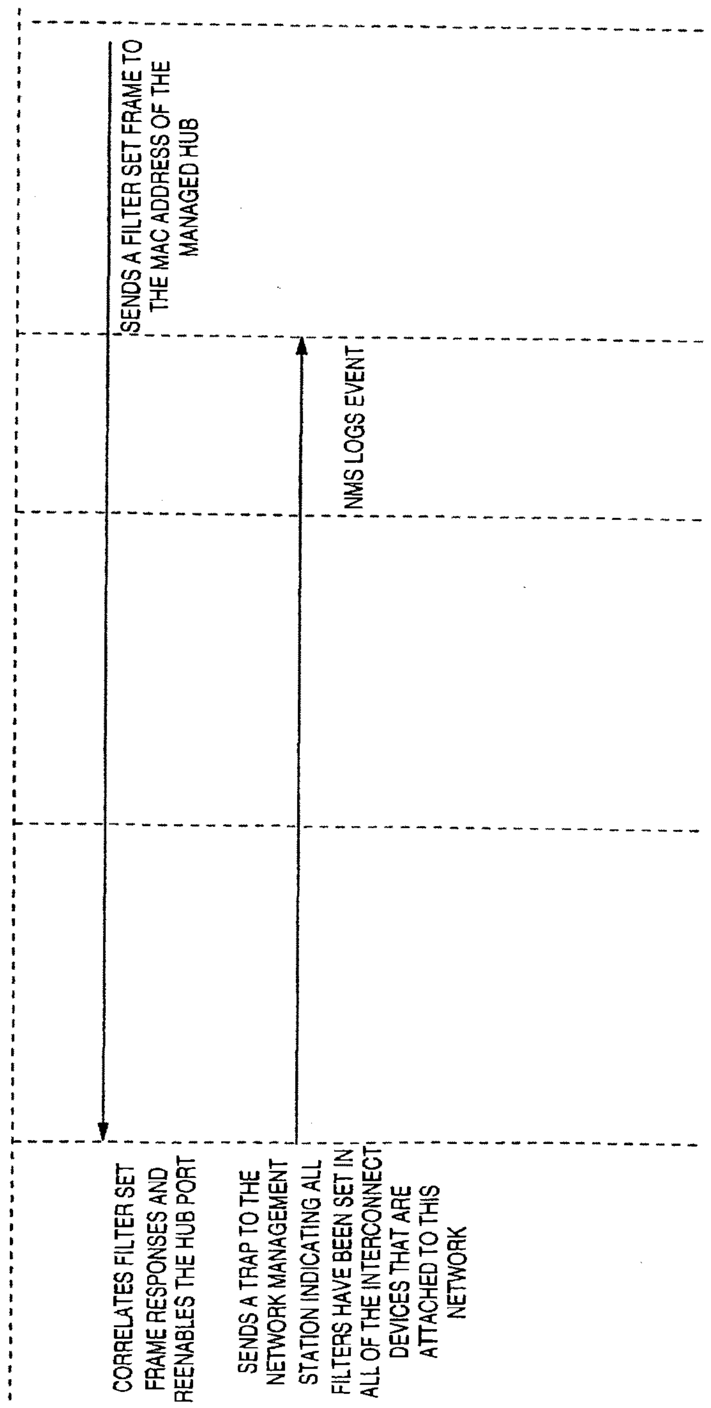

SENDS A FILTER SET FRAME TO THE MAC ADDRESS OF THE MANAGED HUB
NMS LOGS EVENT
CORRELATES FILTER SET FRAME RESPONSES AND REENABLES THE HUB PORT
SENDS A TRAP TO THE NETWORK MANAGEMENT STATION INDICATING ALL FILTERS HAVE BEEN SET IN ALL OF THE INTERCONNECT DEVICES THAT ARE ATTACHED TO THIS NETWORK

FIG. 17B

SYSTEM AND METHOD FOR DETECTING AND PREVENTING SECURITY

REFERENCE TO RELATED APPLICATION

This application is related to the following application having the same assignee and inventorship and containing common disclosure, and is believed to have an identical effective filing date: "Managed Network Device Security Method and Apparatus", U.S. application Ser. No. 08/775,536 filed Jan. 7, 1997.

BACKGROUND OF THE INVENTION

This invention relates in general to computer network security systems and in particular to systems and methods for detecting and preventing intrusion into a campus local area network by an unauthorized user.

As local area networks (LANs) continue to proliferate, and the number of personal computers (PCs) connected to LANs continue to grow at a rapid pace, network security becomes an ever increasing problem for network administrators. As the trend of deploying distributed LANs continues, this provides multiple access points to an enterprise's network. Each of these distributed access points, if not controlled, is a potential security risk to the network.

To further illustrate the demand for improved network security, an IDC report on network management, "LAN Management: The Pivotal Role of Intelligent Hubs", published in 1993, highlighted the importance of network security to LAN administrators. When asked the importance of improving management of specific LAN devices, 75% of the respondents stated network security is very important. When further asked about the growing importance of network security over the next three years, many respondents indicated that it would increase in importance.

More recently, a request for proposal from the U. S. Federal Reserve specified a requirement that a LAN hub must detect an unauthorized station at the port level and disable the port within a 10-second period. Although this requirement will stop an intruder, there is an inherent weakness in this solution in that it only isolates the security intrusion to the port of entry. The rest of the campus network is unaware of an attempted break-in. The detection of the unauthorized station and the disabling of the port is the first reaction to a security intrusion, but many significant enhancements can be made to provide a network-wide security mechanism. Where the above solution stops at the hub/port level, this invention provides significant enhancements to solving the problem of network security by presenting a system wide solution to detecting and preventing security intrusions in a campus LAN environment.

In today's environment, network administrators focus their attention on router management, hub management, server management, and switch management, with the goals of ensuring network up time and managing growth (capacity planning). Security is often an afterthought and at best administrators get security as a by-product of employing other device functions. For example, network administrators may set filters at router, switch, or bridge ports for performance improvements and implicitly realize some level of security as a side effect since the filters control the flow of frames to LAN segments.

The problem with using filters is that their primary focus is on performance improvements, by restricting the flow of certain types of network traffic to specified LAN segments. The filters do not indicate how many times the filter has actually been used and do not indicate a list of the media

access control (MAC) addresses that have been filtered. Therefore, filters do not provide an adequate detection mechanism against break-in attempts.

Another security technique that is commonly employed in hubs is intrusion control. There are token ring and Ethernet managed hubs that allow a network administrator to define, by MAC address, one or more authorized users per hub port. If an unauthorized MAC address is detected at the hub port, then the port is automatically disabled. The problem with this solution is that prevention stops at the hub and no further action is taken once the security intrusion has been detected. This solution does not provide a network-centric, system-wide solution. It only provides a piecemeal solution for a particular type of network hardware namely, the token ring and Ethernet managed hubs. The result is a fragmented solution, where security may exist for some work groups that have managed hubs installed, but not for the entire campus network. At best, the security detection/prevention is localized to the hub level and no solution exists for a network-wide solution.

Other attempts to control LAN access have been done with software program products. For example, IBM Corporation's Lan Network Management (LNM) products LNM for OS2 and LNM for AIX both provide functions called access control to token ring LANs. There are several problems with these solutions. One problem with both of these solutions is that it takes a long time to detect that an unauthorized station has inserted into the ring. An intruder could have ample time to compromise the integrity of a LAN segment before LNM could take an appropriate action. Another problem with the LNM products is that once an unauthorized MAC address has been detected, LNM issues a remove ring station MAC frame. Although this MAC frame removes the station from the ring, it does not prevent the station from reinserting into the ring and potentially causing more damage. Because these products do not provide foolproof solutions, and significant security exposure still exists, they do not provide a viable solution to the problem of network security for campus LAN environments.

Thus, there is a need for a mechanism that ties together all of the piecemeal solutions into a comprehensive system solution that not only provides for detection of security intrusions, but also provides the proactive actions needed to stop the proliferation of security intrusions over the domain of an entire campus network.

SUMMARY OF THE INVENTION

It is, therefore, an object of the invention to provide a system and method for detecting and preventing security intrusions in a computer network.

It is another object of this invention to provide a system and method for detecting and preventing security intrusions in a local area network containing multiple managed devices.

It is a further object of this invention to provide a system and method for detecting and preventing security intrusions in a computer network having a managed hub and at least one interconnect device, such as a router, switch or bridge.

Overall, this invention can be described in terms of the following procedures or phases: discovery, detection, prevention, hub enable, and security clear. During each of these phases, a series of frames are transmitted between the interconnect devices on a campus network. These frames are addressed to a group address (multicast address). This well known group address needs to be defined and reserved for the LAN security functions that are described herein. This group address will be referred to as LAN security feature group address throughout the rest of this description.

The campus LAN security feature relies on managed hubs discovering the interconnect devices in the campus LAN segment that support this LAN security feature. The term "LAN interconnect device" is used throughout this description to refer to LAN switches (token ring and Ethernet 10/100 Mbps), LAN bridges and routers. The managed hub maintains a list of authorized MAC addresses for each port in the managed hub. If the managed hub detects an unauthorized station connecting to the LAN, the hub disables the port and then transmits a security breach detected frame to the LAN security feature group address. Each of the LAN interconnect devices on the campus LAN segment copies the LAN security feature group address and performs the following steps: 1) set up filters to filter the intruding MAC address; 2) forward the LAN security feature group address to other segments attached to the LAN interconnect device; and 3) send an acknowledgement back to the managed hub indicating that the intruding address has been filtered at the LAN interconnect device. Once the managed hub receives acknowledgements from all of the interconnect devices in the campus LAN, the port where the security intrusion was detected is re-enabled for use. Another part of the invention provides a network management station with the capability to override any security filter that was set in the above process.

The following is a brief description of each phase in the preferred embodiment of the invention:

1. Discovery

In this phase, the managed hub determines the interconnect devices in the campus network that are capable of supporting the LAN security feature. The managed hub periodically sends a discovery frame to the LAN security feature group address. The managed hub then uses the responses to build and maintain a table of interconnect devices in the network that support the security feature.

2. Detection

In the detection phase, the managed hub compares the MAC addresses on each port against a list of authorized MAC addresses. If an unauthorized MAC address is detected, then the managed hub disables the port and notifies the other interconnect devices in the campus network by transmitting a security breach detected frame to the LAN security feature group address.

3. Prevention

The prevention phase is initiated when a LAN interconnect device receives the security breach detected frame. Once this frame is received, the LAN interconnect device sets up a filter to prevent frames with the intruding MAC address from flowing through this network device. The LAN interconnect device then forwards the security breach detected frame to the other LAN segments attached to the interconnect device. The LAN interconnect device also transmits a filter set frame back to the managed hub.

4. Hub Enable

The hub enable phase takes place when the managed hub has received all acknowledgements from the LAN interconnect devices in the campus network. When the acknowledgements have been received, the managed hub re-enables the port where the security intrusion occurred.

5. Security Clear Condition

In this phase, a network management station can remove a filter from a LAN interconnect device that was previously set in the prevention step.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be described with respect to a preferred embodiment thereof which is further illustrated and described in the drawings.

FIG. 1 is a block diagram of a campus network in which the present invention can be implemented.

FIG. 2 is a component block diagram for an SNMP managed device.

FIG. 3 is a component block diagram for a network management station.

FIGS. 4A–4C show general frame formats for Ethernet and token ring frames.

FIGS. 5A–5E show the information contained in the Ethernet and token ring frame data fields to represent the different frame types that are implemented in the preferred embodiment.

FIG. 6 illustrates the structure of the Interconnect Device List (ICD).

FIG. 7 illustrates the structure of the Breach List.

FIG. 8 illustrates the structure of the Intrusion List.

FIG. 9 is a flow chart of the processing that occurs in the managed hub to initiate the discovery phase of the invention.

FIG. 10 is a flow chart of the processing that occurs in the interconnect device during the discovery phase of the invention.

FIG. 11 is a flow chart of the processing that occurs in the managed hub during the discovery phase of the invention in response to the receipt of a discovery response frame.

FIG. 12 is a flow chart of the processing that occurs in the managed hub during the detection phase of the invention.

FIG. 13 is a flow chart of the processing that occurs in an interconnect device during the prevention phase of this invention.

FIG. 14 is a flow chart of the processing that occurs in the managed hub during the hub enable phase of the invention.

FIG. 15 is a flow chart of the processing that occurs in the interconnect devices in response to the receipt of a security clear condition frame.

FIG. 16 is an example of the implementation of the invention in a campus LAN environment.

FIG. 17 is an example of the data flows corresponding to the example implementation in a campus LAN environment.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The preferred embodiment of this invention uses the SNMP network management protocol, since SNMP is the most prevalent network management protocol in the industry and is the most widely deployed in campus networks. It should be noted that the concepts in this invention related to network management could also be applied to other network management protocols such as CMIP or SNA.

FIG. 1 illustrates a typical campus network environment in which the present invention can be implemented. As shown in the figure, the campus network 10 contains interconnect devices, such as router 12, router 14, token ring switch 16, bridge 18, managed hubs 20, 22, 24, network management station 26, workstation 28 and file server 30.

The managed hubs and interconnect devices depicted in FIG. 1 are considered SNMP managed devices. The typical component block diagram for an SNMP managed device is illustrated in FIG. 2. A typical managed device is an embedded system that includes a system bus 50, random access memory (RAM) 52, NVRAM 54 to store configuration information, FLASH EPROM 56 to store the operational and boot-up code, a processor or CPU 58 to execute the code instructions, and a media access control (MAC) chip 66 that connects the device to the network **10**. FIG. 2 also shows operational code **60**, TCP/IP protocol stack **62** and SNMP agent code **64**. In most instances, the operational code and the frame processing code execute in FLASH memory **56** or in RAM **52**. The code that implements several phases in this invention is included as a part of the operational code (microcode or firmware) of the managed device. The MAC chip **66** copies the frames corresponding to the different phases into RAM **52** and notifies the processor **58**, usually via an interrupt, that a frame is ready for processing. The operational code **60** handles the interrupt and processes the frame.

FIG. **3** illustrates the typical component block diagram for a network management station such as that indicated by reference numeral **26** in FIG. **1**. The network management station includes a processor **70**, with a system bus **90** to which RAM **72**, direct access storage device (DASD) **74**, other peripherals **76**, display monitor **78**, keyboard **80**, mouse **82** and network interface card **84** are connected.

FIGS. 4A–4C show the general frame formats for Ethernet and token ring frames. The LAN security feature group address is placed in the destination address (DA) field of the discovery request, security breach detected and security clear condition (optionally) frames as discussed more fully below. The data field portion of each frame is used to pass the additional information related to this security feature.

The following describes the information that is included in the data fields of the Ethernet and token ring frame types to represent the different frames that are specific to the preferred embodiment of the invention.

The discovery request frame shown in FIG. 5A is sent to the LAN security feature group address and the data field includes a one byte field which indicates that the frame type (frame type identifierx'01') is a discovery request frame. The time stamp field is the system time value when the discovery request frame is transmitted. It is used to correlate the discovery response frame with the discovery request frame.

The discovery response frame shown in FIG. 5B is sent to the individual MAC address of the managed hub that initiated the request. The data field in this frame includes a one byte field which indicates that the frame type is a discovery response frame (frame type identifierx'02'), and also contains the MAC address of the LAN interconnect device sending the frame, a description of the LAN interconnect device (e.g., IBM 8272 Model 108 Token Ring Switch), and a time stamp that is used to correlate the discovery response frame with the discovery request frame.

The security breach detected frame shown in FIG. 5C is sent to the LAN security feature group address and the data field includes a one byte field which indicates that the frame type is a security breach detected frame (frame type identifierx'03') and contains the MAC address that was detected as the security intruder. Other fields of this frame contain the module number and port number where the security breach was detected and the system time when the security breach was detected. When the time stamp value is used in combination with the intruding MAC address and module and port numbers, it forms an intrusion identifier as will be referred to subsequently. Following the time stamp are device field length indicating the length of the field that follows and address fields. The address field contains the list of addresses that have processed and forwarded the security breach detected frame. It starts with the originating MAC address of the managed hub. Each successive interconnect device that receives the frame, appends its MAC address to

the end of this field and updates the device field length before it forwards the frame. It provides an audit trail or path that the security breach detected frame followed throughout the network. A network management station can monitor the progress of the security breach detected frame through information in the trap frames that it receives.

The filter set frame shown in FIG. 5D is sent to the individual MAC address of the managed hub that initiated the security intrusion condition. The data field includes a one byte field which indicates that the frame type is a filter set frame (frame type identifierx'04') and contains the MAC address of the LAN interconnect device sending the frame. Other fields in this frame are the MAC address of the detected intrusion, the module and port number of the managed hub where the security intrusion was detected, and the time stamp representing the system time when the security breach was detected.

The security clear condition frame shown in FIG. 5E can be sent to the LAN security feature group address or to the individual MAC address of a LAN interconnect device. The data field includes a one byte field which indicates that the frame type is a security clear condition frame (frame type identifierx'05') and contains the intruding MAC address to remove as a filter.

Trap frames are sent to the network management station at various times depending upon the phase of the invention that is being performed. All trap frames have the same basic format with the information in each trap frame varying according to the phase.

In the discovery phase, traps are sent as a result of the managed hub deleting an interconnect device from the list of devices that are in the security domain of interconnect devices. The discovery trap frame contains the trap identifier (x'01'), the MAC address of the interconnect device and device description. This trap indicates that an interconnect device was removed from a managed hub interconnect device list because it did not respond to the managed hub with a discovery response frame within the allotted time period of the discovery window.

Traps sent in the detection phase indicate that the managed hub detected an intrusion on one of the hub ports. Information in this trap frame includes trap identifier (x'02'), the MAC address of the intruding device, the module and port number of the detected intrusion, and the time when the security intrusion was detected.

Traps sent in the prevention phase indicate that the interconnect device has completed the processing of a received security breach detected frame. This trap frame contains the trap identifier (x'03'), the MAC address of the intruding device, the module and port number of the detected intrusion, the time when the security breach was detected and a variable length address field. This last field contains a list of MAC addresses for all the devices that have processed the security breach detected frame. This information provides to the network management station the path that the security breach detected frame followed through the network.

Traps sent in the hub enable phase indicate that the managed hub has reenabled a hub port as a result of receiving filter set frames from all of the interconnect devices in the discovered security domain, i.e., all the discovered interconnect devices. This trap frame contains the trap identifier (x'04'), the MAC address of the intruding device, the module and port number of the detected intrusion, and the time when the security breach was detected.

For token ring networks, the information in the trap frames can be included in frames addressed to the functional address of the LAN manager. The LAN management frame format and defined functional address are specified in the IBM Token Ring Network Architecture (SC30-3374-02) publication.

For managed hubs, the authorized address list (AAL) controls which MAC addresses are allowed to connect to specified ports. Each entry in the AAL consists of two fields: port number and authorized address. The port number identifies a specific port on the hub; the authorized address field specifies the address or addresses that are allowed to connect to the port.

The AAL can be built by the network administrator as part of the configuration of the managed hub. The network administrator identifies the addresses that are allowed to connect to specific ports on the hub. After the initial configuration, the AAL can be updated in several ways. The network management station can add or delete entries in the AAL by sending SNMP management frames. Since most managed hubs provide a Telnet interface into the device to change configuration parameters, a Telnet session could be used to add or delete entries in the AAL. Also, since most managed hubs provide for the attachment of a local console over an RS232 serial port connection which can be used to change configuration parameters, a local console session can be used to add or delete entries in the AAL.

Alternatively, the AAL can be built dynamically through a learning process. Most managed hubs provide a mechanism in the hardware to capture the addresses of the stations that are attached to the ports of a hub. These learned addresses can be provided to the network management station as those stations authorized to access the hub. These learned addresses are then used as the AAL for the managed hub.

The discovery phase is initiated by each managed hub in the campus network. Its purpose is to determine the LAN interconnect devices in the campus LAN that support the LAN security feature. Each managed hub periodically transmits a discovery frame (FIG. 5A) to the LAN security feature group address. The managed hub then uses the information in the response frame (FIG. 5B) to build and maintain a list of all of the devices that support the LAN security feature. This list is referred to as the Interconnect Device List (ICD). The addresses in this list are used in the hub enable phase to correlate the reception of the filter set frame (FIG. 5D) with entries in the list. The managed hubs typically store these ICD lists in management information base (MIB) tables where they can be retrieved, upon request, from a network management station.

The discovery phase can also be used to provide an integrity check on the ICD list of devices supporting the LAN security feature. By periodically transmitting the discovery frame (FIG. 5A) to the LAN security feature group address, checks can then be made to ensure that all of the devices are still in the ICD security list. If any discrepancies are detected, e.g., if a station is removed from the list or added to the list, then an SNMP trap is sent to the network management station. This notification alerts the network administrator that a potential security exposure exists in the campus network. FIG. 6 illustrates the structure of the ICD list along with the information stored in the list for each discovered interconnect device. Other lists that are built and maintained in the detection and prevention phases are the Breach List shown in FIG. 7 and the Intrusion List shown in FIG. 8. Their use will be explained below in the description of the detection and prevention phases.

The detection phase operates at the managed hub level. Each port on the managed hub can be configured to hold one or more MAC addresses of users that are authorized to access the network. The managed hubs can be 10 or 100 Mbps Ethernet or token ring hubs. Current hub chipsets provide the capability to determine the last source MAC address that is seen on a port. When a station attempts to connect to a network, either by inserting into the token ring or by establishing a link state with an Ethernet hub, the last source address seen on the port is compared to the authorized list of MAC addresses that has been defined for this port. If the address is authorized then normal network operations occur. If the address is not authorized, then the managed hub performs the following actions:

1. disables the port;
2. sends an SNMP trap frame to the network management station;
3. sends an alert frame to the functional address of the LAN Manager (token ring); and
4. transmits a security breach detected frame (FIG. 5C) to the LAN security feature group address.

Additional variables in the SNMP trap provide information about the point of intrusion: e.g. the module id (in the case of stackable hubs), the port number, the network number (in cases where hubs have multiple backplanes), and a time stamp (sysUpTime) of when the intrusion was detected. SysUpTime is an SNMP MIB variable that represents the time (units of 0.01s) since the network management portion of the system was last reinitialized.

Some managed hubs support multiple backplanes or networks. In this case, the security breach detected frame is transmitted on all of the active backplanes/networks within the hub.

The well known group address needs to be defined and reserved for LAN security functions. The security breach detected frame (FIG. 5C) containing the MAC address of the station that intruded into the network is sent to the LAN security feature group address.

The prevention phase spans the network. Each interconnect device in the campus network is configured to copy frames addressed to the LAN security feature group address. Upon a security intrusion, the network interconnect devices copy the security breach detected frame (FIG. 5C) and perform the following functions:

1. set filters based on the intruder's MAC address.
2. transmit a security breach detected frame (FIG. 5C) to the LAN security feature group address.
3. send an SNMP trap frame to the network management station.
4. send an alert frame to the functional address of the LAN manager (token ring).
5. transmit filter set frame (FIG. 5D) to the MAC address of the hub that initiated the security breach process.

Setting filters by the network interconnect device prevents intrusion attempts with this MAC address originating elsewhere in the campus network from flowing through this interconnect device. This protects an enterprise's data on this segment of the network from any attacks via the intruder's MAC address.

The interconnect device extracts the intrusion identifier information from the security breach detected frame. If this is the first time the interconnect device has received a security breach detected frame with this intrusion identifier, the interconnect device adds this information to the Intrusion List, then checks to ensure the filter has been set for the intruding MAC address and resets, if required. The interconnect device then transmits the security breach detected frame on all ports except the port on which the security breach detected frame was received.

Sending the trap frame indicates that the filter has been set as a result of receiving the security breach detected frame. Likewise, sending the alert frame indicates that the filter has been set as a result of receiving the security breach detected frame.

The hub enable phase operates at the network level. The hub that initiates the security breach process receives the filter set frames from the interconnect devices in the campus network. The hub then waits to receive responses back from all of the interconnect devices that were determined in the discovery phase to be in the campus network. When all the interconnect devices in the network have responded to the hub with the filter set frame, the hub then re-enables the port for use and then sends a TRAP frame back to the network management station indicating that all filters have been set for the intruding MAC address. The network management station can optionally forward this information to a network management application such as IBM Corporation's NetView/390 product via an alert.

The security clear condition phase of this invention provides the capability for a network administrator to manually override, if necessary, one of the filters that has been set in the prevention phase. The network management station could globally clear, i.e., remove a filter from all LAN interconnect devices by transmitting the security clear condition frame (FIG. 5E) to the LAN security feature group address. The network management station could selectively clear, i.e., remove a filter from a LAN interconnect device by transmitting the security clear condition frame to the MAC address of the specific LAN interconnect device.

FIGS. 9–15 are flow charts that illustrate the processing that occurs in the managed hub and in the interconnect devices during each phase of the invention. The code to implement the discovery phase of this invention runs within the managed hub and interconnect device as event driven threads within the realtime OS embedded system. The flows in FIG. 9 depict the processing that occurs in the managed hub to initiate each discovery phase. This task manages the initialization and update of the Interconnect Device List and timing of the next iteration of the discovery phase. The following briefly describes each logic block in the figure.

Step **100**: Entry to this task can be caused by a power on and/or reset. This would be one of many tasks that would run in response to this event.

Step **101**: There are two lists, a period, a window, and two flags that are used by the managed hub in this invention. The ICD (Interconnect Device) List contains information on the devices found during the discovery phase. The Breach List contains information on intrusions recognized by the hub and in the process of being secured. The period is the time between discovery phases. The window is the time between when a discovery phase is initiated and when an Interconnect Device must respond before being assumed inaccessible due to network or device outage. One flag is an indication that initialization has completed. The other flag is an indication that the security feature is enabled. The lists, the period, the window and the enabled flag may be cleared or loaded from persistent memory. The initialized flag is set to True.

Step **102**: Test for whether the security feature is enabled.

Step **103**: Each managed hub maintains a MIB variable that is called SysUpTime. This is used as a time stamp for security feature frames.

Step **104**: The discovery frame is built with the data field containing the type of the frame—Request.

Step **105**: The frame is sent to the LAN security feature group address.

Step **106**: The discovery phase is initiated periodically as an integrity check on the security feature coverage within the network. The period is adjustable to reflect variable path lengths or round-trip-times between a managed hub and interconnect devices. The period can be set via SNMP. The longer the period, the less the integrity of the network coverage. The shorter the period, the higher the traffic rate required for the security feature.

Step **107**: Set a pointer to the head of the list of ICD (Interconnect Device) List items. The pointer may point to an item or nothing if there are not items in the list. (The ICD List is a list of the interconnect devices that responded in a previous discovery phase). This part of the task is to update the Interconnect Device List by updating items as appropriate or deleting them as necessary.

Step **108**: Does the pointer point to an item in the list or does it point beyond the end of the list?

Step **109**: Each ICD List item has a time stamp from the last discovery response frame received from the device.

Step **110**: Is the time for the item in the ICD List later than current time?

Step **111**: If yes, the managed hub has reset or rolled over its SysUpTime since the last response from the ICD. Set the time in the ICD List item to current time.

Step **112**: Is the difference between the current time and the last response time from the item greater than the discovery window?

Step **113**: Assume the device is inaccessible due to network or device outage and purge the item from the ICD List. Also, decrement the outstanding filter set count on all the Breach List items.

Step **114**: If there is a network management station (NMS) that is receiving traps from the managed hub and the traps are enabled, send a trap indicating that the interconnect device is no longer accessible. If there is an LNM for OS/2 station available and traps are enabled, send a trap to the LNM for OS/2 station.

Step **115**: Move the ICD List pointer to the next item or to the end of the list if no more entries exist. This is for stepping through the entire list of ICD items.

Step **116**: End the task and return to the embedded system OS.

Step **117**: Enter this task due to a timer driven interrupt (set in step **106**).

The flows in FIG. **10** depict the processing that occurs in the interconnect devices during each iteration of the discovery phase. This task responds to the receipt of a discovery request frame by sending a discovery response frame. The following briefly describes each logic block in the figure.

Step **143**: The task is initiated by the receipt of a discovery request frame.

Step **144**: A check is made for whether the security feature is enabled. This determines if any additional processing is required.

Step **145**: The source MAC address and time stamp are extracted for building the response.

Step **146**: The discovery response frame is built using the information from the discovery request frame that was just received.

Step 147: The frame is sent to the originating managed hub.

Step 148: The task ends, returning control to the embedded OS.

The flows in FIG. 11 depict the processing that occurs in the managed hub in response to the receipt of a discovery response frame. This task maintains the state of this iteration of the discovery phase. The following briefly describes each logic block in the figure.

Step 130: The task is initiated in the managed hub by the receipt of a discovery response frame.

Step 131: The interconnect device information is extracted from the frame.

Step 132: The Interconnect Device List is searched for an item with a MAC address matching the source address of the discovery response frame.

Step 133: Has a match been found?

Step 134: If a match is found, update the last response time in the ICD List item with the time stamp that was extracted from the discovery response frame.

Step 135: If there is no match, assume that the device is not in the list because of either network/device outages or the device has just started utilizing the security feature. It is necessary to determine if the discovery window is still large enough. The round-trip-time is calculated, and multiplied by 2 to derive a potential discovery window. If this is larger than the current discovery window, the discovery window needs to be changed.

Step 136: Change the discovery window.

Step 137: Create a new Interconnect Device List item using the source address from the discovery response frame, the device description from the frame, and the time stamp from the frame. Add it to the list.

Step 138: Optionally send a trap to the network management station(s) and if this is a token ring, to the LAN manager functional address.

Step 139: The task ends, returning control to the embedded OS.

The code to implement the detection phase of this invention runs as a separate task independent from the other tasks in the managed hub. The flows in FIG. 12 depict the processing that occurs during the dispatch of the detection phase task. This task simply checks all the ports in the hub to ensure that the station attached to the port has been authorized to establish a connection on this port. The AAL (Authorized Address List) defines which MAC addresses are allowed to connect to specific ports on the hub. The following briefly describes each logic block in the figure.

Step 200: This is the entry point for the detection phase task. Processing starts at port number 1 in the hub and continues until all of the ports in the hub have been processed.

Step 210: This step checks if a station is attached to the port in the hub. If a station is attached, then an address exists for the port. If an address is detected for the port (i.e., a station is attached to the port), then processing continues with step 220. If there is no address detected for this port (i.e., no station is attached), then processing continues with step 230.

Step 220: A check is made here to ensure that the address that has been detected on this port is in the list of authorized addresses. If the address detected on the port is authorized, then continue processing at step 230. If the address detected on the port is not in the authorized list, then processing continues at step 250.

Step 230: A check is made here to see if all of the ports in the hub have been processed. If all of the ports have been processed, then processing resumes at step 200 with the processing of port number 1. If this was not the last port and there are more ports to process, then processing continues at step 240.

Step 240: In this step, the next port in the hub is set up to be processed. Processing then continues at step 210.

Step 250: In this step a check is made to see if the port is already disabled. If the port is already disabled, then the port/network is already secure from intruders on this port. If the port is already disabled, then processing continues at step 230. If the port is enabled, processing then continues at step 260.

Step 260: In this step, the port is disabled. Processing then continues at step 265.

Step 265: In this step, an entry is added to the Breach List containing the following: MAC address that was detected as the intruder, the module and port number where the intrusion was detected, the time (sysUpTime) when the security breach was detected, and the outstanding filter set count which is set to the number of entries in the ICD list. Processing then continues at step 270.

Step 270: In this step, the security breach detected frame is transmitted on all network segments of the hub. The info field of the security breach detected frame includes the following: MAC Address of the intruder, module number, port number, time stamp (sysUpTime), the device field length initialized to 6 (bytes), the 6 byte MAC address of the managed hub. Processing then continues at step 280.

Step 280: In this step, a trap frame is optionally sent to the network management station. The trap frame includes the following information:

(a) trap identifierx'02';

This indicates that the managed hub detected in intrusion on one of the hub ports.

(b) MAC address of the intruding device;

(c) module number of the detected intrusion;

(d) port number of the detected intrusion;

(e) time when the security breach was detected;

Processing then continues at step 290.

Step 290: In this step, a check is made to see if this invention has been implemented in a token ring network. The token ring architecture defines a special functional address that is used by LAN management stations. Functional addresses are only used in token ring environments. If the invention is implemented in a token ring network, processing then continues at step 295. If the invention is implemented in a non-token ring network, processing then continues at step 230.

Step 295: In this step, a frame is sent to the functional address of the LAN manager with the information from step 280. Processing then continues at step 230.

FIG. 13 depicts the flows for the prevention phase of the invention. The prevention phase is implemented in the interconnect devices of the network. The following briefly describe each logic block in the figure.

Step 300: The processing is initiated when the interconnect device receives a frame from the network. The interconnect device copies the frame and saves the port number that the frame was received on. Processing then continues at step 302.

Step 302: In this step, the frame that was copied in step 300 is interrogated and a check is made to determine if the destination address of the frame is equal to the LAN security feature group address. If the received frame is addressed to the LAN security feature group address, then processing continues at step **306**. Otherwise, the frame is of some other type and the processing continues with step **304**.

Step **304**: This step is encountered for all frame types other than the LAN security feature. The normal frame processing code of the interconnect device runs here.

Step **306**: In this step, the intrusion identifier information is copied from the frame. The intrusion identifier consists of the following information:

(a) MAC address of the intruder;

(b) module number;

(c) port number;

(d) time stamp;

Processing then continues at step **308**.

Step **308**: In this step, a check is made to determine if the intrusion identifier is already in the Intrusion List of this interconnect device. If yes, processing then continues at step **316**. If no, processing then continues at step **312**.

Step **312**: In this step, the intrusion identifier information is added to the Intrusion List. Processing then continues at step **316**.

Step **316**: In this step, the current port of the interconnect device is set to port number 1. Processing then continues at step **318**.

Step **318**: In this step, a check is made to determine if the intruding MAC address is already filtered on the current port. If yes, processing then continues at step **322**. If no, processing then continues at step **320**.

Step **320**: In this step, a filter is set for the intruding MAC address on the current port. Processing then continues at step **322**.

Step **322**: In this step a check is made to determine if the filter processing has been applied to all of the ports in the interconnect device. If all of the ports have been processed, processing then continues at step **326**. If there are more ports to process, processing then continues at step **324**.

Step **324**: In this step, the current port is set to the next port in the interconnect device. Processing then continues at step **318**.

Step **326**: In this step, the security breach detected frame is propagated throughout the network. The interconnect device transmits the security breach detected frame on all ports other than the port the original frame was received on. (Reference step **300** where it is determined which port the frame was received on). Before transmitting the security breach detected frame, the ICD appends its MAC address to the addresses field of the frame and increments the device field length field of the frame by 6. This provides the audit trail or the path information for the security breach detected frame. Processing then continues at step **332**.

Step **332**: In this step, the interconnect device transmits the filter set frame to the originator of the security breach detected frame. The originator is determined by extracting the source address from the frame that was copied in step **306**. Processing then continues at step **334**.

Step **334**: In this step, a trap frame is sent to the network management station. The trap frame includes the following information:

(a) trap identifierx'03';

This indicates that the interconnect device has completed the processing of a received security breach detected frame.

(b) MAC address of the intruding device;

(c) module number of the detected intrusion;

(d) port number of the detected intrusion;

(e) time when the security breach was detected;

(f) addresses field;

This is a variable length field that contains a list of all of the devices that have processed the security breach detected frame. This information provides to the network management station the path that the security breach detected frame followed throughout the network.

Processing then continues at step **336**.

Step **336**: In this step, a check is made to see if this invention has been implemented in a token ring network. The token ring architecture defines a special functional address that is used for LAN management stations. Functional addresses are only used in token ring environments. If the invention is implemented in a token ring network, processing then continues at step **338**. If the invention is implemented in a non-token ring network, processing then continues at step **340**.

Step **338**: In this step, a frame containing the same information in the trap frame in step **334** is sent to the functional address of the LAN manager. Processing then continues at step **340**.

Step **340**: In this step, processing resumes again at step **300**.

The code to implement the hub enable phase of this invention runs within the managed hub as event driven threads within the realtime OS embedded system. The flows in FIG. 14 depict the processing that occurs in the managed hub in response to receipt of each filter set frame. The task maintains the necessary lists of interconnect devices and breaches to complete the hub enable phase for each breach. The following briefly describes each logic block in the figure.

Step **400**: The task is initiated in the managed hub by the receipt of a filter set frame.

Step **401**: Get the source address of the frame for finding the associated ICD List item.

Step **402**: The Interconnect Device List is scanned for an item with the same MAC address as the source address of the frame.

Step **403**: Was a match found? If not, assume that the interconnect device is no longer accessible.

Step **404**: If a match is found, decrement the outstanding breach response count in ICD List item by 1. This provides an up-to-date count of outstanding responses for each ICD.

Step **405**: Extract intrusion identifier information from the frame.

Step **406**: Scan the Breach List for an item with a matching intrusion identifier.

Step **407**: Match found?

Step **408**: If a match is found, decrement the outstanding filter set count by 1 in the matching Breach List item.

Step **409**: Have all interconnect devices responded? Are all filters set?

Step **410**: Since the intruder is now being filtered and has been removed from the network, remove the Breach List item.

Step **411**: If there is a listening network management station(s), send a trap. If this is a token ring, send an alert to the LAN manager functional address.

Step 412: Optionally reenable the port. This is a policy decision. It may also reflect the likelihood of the intruder still attempting to intrude via this same port.

Step 413: End the task and return control to the embedded OS.

The code to implement the security clear condition phase of this invention runs within the interconnect devices as event driven threads within the realtime OS embedded system. The flows in FIG. 15 define the processing that occurs in the interconnect devices in response to receipt of each security clear condition frame. The task updates the Intruder List of breaches and completes the security clear condition phase for each breach. The following briefly describes each logic block in the figure.

Step 500: The task is initiated in the interconnect device by the receipt of a security clear condition frame from a network management station.

Step 501: Extract the intruder MAC address from the security clear condition frame.

Step 502: Search the Intrusion List for a matching MAC address.

Step 503: Is there a match?

Step 504: If there is a match, remove the item from the Intrusion List.

Step 505: Remove filter for the intruding MAC address.

Step 506: End the task and return control to the embedded OS.

Two examples are given below to illustrate the actions that are performed by the managed hub and interconnect devices in an implementation of this invention in an operational campus environment. Referring again to FIG. 1, there is depicted a workstation 28, attached to an Ethernet hub 24, that is attempting to gain unauthorized access to a file server 30 that is located on a token ring segment. The security intrusion is detected by the managed Ethernet hub 24, since the MAC address of the workstation 28 is not authorized for this port in the hub. The managed hub 24 then disables the port and transmits the security breach detected frame to the LAN interconnect device 14 on this segment, which, in turn, forwards the security breach detected frame to LAN interconnect devices 12, 16 that are attached to subnet 3 and subnet 4, respectively. LAN interconnect device 12, in turn, forwards the security breach detected frame to LAN interconnect device 18. The LAN interconnect devices 12, 14, 16, 18 set filters on all ports in the device to prevent frames with the intruding MAC address from flowing through the interconnect device.

More specifically, the managed hub 24 disables the port and transmits the security breach detected frame to router 14. The managed hub 24 also sends a trap frame to the management station 26. Router 14 applies the intruder's MAC address as a filter on all of its ports and forwards the security breach detected frame on all of its ports, except the port the security breach detected frame was received on. Router 14 then sends a trap to the network management station 26 and sends a filter set frame back to the managed hub 24. Router 12 and the token ring switch 16 also receive the security breach detected frame and perform the same processing operations as defined above for router 14. The bridge 18 receives the security breach detected frame and performs the same processing operations as done by router 14. The managed hub 24 now correlates all of the received filter set frames with the interconnect devices 12, 14, 16, 18 that were discovered via the discovery request/response frames and reenables the port. The managed hub 24 then sends a trap to the management station 26 to indicate that the intruder's port has been reenabled.

As a practical example of the implementation of this invention in a campus LAN environment, FIG. 16 depicts a university setting in which there is a managed hub on each floor of the buildings in a campus network. The network infrastructure consists of a pair of Ethernet switches attached to a campus backbone. Each Ethernet switch is also attached to a plurality of Ethernet managed hubs (one on each floor in each building). The figure shows a student dormitory that is attached to the same network that runs the university administration applications. There are obvious security concerns about students accessing the proprietary administrative information (i.e., grades, transcripts, payroll, accounts receivable/payable, etc.).

An intruder trying to access the network via one of the managed hub ports in the dormitory is stopped at the port of entry to the network and further access to the campus network is prevented by having the intruder's MAC address filtered on all LAN interconnect devices. The symbols containing a "B" in FIG. 16 indicate the points in the campus network where frames with the intruding MAC address are blocked from access to LAN segments by the setting of filters. The data flows corresponding to the example are shown in FIG. 17 and are self-explanatory.

For simplicity, this invention has used the term managed hub to refer to traditional token ring and Ethernet port concentration devices (e.g., IBM 8238, IBM 8224, IBM 8225, IBM 8250, IBM 8260). In reality, the functions of the managed hub can be extended to LAN switches (both token ring and Ethernet) where dedicated stations could be attached directly to the switch port. LAN switches would have to add the functionality of authorizing a set of MAC addresses that could attach to a switch port and detecting any unauthorized accesses to the switch port.

To describe the key aspects of this LAN security invention, it was easiest to illustrate with an implementation using managed hubs. In reality, many large enterprises use a combination of both managed hubs and unmanaged hubs throughout their networks. This invention is readily extendible and the security detection mechanism can easily be integrated into the function of a LAN bridge. The bridge would keep the list of authorized addresses for a given LAN segment where access to the LAN is via low cost unmanaged concentrators. The bridge would then detect any new addresses on the LAN segment and compare the addresses against the authorized list. If an unauthorized address was detected, the bridge would then set up filters for the intruding MAC address, and transmit the security breach detected frame to the other interconnect devices attached to the campus network. In this case, the intruder would be isolated to the LAN segment where the intrusion was first detected. This example shows that the composite function of the managed hub could be integrated into a LAN bridge and the bridge could control the security access for a large segment consisting of unmanaged concentrators.

Another special use of this invention involves the tasks of a network administrator. A key day-to-day task for most network administrators falls into the category of moves, adds, and changes to network configuration. In this invention, the network management station has complete awareness of all of the authorized users throughout the campus network. In the event that a security breach is detected, in the special case where an authorized user is trying to gain access through an unauthorized port, the network management station could detect this situation and automatically take the appropriate actions (i.e., remove filters from the interconnect devices since this is an authorized user). This type of action would assist administrators that work in dynamic environments where there are frequent moves, adds and changes.

The preferred embodiment of the invention has relied upon the detection of unauthorized MAC addresses by the managed hub. It can easily be modified to apply to the network layer (layer 3) or higher layers, in the Open System Interconnection (OSI) protocol stack and work with such well known network protocols as TCP/IP, IPX, HTTP, AppleTalk, DECnet and NETBIOS among others.

Currently, many LAN switches have custom application specific integrated circuits (ASICs) that are designed to detect or recognize frame patterns in hardware. These LAN switches use this frame type recognition capability primarily for frame forwarding based on the IP address and for placing switch ports in a virtual LAN (VLAN). In order to provide security protection at the network layer, it will be clear to one skilled in the art that the authorized address list (AAL) described herein can be extended to include IP addresses. The so-modified AAL, coupled with the LAN switch capability to detect IP addresses in a frame will enable implementation of the detection and prevention phases to support IP addresses. In the detection phase, the ASIC-based LAN switch can be used to obtain the IP address that is connected to a port. The detected IP address would then be compared to the authorized IP addresses in the AAL. If an unauthorized IP address is detected, the invention works as previously described with the disabling of the port and the transmission of the security breach detected frame. In the prevention phase, the interconnect devices are notified of intruding IP addresses and then apply filters for the intruding IP address.

The present invention can also be modified to operate at the application layer (layer 7) of the OSI protocol stack. Currently, several commercially available LAN switches, such as the model 8273 and model 8274 LAN switches available from IBM Corporation, provide a capability for a user-defined policy for creating a VLAN. This user-defined policy enables one to specify an offset into a frame and a value (pattern) to be used to identify the frame. Once the user-defined policy has been defined, the switch ASIC detects all frames matching the specified pattern and places them into a specific VLAN. Since the custom ASIC recognizes the user-defined pattern, it can be programmed to recognize portions of a frame that identify a specific application. This application pattern can then be used as the detection criteria in the invention and thus provide application layer security.

The present invention can be modified further to provide additional security by encryption of the data fields in the frames that are used to implement the inventive concepts described above. One of the most widely known and recognized encryption algorithms is the Data Encryption Standard (DES). The implementation of DES or other encryption algorithm to encrypt the data fields of frames described in this invention can ensure the privacy and integrity of the communication between managed hubs, interconnect devices and network management stations. Security protocols such as Secure Sockets Layer (SSL) utilizing public key encryption techniques are becoming standardized and can be used to further enhance the invention described herein.

While the invention has been particularly shown and described with reference to the particular embodiments thereof, it will be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention.

Having thus described our invention, what we claim and desire to secure as Letters Patent is as follows:

1. A method for providing security against intrusion in a computer network having a plurality of managed devices, said method comprising the steps of:

discovering by a first managed device each of said plurality of managed devices that are enabled to provide network security;

detecting an unauthorized address on a first port of said first managed device and disabling said first port;

setting a filter at each of said plurality of managed devices to prevent frames having the unauthorized address from being forwarded through said computer network; and

reenabling said first port after said filtering step has been completed.

2. The method for providing security against intrusion of claim 1 further comprising the step of removing of said filter that had been set at each of said plurality of managed devices.

3. The method for providing security against intrusion of claim 1 wherein said first managed device is a managed hub.

4. The method for providing security against intrusion of claim 1 wherein said first managed device is a switch.

5. The method for providing security against intrusion of claim 1 wherein said plurality of managed devices includes a token ring switch.

6. The method for providing security against intrusion of claim 1 wherein said plurality of managed devices includes an Ethernet switch.

7. The method for providing security against intrusion of claim 1 wherein said plurality of managed devices includes a bridge.

8. The method for providing security against intrusion of claim 1 wherein said plurality of managed devices includes a router.

9. The method for providing security against intrusion of claim 1 wherein said computer network includes a local area network.

10. The method for providing security against intrusion of claim 1 further comprising the steps of building and maintaining an authorized address list at said first managed device of addresses that are allowed to connect to each port in said first managed device.

11. The method for providing security against intrusion of claim 10 wherein each entry in said authorized address list includes a port number and an authorized address.

12. The method for providing security against intrusion of claim 1 wherein said discovering step includes the steps of:

transmitting a discovery request frame by said first managed device, said discovery request frame having a security feature group address;

receiving said discovery request frame at each of said plurality of managed devices and transmitting a discovery response frame back to said first managed device;

building and maintaining an interconnect device list at said first managed device of said plurality of managed devices that transmitted said discovery response frame back to said first managed device.

13. The method for providing security against intrusion of claim 12 wherein each entry in said interconnect device list includes an address of the managed device that sent the discovery response frame and a time stamp extracted from said discovery response frame.

14. The method for providing security against intrusion of claim 11 wherein said detecting step includes the steps of:

comparing, for each port, a source address of a station attempting to connect to said port with the authorized address list of addresses for said port and determining whether said source address is on said authorized address list.

15. The method for providing security against intrusion of claim 12 wherein following said disabling step said method further includes:

sending a trap frame by said first managed device to a network management station indicating that an intrusion has been detected on said first port; and

transmitting a security breach detected frame by said first managed device and having said security feature group address to said plurality of managed devices that have entries in said interconnect device list.

16. The method for providing security against intrusion of claim 15 wherein said security breach detected frame includes a source address of an unauthorized station, the port number of said first managed device at which the intrusion occurred, and a time stamp representing the time at which the unauthorized station was detected.

17. The method for providing security against intrusion of claim 16 wherein following the receiving of said security breach detected frame and setting of filters, each of said plurality of managed devices performs the additional steps of:

transmitting said security breach detected frame on all ports except the port on which said each managed device received said security breach detected frame;

sending a trap frame to the network management station indicating that said filter has been set as a result of receiving said security breach detected frame; and

transmitting a filter set frame to said first managed device.

18. The method for providing security against intrusion of claim 17 wherein said filter set frame includes the address of said each managed device sending said filter set frame, the source address of said unauthorized station, the port number of said first managed device at which the intrusion occurred, and a time stamp representing the time at which the unauthorized station was detected.

19. The method for providing security against intrusion of claim 1 wherein following said reenabling step said first managed device sends a trap frame to a network management station indicating that said filtering step has been completed.

20. The method for providing security against intrusion of claim 2 wherein said removing step includes transmitting a security clear condition frame to said plurality of managed devices.

21. The method for providing security against intrusion of claim 2 wherein said removing step includes transmitting a security clear condition frame to a selected managed device of said plurality of managed devices.

22. The method for providing security against intrusion of claim 20 or 21 wherein said security clear condition frame includes said unauthorized address.

23. A system for providing security against intrusion in a computer network having a plurality of managed devices, said system comprising:

means for discovering at a first managed device each of said plurality of managed devices that are enabled to provide network security;

means for detecting an unauthorized address on a first port of said first managed device and means for disabling said first port;

means for setting a filter at each of said plurality of managed devices to prevent frames having the unauthorized address from being forwarded through said computer network; and

means for reenabling said first port of said first managed device after said filtering step has been completed.

24. The system for providing security against intrusion of claim 23 further comprising means at a network management station for generating a security clear condition frame to initiate the removing of said filter that had been set at each of said plurality of managed devices.

25. The system for providing security against intrusion of claim 23 wherein said first managed device is a managed hub.

26. The system for providing security against intrusion of claim 23 wherein said first managed device is a switch.

27. The system for providing security against intrusion of claim 23 wherein said plurality of managed devices includes a token ring switch.

28. The system for providing security against intrusion of claim 23 wherein said plurality of managed devices includes an Ethernet switch.

29. The system for providing security against intrusion of claim 23 wherein said plurality of managed devices includes a bridge.

30. The system for providing security against intrusion of claim 23 wherein said plurality of managed devices includes a router.

31. The system for providing security against intrusion of claim 23 wherein said computer network includes a local area network.

32. The system for providing security against intrusion of claim 23 further comprising means for building and maintaining an authorized address list at said first managed device of addresses that are allowed to connect to each port in said first managed device.

33. The system for providing security against intrusion of claim 32 wherein each entry in said authorized address list includes a port number and an authorized address.

34. The system for providing security against intrusion of claim 23 wherein said means for discovering includes:

means for transmitting a discovery request frame by said first managed device, said discovery request frame having a security feature group address;

means for receiving said discovery request frame at each of said plurality of managed devices and means for transmitting a discovery response frame back to said first managed device;

means for building and maintaining an interconnect device list at said first managed device of said plurality of managed devices that transmitted said discovery response frame back to said first managed device.

35. The system for providing security against intrusion of claim 34 wherein each entry in said interconnect device list includes an address of the managed device that sent the discovery response frame and a time stamp extracted from said discovery response frame.

36. The system for providing security against intrusion of claim 33 wherein said means for detecting includes:

means for comparing, for each port, a source address of a station attempting to connect to said port with the authorized address list of addresses for said port and means for determining whether said source address is on said authorized address list.

37. The system for providing security against intrusion of claim 34 further including:

means for sending a trap frame by said first managed device to a network management station indicating that an intrusion has been detected on said first port; and

means for transmitting a security breach detected frame by said first managed device and having said security feature group address to said plurality of managed devices that have entries in said interconnect device list.

38. The system for providing security against intrusion of claim 37 wherein said security breach detected frame includes a source address of an unauthorized station, the port number of said first managed device at which the intrusion occurred, and a time stamp representing the time at which the unauthorized station was detected.

39. The system for providing security against intrusion of claim 38 wherein each of said plurality of managed devices further comprises:

means for transmitting said security breach detected frame on all ports except the port on which said each managed device received said security breach detected frame;

means for sending a trap frame to the network management station indicating that said filter has been set as a result of receiving said security breach detected frame; and

means for transmitting a filter set frame to said first managed device.

40. The system for providing security against intrusion of claim 39 wherein said filter set frame includes the address of said each managed device sending said filter set frame, the source address of said unauthorized station, the port number of said first managed device at which the intrusion occurred, and a time stamp representing the time at which the unauthorized station was detected.

41. The system for providing security against intrusion of claim 23 wherein said first managed device further comprises means for sending a trap frame to a network management station indicating that said filter has been set at each of said plurality of managed devices.

42. The system for providing security against intrusion of claim 24 wherein said security clear condition frame includes said unauthorized address.

43. A method for providing security against intrusion in a computer network having a managed hub and at least one interconnect device, said method comprising the steps of:

building and maintaining an authorized address list at said managed hub of addresses that are allowed to connect to each port in said managed hub;

discovering by said managed hub each interconnect device that is enabled to provide network security;

detecting an unauthorized address on a first port of said managed hub and disabling said first port;

setting a filter at each interconnect device to prevent frames having the unauthorized address from being forwarded through said computer network; and

reenabling said first port after said filtering step has been completed.

44. The method for providing security against intrusion of claim 43 further comprising the step of removing of said filter that had been set at each interconnect device.

45. The method for providing security against intrusion of claim 43 wherein said at least one interconnect device includes a token ring switch, an Ethernet switch, a bridge or a router.

46. The method for providing security against intrusion of claim 43 wherein said discovering step includes the steps of:

transmitting a discovery request frame by said managed hub, said discovery request frame having a security feature group address;

receiving said discovery request frame at each interconnect device and transmitting a discovery response frame back to said managed hub;

building and maintaining an interconnect device list at said managed hub of each interconnect device that transmitted said discovery response frame back to said managed hub.

47. The method for providing security against intrusion of claim 46 wherein said detecting step includes the steps of:

comparing, for each port, a source address of a station attempting to connect to said port with an authorized address list of addresses for said port and determining whether said source address is on said authorized address list.

48. The method for providing security against intrusion of claim 46 wherein following said disabling step said method further includes:

sending a trap frame by said managed hub to a network management station indicating that an intrusion has been detected on said first port; and

transmitting a security breach detected frame by said managed hub and having said security feature group address to each interconnect device that has an entry in said interconnect device list.

49. The method for providing security against intrusion of claim 48 wherein following the receiving of said security breach detected frame and setting of filters, each interconnect device performs the additional steps of:

transmitting said security breach detected frame on all ports except the port on which said each interconnect device received said security breach detected frame;

sending a trap frame to the network management station indicating that said filter has been set as a result of receiving said security breach detected frame; and

transmitting a filter set frame to said managed hub.

50. The method for providing security against intrusion of claim 43 wherein following said reenabling step said managed hub sends a trap frame to a network management station indicating that said filtering step has been completed.

51. The method for providing security against intrusion of claim 44 wherein said removing step includes transmitting a security clear condition frame to each interconnect device.

52. A system for providing security against intrusion in a computer network having a managed hub and at least one interconnect device, said system comprising:

means for building and maintaining an authorized address list at said managed hub of addresses that are allowed to connect to each port in said managed hub;

means for discovering by said managed hub each interconnect device that is enabled to provide network security;

means for detecting an unauthorized address on a first port of said managed hub and means for disabling said first port;

means for setting a filter at each interconnect device to prevent frames having the unauthorized address from being forwarded through said computer network; and

means for reenabling said first port of said managed hub after said filtering step has been completed.

53. The system for providing security against intrusion of claim 52 further comprising means at a network management station for generating a security clear condition frame to initiate the removing of said filter that had been set at each interconnect device.

54. The system for providing security against intrusion of claim 52 wherein said at least one interconnect device includes a token ring switch, an Ethernet switch, a bridge or a router.

55. The system for providing security against intrusion of claim 52 wherein said means for discovering includes:

means for transmitting a discovery request frame by said managed hub, said discovery request frame having a security feature group address;

means for receiving said discovery request frame at each interconnect device and means for transmitting a discovery response frame back to said managed hub;

means for building and maintaining an interconnect device list at said managed hub of each interconnect device that transmitted said discovery response frame back to said managed hub.

56. The system for providing security against intrusion of claim 55 wherein said means for detecting includes:

means for comparing, for each port, a source address of a station attempting to connect to said port with an authorized address list of addresses for said port and means for determining whether said source address is on said authorized address list.

57. The system for providing security against intrusion of claim 55 further including:

means for sending a trap frame by said managed hub to a network management station indicating that an intrusion has been detected on said first port; and

means for transmitting a security breach detected frame by said managed hub and having said security feature group address to each interconnect device that has an entry in said interconnect device list.

58. The system for providing security against intrusion of claim 57 wherein each interconnect device further comprises:

means for transmitting said security breach detected frame on all ports except the port on which said each interconnect device received said security breach detected frame;

means for sending a trap frame to the network management station indicating that said filter has been set as a result of receiving said security breach detected frame; and

means for transmitting a filter set frame to said managed hub.

59. The system for providing security against intrusion of claim 52 wherein said managed hub further comprises means for sending a trap frame to a network management station indicating that said filter has been set at each interconnect device.

* * * * *

US005796942A

United States Patent [19]

Esbensen

[11] **Patent Number: 5,796,942**

[45] **Date of Patent: Aug. 18, 1998**

[54] **METHOD AND APPARATUS FOR AUTOMATED NETWORK-WIDE SURVEILLANCE AND SECURITY BREACH INTERVENTION**

[75] Inventor: **Daniel Esbensen**, Kihei, Hi.

[73] Assignee: **Computer Associates International, Inc.**, Islandia, N.Y.

[21] Appl. No.: **749,352**

[22] Filed: **Nov. 21, 1996**

[51] **Int. Cl.**^{6} **G06F 11/00**; G06F 13/00

[52] **U.S. Cl.** **395/187.01**; 395/200.59

[58] **Field of Search** 395/187.01, 186, 395/200.57, 200.58, 200.59; 364/286.4

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,032,979	7/1991	Hecht et al.	395/187.01
5,101,402	3/1992	Chiu et al.	370/17
5,414,833	5/1995	Hershey et al.	395/187.01
5,488,715	1/1996	Wainwright	395/182.02
5,524,238	6/1996	Miller et al.	395/600
5,557,742	9/1996	Smaha et al.	395/186
5,606,668	2/1997	Shwed	395/187.01
5,621,889	4/1997	Lermuzeaux et al.	395/186
5,699,513	12/1997	Feigen et al.	395/187.01

OTHER PUBLICATIONS

Winkler, "A Unix Prototype for Intrusion and Anomaly Detection in Secure Networks", NESC Conference, pp. 1–10, Oct. 1990.

Sebring et al., "Expert System in Intrusion Detection : A Case Study", pp. 74–81.

Debar et al., "A Neural Network Component for an Intrusion Detection System", IEEE, pp. 240–250, 1992.

Dowell et al., "The Computer Watch Data Reduction Tool", pp. 99–108.

Snapp et al., "DIDS(Distributed Intrusion Detection System)–Motivation, Architecture, and Early Prototype", pp. 167–176.

Tener, "Discovery: An Expert System in the Commercial Data Security Environment", pp. 45–53, Computer Security Journal vol. 6, No. 1, Dec. 1986.

Avritzer et al., "Reliability Testing of Rule-Based Systems", IEEE, pp. 1–7, Sep. 1996.

Snapp, "Signature Analysis and Communication Issues in a Distributed Intrusion Detection System", Master Thesis–UCA, pp. 1–40, 1991.

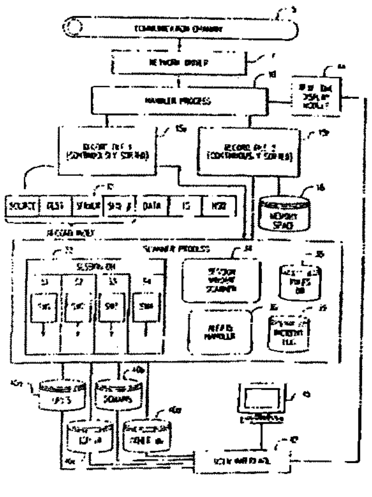
Primary Examiner—Robert W. Beausoliel, Jr.
Assistant Examiner—Scott T. Baderman
Attorney, Agent, or Firm—Thomas E. O'Connor, Jr.

[57] **ABSTRACT**

A network surveillance system includes a handler process (**10**) for capturing network packets and filtering invalid packets, a first and second continuously sorted record file (**15***a*, **15***b*), and a scanner process (**30**) for scanning all sessions occurring on the network and checking for the presence of certain rules (**38**). When a rule is met, indicating a security incident, a variety of appropriate actions may be taken, including notifying a network security officer via electronic or other mail or recording or terminating a network session. The surveillance system operates completely independently of any other network traffic and the network file server and therefore has no impact on network performance. According to a further embodiment, the invention may include remote surveillance agents (**100***a–c*) for gathering network packets at a remote location and transferring them to a server (**110**) for analysis by a network surveillance system.

20 Claims, 5 Drawing Sheets

Microfiche Appendix Included
(2 Microfiche, 64 Pages)



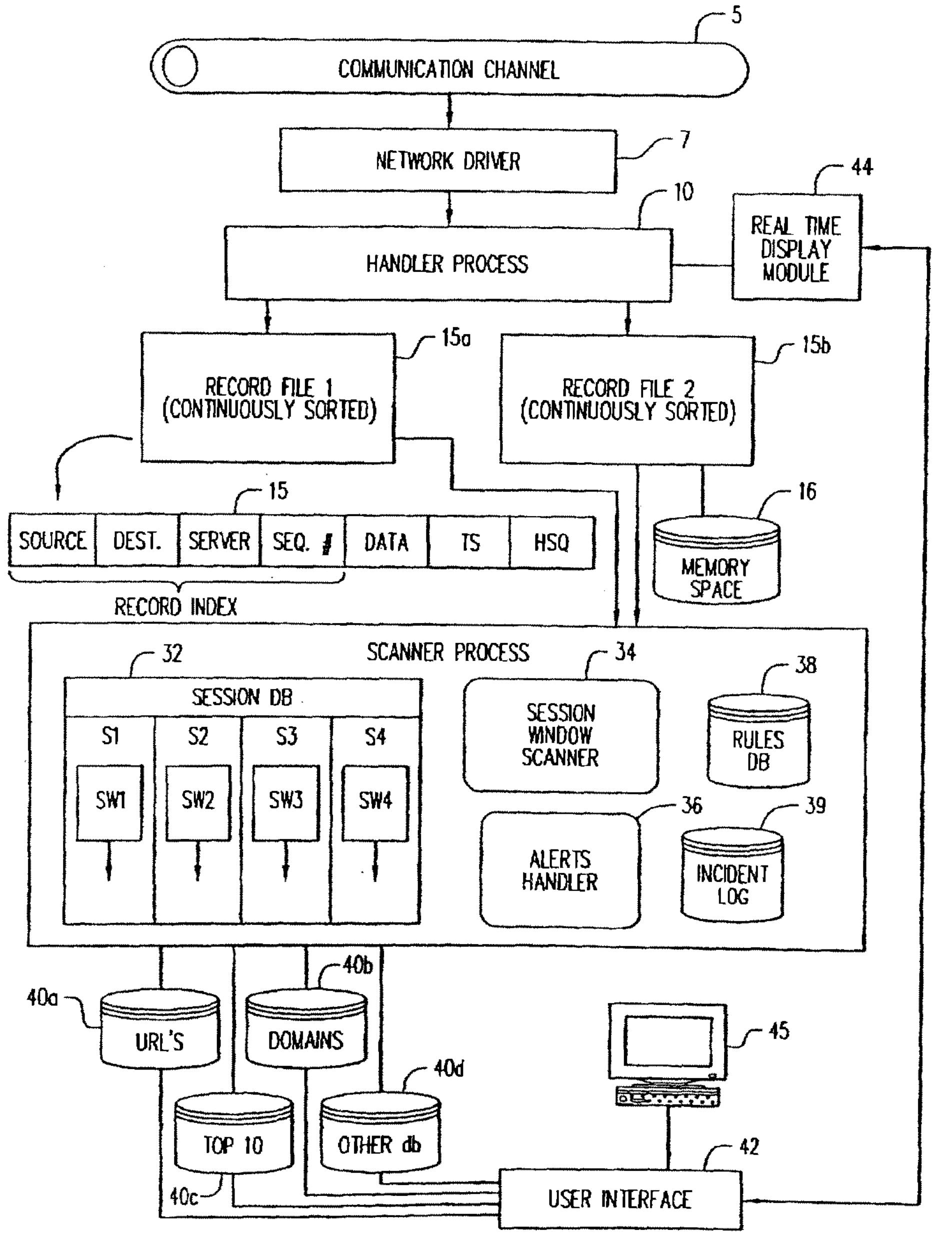

5
COMMUNICATION CHANNEL
NETWORK DRIVER
7
44
10
REAL TIME DISPLAY MODULE
HANDLER PROCESS
15a
15b
RECORD FILE 1 (CONTINUOUSLY SORTED)
RECORD FILE 2 (CONTINUOUSLY SORTED)
15
16
SOURCE
DEST.
SERVER
SEQ. #
DATA
TS
HSQ
MEMORY SPACE
RECORD INDEX
SCANNER PROCESS
32
34
38
SESSION DB
S1
S2
S3
S4
SW1
SW2
SW3
SW4
SESSION WINDOW SCANNER
RULES DB
36
39
ALERTS HANDLER
INCIDENT LOG
40a
40b
URL'S
DOMAINS
45
40d
TOP 10
OTHER db
42
40c
USER INTERFACE

FIG. 1

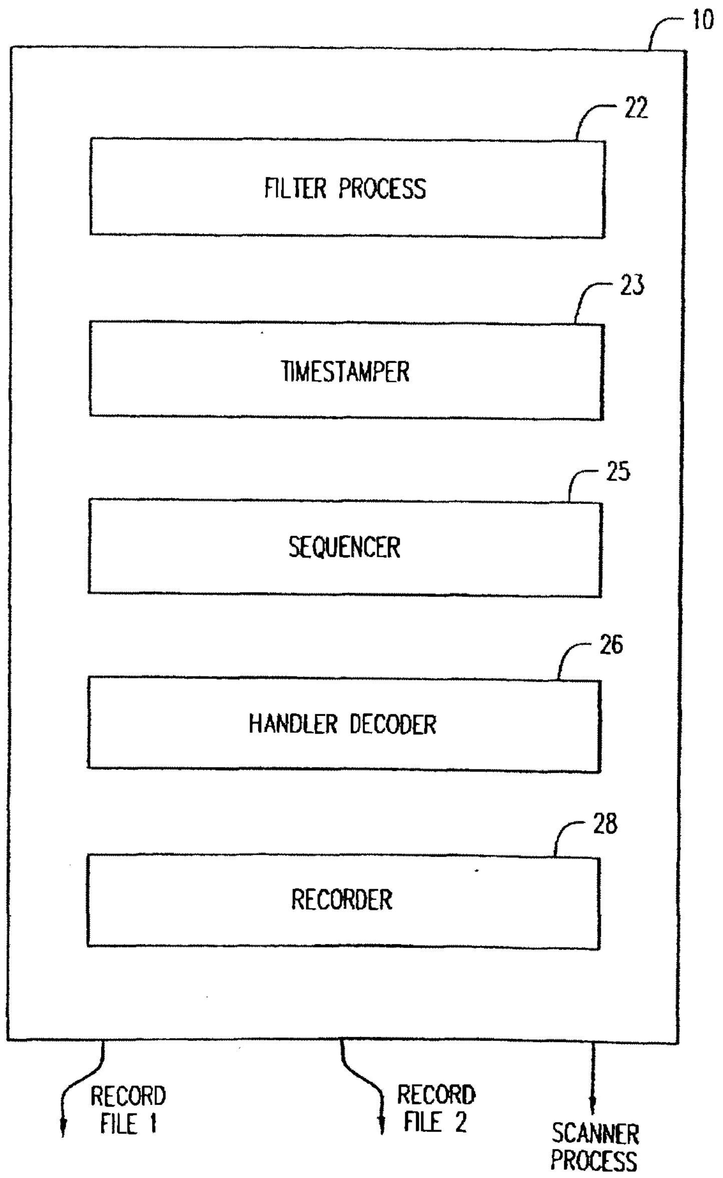

10
22
FILTER PROCESS
23
TIMESTAMPER
25
SEQUENCER
26
HANDLER DECODER
28
RECORDER
RECORD FILE 1
RECORD FILE 2
SCANNER PROCESS

FIG. 2

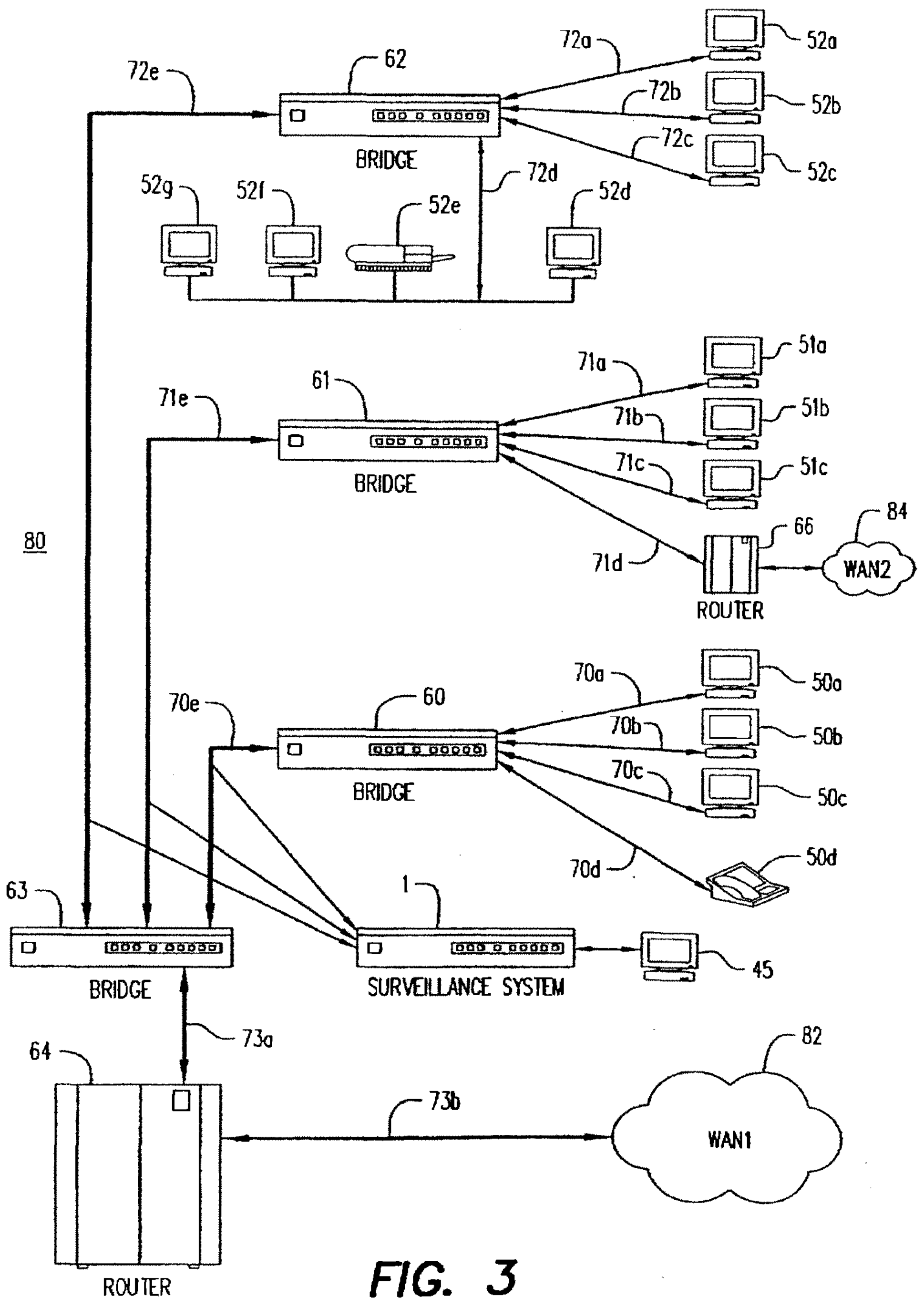

72e
62
72a
52a
72b
52b
BRIDGE
72c
52c
72d
52g
52f
52e
52d
61
71a
51a
71e
71b
51b
BRIDGE
71c
51c
80
84
66
71d
WAN2
ROUTER
70a
50a
60
70e
70b
50b
BRIDGE
70c
50c
70d
50d
63
1
BRIDGE
SURVEILLANCE SYSTEM
45
73a
64
82
73b
WAN1
ROUTER

FIG. 3

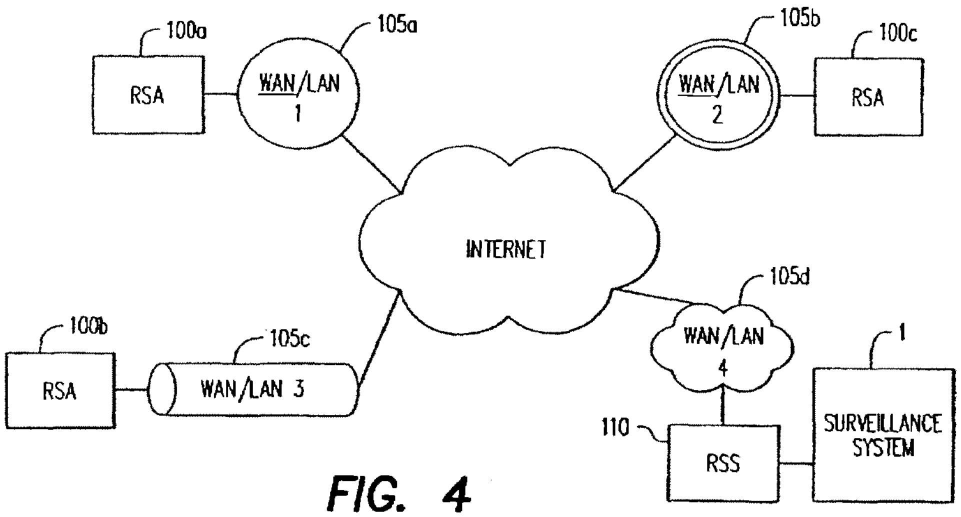

100a
RSA
105a
WAN/LAN 1
105b
WAN/LAN 2
100c
RSA
INTERNET
105d
WAN/LAN 4
100b
RSA
105c
WAN/LAN 3
1
SURVEILLANCE SYSTEM
110
RSS

FIG. 4

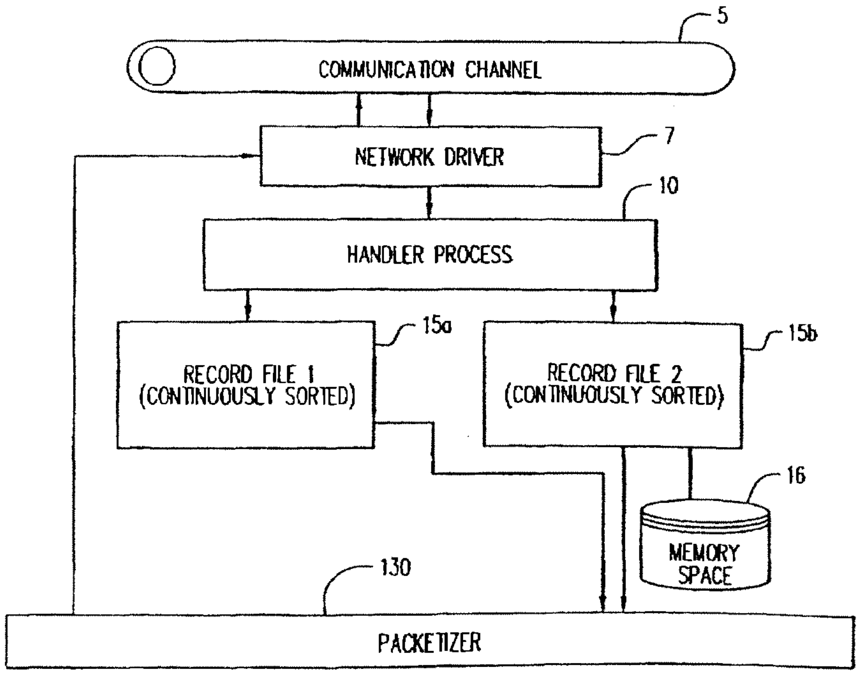

5
COMMUNICATION CHANNEL
NETWORK DRIVER
7
10
HANDLER PROCESS
RECORD FILE 1 (CONTINUOUSLY SORTED)
15a
RECORD FILE 2 (CONTINUOUSLY SORTED)
15b
16
MEMORY SPACE
130
PACKETIZER

FIG. 5

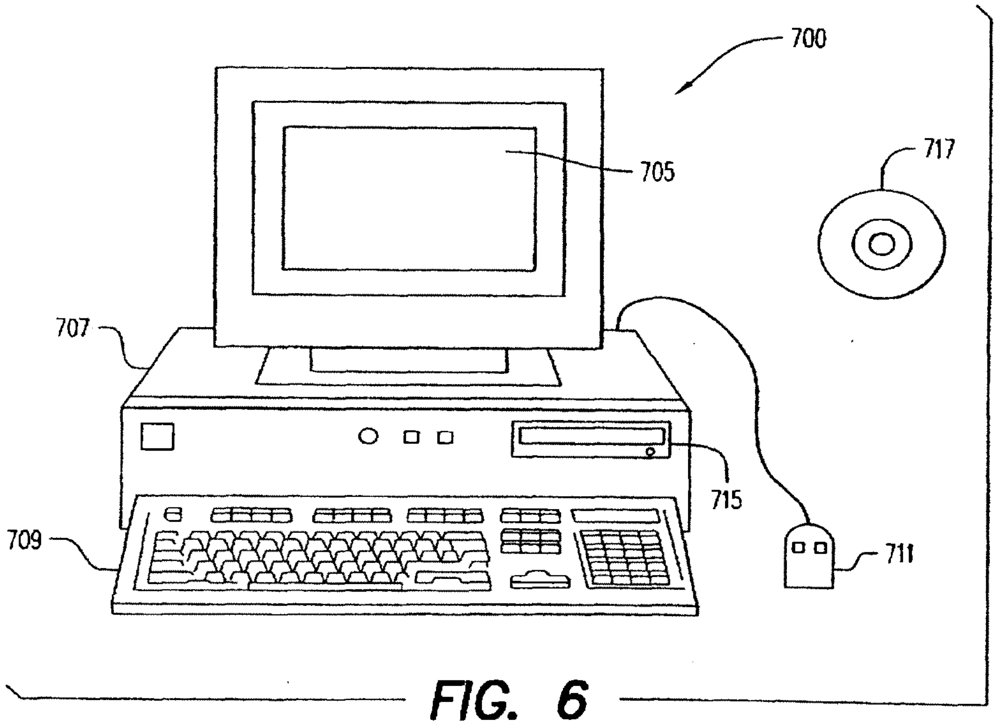


FIG. 6

METHOD AND APPARATUS FOR AUTOMATED NETWORK-WIDE SURVEILLANCE AND SECURITY BREACH INTERVENTION

COPYRIGHT NOTICE

MICROFICHE APPENDIX

A microfiche appendix including 64 frames on two fiche is included herewith.

BACKGROUND OF THE INVENTION

This invention relates to transmission of information between multiple digital devices on a network and between multiple networks on an internetwork. More particularly, this invention relates to a method and apparatus for ensuring secure network communications by conducting surveillance and checking of all or nearly all data transmitted on a network, by network session reconstruction, and by security breach intervention.

Networking Devices Standards

This specification presumes some familiarity with the general concepts, protocols, and devices currently used in LAN networking applications and in WAN internetworking applications. As these standards are widely publicly available, they will not be fully discussed here.

Generalized Lan Configuration

FIG. 3 is a generalized diagram of a local area network (LAN) **80** of a type that might be used today in a moderate-sized office or academic environment and as an example for discussion purposes of one type of network in which the present invention may be effectively employed. LANs are arrangements of various hardware and software elements that operate together to allow a number of digital devices to exchange data within the LAN and also may include internet connections to external wide area networks (WANs) such as WANs **82** and **84**. Typical modern LANs such as **80** are comprised of one to many LAN intermediate systems (ISs) such as ISs **60**–**62** that are responsible for data transmission throughout the LAN and a number of end systems (ESs) such as ESs **50***a*–*d*, **51***a*–*c*, and **52***a*–*g*, that represent the end user equipment. The ESs may be familiar end-user data processing equipment such as personal computers, workstations, modems for dial-up connections, and printers and additionally may be digital devices such as digital telephones or real-time video displays. Different types of ESs can operate together on the same LAN. Many different LAN configurations are possible, and the invention is not limited in application to the network shown in FIG. 3.

Security problems in network communications

A problem that has increasingly arisen in LAN and WAN environments is that in most prior art networks packet traffic on the line is fundamentally insecure. LANs are often designed to provide easy and flexible access to network-wide resources to any user process connected to the LAN, including processes connected through internet or dial-up connection. Within a corporate LAN, many users may have access to computer files containing data, such as account balances or financial transaction information, that may be manipulated in order to commit or cover-up crime. Firewalls are one technology to prevent unauthorized access from outside a LAN to files on the LAN. But the vast majority of computer crime is perpetrated by authorized, inside users of the LAN, accessing or manipulating data in ways that are not authorized. Firewalls offer no protection against unauthorized insider access to LAN resources.

Other security issues involve spoofing and sniffing. In a LAN segment such as **72***d*, for example, every ES on the LAN segment will hear every packet sent to any ES on that segment. In general, each ES in the network has a unique ethernet (or MAC) address, and an ES will discard any packets it hears that are not addressed to its MAC address. However, ESs are not forced by the network to discard packets not addressed to them and may operate in a promiscuous mode in which the ES reads every packet it hears on the network and passes that packet up to higher layer software running in the ES. While promiscuous mode has legitimate uses during adaptor configuration or debugging, it can also be used by an ES to read and examine all the network traffic on the network without authorization. This activity is sometimes known in the art as sniffing.

A problem related to sniffing can happen during transmissions from a LAN whereby software running on the LAN can send the outgoing packet addresses to mimic another ES's packets. This technique is known in the art as spoofing. An unscrupulous user spoofing another's packets can introduce unwanted data, such as viruses, into a packet stream being transmitted from the ES, or can hijack a user's network session and gain unauthorized access to other system resources.

A number of techniques have been proposed or implemented to enhance network security. In general, all of these techniques rely on verification of either a MAC address, and IP address, or a user identification. These techniques are limited, however, because there is no guarantee that packets being transmitted on the network have a valid MAC or IP address in their packet header and there is also no guarantee that an authorized user of a LAN will not access or manipulate LAN data in an unauthorized way.

What is needed is a simple, inexpensive, system for monitoring the activity on a network and scanning for unauthorized network activity and automatically taking action when unauthorized activity is detected. Ideally, such a technique should be implementable on a network without decreasing network performance.

For purposes of clarity, the present discussion refers to network devices and concepts in terms of specific examples. However, the method and apparatus of the present invention may operate with a wide variety of types of network devices including networks dramatically different from the specific examples illustrated in FIG. 3 and described below. It is therefore not intended that the invention be limited except as done so in the attached claims.

In many existing LAN systems, data on the network is grouped into discrete units referred to as packets, each having an indication of source and destination. While the present invention is not limited to packetized data, data is described herein in terms of packets in order to ease understanding.

SUMMARY OF THE INVENTION

The invention is an improved method and apparatus for transmitting data in a LAN. According to the present invention, a Network Security Agent™ surveillance system, is able to read all packets transmitted on a network segment, reconstruct all user sessions, and scan all user sessions for noteworthy or suspicious activity, all in real-time and without any significant impact on network performance. When any noteworthy or suspicious activity is detected, alerts are generated and appropriate intervention actions can be taken.

The present invention makes use of Packet Sniffing, Session Reconstruction, and Session Scanning in order to scan sessions for unauthorized activity and, when unauthorized activity is detected, predetermined automatic intervention action is taken. The present invention uses automatic real-time session reconstruction and scanning to accomplish network surveillance on the tens of millions of packets generated on a typical LAN each day.

In accordance with the present invention, hardware and software elements are optimally designed to be able to read all packets on the LAN in real-time and reconstruct sessions. Customized routines for reading low-level packets directly from the ethernet controller are incorporated in the invention in order to capture 100% of all network traffic.

In one embodiment, the invention includes software elements written in a language optimized for data handling and I/O. The invention includes a set of user interfaces to allow a network administrator to review data gathered by the invention and to set certain parameters.

The invention will be better understood with reference to the following drawings and detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a network surveillance system according to the present invention.

FIG. 2 is a block diagram of a handler process in accordance with an embodiment of the invention.

FIG. 3 is a diagram of a generalized LAN in which the present invention may be employed.

FIG. 4 illustrates a number of remote networks with remote surveillance system agents according to an embodiment of the invention.

FIG. 5 illustrates a remote surveillance system agent according to an embodiment of the invention.

FIG. 6 is a block diagram of a computer system which may be configured with a software embodiment in accordance with the invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Overview

FIG. 1 is a block diagram of a network surveillance system in accordance with one embodiment of the present invention. Shown in FIG. 1 is a communication channel 5 which indicates a connection to a LAN or other data communication medium. Data, either packetized or otherwise, is received from channel 5 by a network driver 7 which may include hardware and software components for quickly reading the signals on channel 5 and translating them into computer readable data. Network driver 7 may be a preexisting or custom network interface and is set to be in promiscuous mode in which it receives all or nearly all data transmitted on channel 5. Data received on network driver 7 are passed to handler process 10, which may preform some filtering or processing of the data as described below, before placing the data as records into one of files 15*a* or 15*b* as described below. Files 15*a* and 15*b* are continuously sorted as is known in the art. Scanner process 30 reads records from files 15*a*–*b* and organizes the records into a session database 32. Session data base 32 contains a sequential listing of all

packets received in a particular session. According to the invention, scanner process 30 includes a session window (SW) scanner 34. SW scanner 34 defines session windows for reading windows of data in session data base 32 and testing a set of rules 38 against those windows of data.

According to the invention, session windows are constructed so as to provide an overlapping and sliding window of data so that rules may be fully tested even if the data that would fire the rule is split on packet reception between record file 1 and record file 2. Data bases 40*a*–*d* are maintained to provide information regarding network usage parameters such as accessed URLs, accessed domains, the top ten URLs accessed, etc. A user interface 42 is designed to accept user instructions from a work station such as 45 and to display requested data to the work station 45 as described below. An optional real time display engine 44 may interact with handler process 10 to display real-time session data.

According to the invention, newly transmitted packets on channel 5 are captured even while previously captured packets are being scanned by incorporating two record files 15*a* and 15*b* which operate such that while a record file is being scanned and analyzed for surveillance incidents, the other record file is being filled with continuously sorted packets by handler process 10. Associated with the record files also may be a memory space 16 for storing larger amounts of packet data.

Handler Process

FIG. 2 illustrates the functions of handler process 10 according to one embodiment of the invention. Handler 10 reads all or a large subset of data on channel 5 and selects session packets for later reconstruction. Handler 10 communicates with scanner 30 and real time display engine 44.

Handler 10 prioritizes reading packets from channel 5, which on a busy LAN can be in excess of 50,000,000 packets a day. One embodiment of the handler uses a small state-table and is completely event driven. Reading data from network 5 packets takes the highest priority so that no desired packets are missed.

Handler process 10 includes a filtering process 22 for initial packet filtering. Filtering process 22 can be set, according to the invention, to filter out packets based on a number of criteria including filtering out invalid packets due to a bad check sum or certain identifications.

Handler process 10 also includes a timestamper 23 for adding a time stamp to each network packet received and a sequencer 25 for adding a sequence number to each packet received in order to uniquely identify each packet. Handler decoder 26 partially decodes network packets and can be programmed to handle certain internal packet compression.

Recorder 28 writes each processed data packet out as a record into a continuously sorted record files 15*a*–15*b*. Which file is written to is determined scanner process 30, as described below. A representative record 18 is shown in FIG. 1 having a number of fields including indications for a source, a destination or group of destinations, a server, a sequence number, data, a timestamp (T.S.), and a handle sequence number (HSQ).

Scanner Process

Scanner 30's primary task is session reconstruction and session scanning. At timed intervals, scanner 30 sets a flag requesting a group of packets for session reconstruction. The packets are generally provided by handler 10 from either file 15*a* or 15*b* and handler 10 begins storing newly received records in the file not being accessed by scanner 30. When scanner 30 receives the packets, it immediately proceeds to reconstruct sessions.

Sessions are reconstructed based on any combination of source and destination indications such as IP address and port (for TCP/IP) or Local Area Transport (LAT) virtual circuit and slot. Each identified session is reconstructed separately along with a session identifier. Some portion of previously reconstructed session data is maintained to allow SW scanner 34 to detect patterns that may cross record files.

Rules and Intervention Actions

The reconstructed session is passed through a series of user-defined rules 38. In one embodiment. each rule consists of simply an alert name and a pattern. When SW scanner 34 detects that a session window contains the pattern. the alert is triggered.

Associated with each alert name is a description of the alert. a list of actions to be taken when the alert is triggered. and the priority level of the alert. When the alert is triggered. an incident is logged in log 39. Incident log 39 contains identifying data of the incident such as the name of the alert. description. user login name. location (TCP/IP or LAT address/port). and a snapshot of the session-with an arrow pointing to the pattern that caused the alert to be triggered.

After logging the incident. any alert actions are taken by alert handler 36. Possible alert actions include sending email to someone or group of people containing for example the name of the triggered alert. location (TCP/IP or LAT address/port). user login name. and a snapshot of the session with an indication of the pattern that caused the alert to be triggered.

Another possible alert action includes recording the session from the alert moment forward for playback later on. The recording contains. keystroke-for-keystroke. everything that the user does that involves transmission over the network. An alert may also take action to terminate the user connection that generated it.

Scanner 30 also may handle session data base cleanup procedures—such as purging inactive login information.

Real Time Display Module

Real time display module 44 is an optional component of the invention that is in charge of displaying sessions in real-time. When real time display module 44 receives a watch message from either alert handler 36 or user interface module 42. it creates a terminal-emulation pop-up window. Each window displays a user session in real time keystroke by keystroke. In this situation. both scanner 30 and real time display module 44 will receive certain packets from handler 10. Real time display module 44 then sends a message to handler 10. requesting that packets from the watched session be duplicated and sent to real time display module 44. When watch packets are received. they are formatted and sent to the appropriate terminal-emulation pop-up window.

If the session is disconnected. a session closed message is displayed in the pop-up window and watching of the session is halted. If the user manually closes the pop-up window. session watching is also discontinued for that session.

User Interface Module

User interface module 42 provides a user interface to the network surveillance system. From module 42. sessions can be viewed. reports generated. alerts and rules defined. and session actions taken.

Module 42 communicates with real time display module 44 when session watching is requested. All other displays and actions performed by module 42 are performed through data base operations. Scanner 30 notices data base changes (such as new alerts or rules) and rebuilds its internal tables as needed.

Module 42 can be operated either with a mouse. directly from the keyboard. or by any other method for interfacing between a computer work station and a user. Extensive on-line help is provided at all decision points.

EXAMPLE

The operation of the invention may be further understood by an example. For the purposes of this example. assume that LAN 80 is a local area network in an investment management firm. The network may include a number of functions which a particular employee is authorized to use at any time from any location. including from a dial-up connection. One such function that an employee may access at any time is interoffice email functions. In addition. the LAN may include data of a sensitive nature pertaining to customer accounts. which normally would only be accessed by authorized employees during business hours while on-site at the office handling customer accounts. Standard prior art security measures. such as file access authorization. might designate certain employees to have access to this data. but would usually not limit that access based on whether the employee was connecting via a dial-up connection or whether the employee was attempting to access the data during valid business hours.

According to the current invention. a rule could be set up to monitor access to any file within the customer file structure. This rule could be a very simple rule that checked for a certain text string being passed from a client process to a server process over the network where that text string represented a file path name. To further illustrate aspects of the invention. assume that the complete file path name is divided into more than one network packet and that the two network packets are received just as scanner 30 requests a switch from record file 1 to record file 2.

Such a rule may be represented as:

```
IF          text_contains("÷atacustomer") AND
            (time()=off_hours OR connection()=dial_up)
THEN
            email(session_data, supervisor)
            terminate_session ()
ENDIF
```

According to this example. a first packet from a session S2 ending with the data "\data\cu" is transmitted on channel 5 and placed by handler 10 into record file 15*a*. before the next packet from S2 is received. scanner 30 signals to handler 10 to switch record files. Scanner 30 then reads the data in record file 1. and places data from S2 in the appropriate session database file. Session window scanner 34 then scans the text in SW2 for the above rule. and since the text is not found. the rule does not fire.

In the meantime. a second packet from session S2 beginning with the data "stomer"is transmitted on channel 5 and placed by handler 10 into record file 15*b*. When scanner 30 has fully analyzed the data from 15*a*. it switches to 15*b* and places the additional data from S2 in the appropriate session database file. Session window scanner 34 then scans the text in SW2 for the above rule. and. because SW2 includes an overlap of at least 13 bytes. the rule fires. The incident is logged in 39 and the alert is handled by handler 36.

Specific Implementation

A primary challenge of the present invention is to be able to read all data packets on the LAN in real-time. In one specific installation. an OpenVMS operating system. running on a Digital Alpha/AXP CPU at speeds of 233 Mhz to 500 Mhz was chosen to keep up with the heavy processing demands of reading 100% of a busy LAN's packets while handling session reconstruction, real-time scanning, and real-time display tasks.

Customized routines for reading low-level packets directly from a network controller were written in C using the OpenVMS' asynchronous QIO services. The real-time display module was also written in C.

For session reconstruction and real-time session scanning, one embodiment was implemented using the INTOUCH 4GL(TM) programming language, developed by the assignee of the present invention. INTOUCH 4GL is a high performance language designed specifically for data manipulation and text scanning. For use by the surveillance agent INTOUCH 4GL was enhanced by including specialized functions for high-speed pattern matching.

INTOUCH 4GL was also used for the user interface and incident tracking, reporting, data base maintenance, and recorded session playback.

Remote Surveillance Agent

FIGS. 4 and 5 illustrate a different embodiment of the invention wherein a number of remote surveillance agents (RSAs) may be utilized along with an internet in order to capture network data traffic on one site and have that traffic analyzed and sessions reconstructed at another site. FIG. 4 shows RSAs **100*a–c*** connected to different WAN/LAN networks **105*a***. According to this embodiment, RSAs **100*a–c*** collect all network data traffic from the LAN or WAN to which they are attached, but instead of fully scanning that traffic, RSAs **100*a–c*** store collected packets into a form that may be transmitted to remote surveillance server (RSS) **110**. RSS **110** receives the information for RSAs **100*a–c*** and presents this information to a surveillance system **1** according to the invention, which performs session reconstruction, rule checking, and alert handling as described above.

According to one specific embodiment RSAs **100*a–c*** collect multiple packets on their attached WAN/LAN and compress multiple packets into a single internet packet which may be transmitted back through the WAN/LAN, over the internet, to RSS **110**. According to this embodiment, RSAs **100*a–c*** can in this way allow a surveillance system **1** located in one city to monitor several WAN/LANs located in different cities simply by plugging an RSA into the remote network without making any other changes to the network.

FIG. 5 illustrates one example of an RSA according to the invention. LAN/WAN data is received and processed by handler process **10** substantially as described above and stored in one of a plurality of record files **15*a–b***. Record file data is then read by internet packetized **130**, which stores multiple LAN/WAN packets into an internet packet which is then passed to driver **7** for transmission to RSS **110** via the internet. In an alternative embodiment, LAN/WAN packets are received by an RSA and timestamped and immediately transmitted over the internet, either singly or in groups, with minimal additional processing by the RSA.

The present invention may be embodied in software instructions either recorded on a fixed media or transmitted electronically. In such a case, the surveillance system **1** of FIG. 3 will be a high performance computer system and the software instructions will cause the memory and other storage medium of computer **1** to be configured as shown in FIG. 1 and will cause the processor of computer **1** to operate in accordance with the invention.

FIG. 6 illustrates an example of a computer system used to execute the software of the present invention. FIG. 7 shows a computer system **700** which includes a monitor **705**, cabinet **707**, keyboard **709**, and mouse **711**. Cabinet **707** houses a disk drive **715** for reading a CD-ROM or other type

disk **717** and houses other familiar computer components (not shown) such as a processor, memory, disk drives, and the like, as well as an adaptor **1** for connection to a communication channel **5**.

The invention has now been explained with reference to specific embodiments. Other embodiments will be apparent to those of skill in the art. In particular, specific processing orders have been described and functions have been described as being in particular orders, however, many of these sub functions could be differently arranged without changing the essential operation of the invention. It is therefore not intended that this invention be limited, except as indicated by the appended claims.

What is claimed is:

1. A network surveillance system for conducting surveillance on a network independent of a network server comprises:

a network driver for capturing data on a network, said data not necessarily addressed to said surveillance system;

a handler process for receiving data from said network driver and storing said data in real time;

a plurality of record files for receiving network data and storing said data before further examination;

a scanner process for designating one of said plurality of record files as a receive file while reading data from another of said plurality of record files and for using said data to construct a plurality of session data streams, said session data streams providing a sequential reconstruction of network data traffic organized by session;

a session window scanner for reading a window of data in one of said plurality of session data streams;

a set of surveillance rules defining data patterns which, when met, will trigger a surveillance alert; and

an alerts handler for responding to fired rules and taking defined actions.

2. The device according to claim **1** further comprising:

a user interface allowing a user to view sessions in real time and to access a plurality of data bases containing session events maintained by said scanner process.

3. The device according to claim **1** wherein said handler process filters certain network data and adds an indication of the time when certain network data is received from the network.

4. The device according to claim **1** wherein said plurality of record files are continuously sorted according to a record index.

5. The device according to claim **1** wherein said session window includes an overlap portion of previously examined data from said session data base in order to test for rules that would apply to data contained in more than one record.

6. The device according to claim **5** wherein said session window overlap is determined by the longest text string that could trigger a rule.

7. The device according to claim **1** wherein said alerts handler may respond to an alert by transmitting a message to a specified plurality of destinations.

8. The device according to claim **1** wherein said alerts handler may respond to an alert by forcing a user session to terminate.

9. The device according to claim **1** wherein said alerts handler may respond to an alert by recording a session.

10. A fixed computer readable medium containing computer executable program code, which, when loaded into an appropriately configured computer system will cause the computer to embodiment the device of claim **1**.

11. A method for for conducting surveillance on a network comprises:

capturing data on a network;

storing said data in real time in one of a plurality of record files;

using said data to construct a plurality of session data streams, said session data streams providing a sequential reconstruction of network data traffic organized by session;

reading a window of data in one of said plurality of session data streams;

testing said window of data against a set of surveillance rules; and

responding to fired rules by taking defined interventions.

12. The method according to claim 11 further comprising presenting a view of reconstructed sessions to a user in real time.

13. The method according to claim 11 further comprising filtering certain network data packets before storing.

14. The method according to claim 11 further comprising continuously sorting record files.

15. The method according to claim 11 further comprising examining an overlap portion of previously examined data in order to test rules that would apply to data contained in more than one record.

16. The method according to claim 15 wherein said session window overlap is determined by the longest text string that could trigger a rule.

17. The method according to claim 11 further comprising responding to an alert by transmitting a message to a specified plurality of destinations.

18. The method according to claim 11 further comprising responding to an alert by forcing a user session to terminate.

19. The method according to claim 11 further comprising responding to an alert by recording a session.

20. A fixed computer readeable medium containing computer executable program code, which, when loaded into an appropriately configured computer system will cause the computer to embodiment the method of claim 11.

* * * * *

Linux FreeS/WAN Index file

This is an index file for the Linux FreeS/WAN documentation. Most files described here are in the doc directory after the distribution is unpacked and are in HTML format. If you prefer text files over HTML, see doc/README for instructions on creating them.

Files most users should read

- How to set up a simple network with FreeS/WAN. This also covers initial installation.
- Configuration of FreeS/WAN.
- relation between IPSEC and firewalls
- a list of FreeS/WAN man pages with links to HTML versions.
  They are also of course available via the *man* command.
- information on the project mailing list
- problem reporting
- Troubleshooting using our ipsec_barf(8) and ipsec_look(8) tools and other tools such as tcpdump (8) and sniffers
- a (still rudimentary) FAQ document

Distribution text files

Text files in the main distribution directory are README, INSTALL, CREDITS, CHANGES, and COPYING.

License and copyright information

All code and documentation written for this project is distributed under either the GNU General Public License (GPL) or the GNU Library General Public License. For details see COPYING.

Not all code in the distribution is ours, however. See CREDITS for details. In particular, note that the Libdes library has its own license

Printed documentation

Those who prefer documentation in printed form can, of course, print any of the HTML documents or man pages in the usual way, and are free to write whatever scripts they like to reformat them in the process. (We would like to see any interesting scripts you come up with. Please post them, or a suitable pointer, to the mailing list. Of course, if they have any code specifically related to cryptography, you must consult your local export laws first.)

We also provide three files designing for use with the "make book" command in the Amaya web browser/editor from the World Wide Web Consortium.

Going to any of these files with Amaya and clicking on the "make book" command will give you one large file, with an automatically generated table of contents, for browsing or printing:

- Setup, configuration, troubleshooting

- Background information
- Man pages

These files are also usable without Amaya. Without Amaya, you cannot build the single large "book" file, but you can follow the links to its components.

Project background information

- Our project leader's rationale for starting this
- Project Overview: goals, protocols, and components
- Lists of IPSEC features
  - implemented in Linux FreeS/WAN
  - not yet in Linux FreeS/WAN
- DES and its vulnerability to cracking.
- Export laws

Reference information

Automatically generated link files

- Table of Contents for HTML documentation
- Permuted index of HTML files

Run 'make' in the doc directory if these files aren't there.

Other reference files

- Roadmap, where things are in the distribution
- Glossary of terms and acronyms
- Bibliography
- Web links for
  - Linux FreeS/WAN project
  - IPSEC protocols
  - Linux
  - Cryptography and security
- Mailing lists
- List of IPSEC and other security RFCs

Specialised information

- Troubleshooting using our ipsec_barf(8) and ipsec_look(8) tools and other tools such as tcpdump (8) and sniffers
- Compatibility information culled from the mailing list on using FreeS/WAN with:
  - Linux distibutions other than Redhat
  - CPUs other than Intel architecture
  - other IPSEC implementations
- Configuration for setups with unusual requirements such as:
  - extruded subnet (IP sees one network, but there are two or more physical sites involved).

This file is part of the documentation for the Linux FreeS/WAN project.
See the documentation index or project home page for more information.

Swan: Securing the Internet against Wiretapping by project founder John Gilmore

My project for 1996 was to **secure 5% of the Internet traffic against passive wiretapping**. It didn't happen in 1996, so I'm still working on it in 1997, 1998, and 1999! If we get 5% in 1999 or 2000, we can secure 20% the next year, against both active and passive attacks; and 80% the following year. Soon the whole Internet will be private and secure. The project is called S/WAN or S/Wan or Swan for Secure Wide Area Network; since it's free software, we call it FreeSwan to distinguish it from various commercial implementations. RSA came up with the term "S/WAN". Our main web site is at http://www.xs4all.nl/~freeswan/. Want to help?

The idea is to deploy PC-based boxes that will sit between your local area network and the Internet (near your firewall or router) which opportunistically encrypt your Internet packets. Whenever you talk to a machine (like a Web site) that doesn't support encryption, your traffic goes out "in the clear" as usual. Whenever you connect to a machine that does support this kind of encryption, this box automatically encrypts all your packets, and decrypts the ones that come in. In effect, each packet gets put into an "envelope" on one side of the net, and removed from the envelope when it reaches its destination. This works for all kinds of Internet traffic, including Web access, Telnet, FTP, email, IRC, Usenet, etc.

The encryption boxes are standard PC's that use freely available Linux software that you can download over the Internet or install from a cheap CDROM.

This wasn't just my idea; lots of people have been working on it for years. The encryption protocols for these boxes are called IPSEC (IP Security). They have been developed by the IP Security Working Group of the Internet Engineering Task Force, and will be a standard part of the next major version of the Internet protocols (IPv6). For today's (IP version 4) Internet, they are an option.

The Internet Architecture Board and Internet Engineering Steering Group have taken a strong stand that the Internet should use powerful encryption to provide security and privacy. I think these protocols are the best chance to do that, because they can be deployed very easily, without changing your hardware or software or retraining your users. They offer the best security we know how to build, using the Triple-DES, RSA, and Diffie-Hellman algorithms.

This "opportunistic encryption box" offers the "fax effect". As each person installs one for their own use, it becomes more valuable for their neighbors to install one too, because there's one more person to use it with. The software automatically notices each newly installed box, and doesn't require a network administrator to reconfigure it. Instead of "virtual private networks" we have a "REAL private network"; we add privacy to the real network instead of layering a manually-maintained virtual network on top of an insecure Internet.

Deployment of IPSEC

The US government would like to control the deployment of IP Security with its crypto export laws. This isn't a problem for my effort, because the cryptographic work is happening outside the United States. A foreign philanthropist, and others, have donated the resources required to add these protocols to the Linux operating system. Linux is a complete, freely available operating system for IBM PC's and several kinds of workstation, which is compatible with Unix. It was written by Linus Torvalds, and is still maintained by a talented team of expert programmers working all over the world and coordinating over the Internet. Linux is distributed under the GNU Public License, which gives everyone the right to copy it, improve it, give it to their friends, sell it commercially, or do just about anything else with it, without paying anyone for the privilege.

Organizations that want to secure their network will be able to put two Ethernet cards into an IBM PC, install Linux on it from a $30 CDROM or by downloading it over the net, and plug it in between their Ethernet and their Internet link or firewall. That's all they'll have to do to encrypt their Internet traffic everywhere outside their own local area network.

Travelers will be able to run Linux on their laptops, to secure their connection back to their home network (and to everywhere else that they connect to, such as customer sites). Anyone who runs Linux on a standalone PC will also be able to secure their network connections, without changing their application software or how they operate their computer from day to day.

There will also be numerous commercially available firewalls that use this technology. RSA Data Security is coordinating the S/Wan (Secure Wide Area Network) project among more than a dozen vendors who use these protocols. There's a compatability chart that shows which vendors have tested their boxes against which other vendors to guarantee interoperatility.

Eventually it will also move into the operating systems and networking protocol stacks of major vendors. This will probably take longer, because those vendors will have to figure out what they want to do about the export controls.

Current status

My initial goal of securing 5% of the net by Christmas '96 was not met. It was an ambitious goal, and inspired me and others to work hard, but was ultimately too ambitious. The protocols were in an early stage of development, and needed a lot more protocol design before they could be implemented. As of April 1999, we have released version 1.0 of the software (freeswan-1.0.tar.gz), which is suitable for setting up Virtual Private Networks using shared secrets for authentication. It does not yet do opportunistic encryption, or use DNSSEC for authentication; those features are coming in a future release.

Protocols

The low-level encrypted packet formats are defined. The system for publishing keys and providing secure domain name service is defined. The IP Security working group has settled on an NSA-sponsored protocol for key agreement (called ISAKMP/Oakley), but it is still being worked on, as the protocol and its documentation is too complex and incomplete. There are prototype implementations of ISAKMP. The protocol is not yet defined to enable opportunistic encryption or the use of DNSSEC keys.

Linux Implementation

The Linux implementation has reached its first major release and is ready for production use in manually-configured networks, using Linux kernel version 2.0.36.

Domain Name System Security

There is now a release of BIND 8.2 that includes most DNS Security features.

The first prototype implementation of Domain Name System Security was funded by DARPA as part of their Information Survivability program. Trusted Information Systems wrote a modified version of BIND, the widely-used Berkeley implementation of the Domain Name System.

TIS, ISC, and I merged the prototype into the standard version of BIND. The first production version that supports KEY and SIG records is **bind-4.9.5**. This or any later version of BIND will do for publishing keys. It is available from the Internet Software Consortium. This version of BIND is not export-controlled since it does not contain any cryptography. Later releases starting with BIND 8.2 include cryptography for authenticating DNS records, which is also exportable. Better documentation is needed.

Why?

Because I can. I have made enough money from several successful startup companies, that for a while I don't have to work to support myself. I spend my energies and money creating the kind of world that I'd like to live in and that I'd like my (future) kids to live in. Keeping and improving on the civil rights we have in the United States, as we move more of our lives into cyberspace, is a particular goal of mine.

What You Can Do

Install the latest BIND at your site.

You won't be able to publish any keys for your domain, until you have upgraded your copy of BIND. The thing you really need from it is the new version of *named*, the Name Daemon, which knows about the new KEY and SIG record types. So, download it from the Internet Software Consortium and install it on your name server machine (or get your system administrator, or Internet Service Provider, to install it). Both your primary DNS site and all of your secondary DNS sites will need the new release before you will be able to publish your keys. You can tell which sites this is by running the Unix command "dig MYDOMAIN ns" and seeing which sites are mentioned in your NS (name server) records.

Set up a Linux system and run a 2.0.x kernel on it

Get a machine running Linux (say the 5.2 release from Red Hat). Give the machine two Ethernet cards.

Install the Linux IPSEC (Freeswan) software

If you're an experienced sysadmin or Linux hacker, install the freeswan-1.0 release, or any later release or snapshot. These releases do NOT provide automated "opportunistic" operation; they must be manually configured for each site you wish to encrypt with.

Get on the linux-ipsec mailing list

The discussion forum for people working on the project, and testing the code and documentation, is: linux-ipsec@clinet.fi. To join this mailing list, send email to linux-ipsec-REQUEST@clinet.fi containing a line of text that says "subscribe linux-ipsec". (You can later get off the mailing list the same way -- just send "unsubscribe linux-ipsec").

Check back at this web page every once in a while

I update this page periodically, and there may be new information in it that you haven't seen. My intent is to send email to the mailing list when I update the page in any significant way, so subscribing to the list is an alternative.

Would you like to help? I can use people who are willing to write documentation, install early releases for testing, write cryptographic code outside the United States, sell pre-packaged software or systems including this technology, and teach classes for network administrators who want to install this technology. To offer to help, send me email at gnu@toad.com. Tell me what country you live in and what your citizenship is (it matters due to the export control laws; personally I don't care). Include a copy of your resume and the URL of your home page. Describe what you'd like to do for the project, and what you're uniquely qualified for. Mention what other volunteer projects you've been involved in (and how they worked out). Helping out will require that you be able to commit to doing particular things, meet your commitments, and be responsive by email. Volunteer projects just don't work without those things.

Related projects

IPSEC for NetBSD
: This prototype implementation of the IP Security protocols is for another free operating system. Download BSDipsec.tar.gz.

IPSEC for OpenBSD
: This prototype implementation of the IP Security protocols is for yet another free operating system. It is directly integrated into the OS release, since the OS is maintained in Canada, which has freedom of speech in software.

---

gnu@toad.com, gnu@eff.org, my home page
An equal opportunistic encryptor.

This file is part of the documentation for the Linux FreeS/WAN project.
See the documentation index or project home page for more information.

Glossary for the Linux FreeS/WAN project

Entries are in alphabetical order. Some entries are only one line or one paragraph long. Others run to several paragraphs. I have tried to put the essential information in the first paragraph so you can skip the other paragraphs if that seems appropriate.

Jump to a letter in the glossary

numeric A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Other glossaries

Other glossaries which overlap this one include:

- glossary portion of the Cryptography FAQ
- an extensive crytographic glossary on Terry Ritter's page.
- The NSA's glossary of computer security on the SANS Institute site.
- an Internet Draft Crypto Glossary
- the IETF provide a glossary of Internet terms as RFC 1983
- a small glossary for Internet Security at PC magazine
- The glossary from Richard Smith's book Internet Cryptography

More general glossary or dictionary information:

- Free Online Dictionary of Computing (FOLDOC)
    - North America
    - Europe
    - Japan

  There are many more mirrors of this dictionary.
- CRC dictionary of Computer Science
- The Jargon File, the definitive resource for hacker slang and folklore
    - North America
    - Holland
    - home page

  There are also many mirrors of this. See the home page for a list.
- A general technology glossary
- An online dictionary resource page with pointers to many dictionaries for many languages
- A search engine that accesses several hundred online dictionaries
- O'Reilly Dictionary of PC Hardware and Data Communications Terms

Definitions

3DES (Triple DES)

Using three DES encryptions on a single data block, with at least two different keys, to get higher security than is available from a single DES pass. The three-key version of 3DES is the default encryption algorithm for Linux FreeS/WAN.

IPSEC always does 3DES with three different keys, as required by RFC 2451. For an explanation of the two-key variant, see two key triple DES. Both use an EDE encrypt-decrypt-encrpyt sequence of operations.

Single DES is insecure.

Double DES is ineffective. Using two 56-bit keys, one might expect an attacker to have to do 2^{112} work to break it. In fact, only 2^{57} work is required with a meet-in-the-middle attack, though a large amount of memory is also required. Triple DES is vulnerable to a similar attack, but that just reduces the work factor from the 2^{168} one might expect to 2^{112}. That provides adequate protection against brute force attacks, and no better attack is known.

3DES can be somewhat slow compared to other ciphers. It requires three DES encryptions per block. DES was designed for hardware implementation and includes some operations which are difficult in software. However, the speed we get is quite acceptable for many uses. See benchmarks below for details.

Active attack

An attack in which the attacker does not merely eavesdrop (see passive attack) but takes action to change, delete, reroute, add, forge or divert data. Perhaps the best-known active attack is man-in-the-middle. In general, authentication is a useful defense against active attacks.

AES

The Advanced Encryption Standard, a new block cipher standard to replace DES being developed by NIST, the US National Institute of Standards and Technology. DES used 64-bit blocks and a 56-bit key. AES ciphers use a 128-bit block and are required to support 128, 192 and 256-bit keys. Some of them support other sizes as well. The larger block size helps resist birthday attacks while the large key size prevents brute force attacks.

Fifteen proposals meeting NIST's basic criteria were submitted in 1998 and subjected to intense discussion and analysis, "round one" evaluation. In August 1999, NIST narrowed the field to five "round two" candidates:

- Mars from IBM
- RC6 from RSA
- Rijndael from two Belgian researchers
- Serpent, a British-Norwegian-Israeli research collaboration
- Twofish from the consulting firm Counterpane

We expect IPSEC will eventually use the AES winner, and we expect to see a winner (or more than one; there is an ongoing discussion on that point) declared in the summer of 2000.

Adding one or more AES ciphers to Linux FreeS/WAN would be useful undertaking, and considerable freely available code exists to start from. One complication is that our code is built for a 64-bit block cipher and AES uses a 128-bit block. Volunteers via the mailing list would be welcome.

For more information, see the NIST AES home page or the Block Cipher Lounge AES page. For code and benchmarks see Brian Gladman's page .

AH
: The IPSEC Authentication Header, added after the IP header. For details, see our IPSEC Overview document and/or RFC 2402.

Alice and Bob
: A and B, the standard example users in writing on cryptography and coding theory. Carol and Dave join them for protocols which require more players.

Bruce Schneier extends these with many others such as Eve the Eavesdropper and Victor the Verifier. His extensions seem to be in the process of becoming standard as well. See page 23 of Applied Cryptography

Alice and Bob have an amusing biography on the web.

ARPA
: see DARPA

ASIO
: Australian Security Intelligence Organisation.

Asymmetric cryptography
: See public key cryptography.

Authentication
: Ensuring that a message originated from the expected sender and has not been altered on route. IPSEC uses authentication in two places:

- authenticating the players in IKE's Diffie-Hellman key exchanges to prevent man-in-the-middle attacks. This can be done in a number of ways. The methods supported by FreeS/WAN are discussed in our configuration document.
- authenticating packets on an established SA, either with a separate authentication header or with the optional authentication in the ESP protocol. In either case, packet authentication uses a hashed message athentication code technique.

Outside IPSEC, passwords are perhaps the most common authentication mechanism. Their function is essentially to authenticate the person's identity to the system. Passwords are generally only as secure as the network they travel over. If you send a cleartext password over a tapped phone line or over a network with a packet sniffer on it, the security provided by that password becomes zero. Sending an encrypted password is no better; the attacker merely records it and reuses it at his convenience. This is called a replay attack.

A common solution to this problem is a challenge-response system. This defeats simple eavesdropping and replay attacks. Of course an attacker might still try to break the cryptographic algorithm used, or the random number generator.

Automatic keying
: A mode in which keys are automatically generated at connection establisment and new keys automaically created periodically thereafter. Contrast with manual keying in which a single stored key is used.

IPSEC uses the Diffie-Hellman key exchange protocol to create keys. An authentication mechansim is required for this. The methods supported by FreeS/WAN are discussed in our configuration document.

Having an attacker break the authentication is emphatically not a good idea. An attacker that breaks authentication, and manages to subvert some other network entities (DNS, routers or gateways), can use a man-in-the middle attack to break the security of your IPSEC connections.

However, having an attacker break the authentication in automatic keying is not quite as bad as losing the key in manual keying.

- An attacker who reads /etc/ipsec.conf and gets the keys for a manually keyed connection can, without further effort, read all messages encrypted with those keys, including any old messages he may have archived.
- Automatic keying has a property called perfect forward secrecy. An attacker who breaks the authentication gets none of the automatically generated keys and cannot immediately read any messages. He has to mount a successful man-in-the-middle attack in real time before he can read anything. He cannot read old archived messages at all and will not be able to read any future messages not caught by man-in-the-middle tricks.

That said, the secrets used for authentication, stored in ipsec.secrets(5), should still be protected as tightly as cryptographic keys.

Bay Networks
: A vendor of routers, hubs and related products, now a subsidiary of Northern Telecom. Interoperation between their IPSEC products and Linux FreeS/WAN was problematic at last report; see our compatibility document.

benchmarks
: Our default block cipher, triple DES, is slower than many alternate ciphers that might be used. Speeds achieved, however, seem adequate for many purposes. For example, the assembler code from the LIBDES library we use encrypts 1.6 megabytes per second on a Pentium 200, according to the test program supplied with the library.

The University of Wales at Aberystwyth has done quite detailed tests and put their results on the web.

Even a 486 can handle a T1 line, according to this mailing list message:

```
Subject: Re: linux-ipsec: IPSec Masquerade
   Date: Fri, 15 Jan 1999 11:13:22 -0500
   From: Michael Richardson

. . . A 486/66 has been clocked by Phil Karn to do
10Mb/s encryption.. that uses all the CPU, so half that to get some CPU,
and you have 5Mb/s. 1/3 that for 3DES and you get 1.6Mb/s....
```

From an Internet Draft *The ESP Triple DES Transform*:

```
   Phil Karn has tuned DES-EDE3-CBC software to achieve 6.22 Mbps with a
   133 MHz Pentium.  Other DES speed estimates may be found at
   [Schneier95, page 279].  Your milage may vary.
```

If you want to measure the loads FreeS/WAN puts on a system, note that tools such as top or measurements such as load average are more-or-less useless for this. They are not designed to measure something that does most of its work inside the kernel.

BIND
: Berkeley Internet Name Daemon, a widely used implementation of DNS (Domain Name Service). See our bibliography for a useful reference. See the BIND home page for more information and the latest version.

Birthday attack
: A cryptographic attack based on the mathematics exemplified by the birthday paradox. This math turns up whenever the question of two cryptographic operations producing the same result becomes an issue:

- collisions in message digest functions.
- identical output blocks from a block cipher
- repetition of a challenge in a challenge-response system

Resisting such attacks is part of the motivation for:

- hash algorithms such as SHA and RIPEMD-160 giving a 160-bit result rather than the 128 bits of MD4, MD5 and RIPEMD-128.
- AES block ciphers using a 128-bit block instead of the 64-bit block of most current ciphers
- IPSEC using a 32-bit counter for packets sent on an automatically keyed SA and requiring that the connection always be rekeyed before the counter overflows.

Birthday paradox
: Not really a paradox, just a rather counter-intuitive mathematical fact. In a group of 23 people, the chance of a least one pair having the same birthday is over 50%.

The second person has 1 chance in 365 (ignoring leap years) of matching the first. If they don't match, the third person's chances of matching one of them are 2/365. The 4th, 3/365, and so on. The total of these chances grows more quickly than one might guess.

Block cipher
: A symmetric cipher which operates on fixed-size blocks of plaintext, giving a block of ciphertext for each. Contrast with stream cipher. Block ciphers can be used in various modes when multiple block are to be encrypted.

DES is among the the best known and widely used block ciphers, but is now obsolete. Its 56-bit key size makes it highly insecure today. Triple DES is the default transform for Linux FreeS/WAN because it is the only cipher which is both required in the RFCs and apparently secure.

The current generation of block ciphers -- such as Blowfish, CAST-128 and IDEA -- all use 64-bit blocks and 128-bit keys. The next generation, AES, uses 128-bit blocks and supports key sizes up to 256 bits.

The Block Cipher Lounge web site has more information.

Blowfish
: A block cipher using 64-bit blocks and keys of up to 448 bits, designed by Bruce Schneier and used in several products.

This is not required by the IPSEC RFCs and not currently used in Linux FreeS/WAN.

Brute force attack (exhaustive search)
: Breaking a cipher by trying all possible keys. This is always possible in theory (except against a one-time pad), but it becomes practical only if the key size is inadequate. For an important

example, see our document on the insecurity of DES with its 56-bit key. For an analysis of key sizes required to resist plausible brute force attacks, see this paper.

Longer keys protect against brute force attacks. Each extra bit in the key doubles the number of possible keys and therefore doubles the work a brute force attack must do. A large enough key defeats **any** brute force attack.

For example, the EFF's DES Cracker searches a 56-bit key space in an average of a few days. Let us assume an attacker that can find a 64-bit key (256 times harder) by brute force search in a second (a few hundred thousand times faster). For a 96-bit key, that attacker needs 2^{32} seconds, just over a century. Against a 128-bit key, he needs 2^{32} centuries or about 400,000,000,000 years. Your data is then obviously secure against brute force attacks. Even if our estimate of the attacker's speed is off by a factor of a million, it still takes him 400,000 years to crack a message.

This is why

- single DES is now considered dangerously insecure
- any cipher we add to Linux FreeS/WAN will have *at least* a 90-bit key
- all of the current generation of block ciphers use a 128-bit or longer key
- AES ciphers support keysizes 128, 192 and 256 bits

Cautions:
Inadequate keylength always indicates a weak cipher but it is important to note that *adequate keylength does not necessarily indicate a strong cipher*. There are many attacks other than brute force, and adequate keylength *only* guarantees resistance to brute force. Any cipher, whatever its key size, will be weak if design or implementation flaws allow other attacks.

Also, *once you have adequate keylength* (somewhere around 90 or 100 bits), *adding more key bits make no practical difference*, even against brute force. Consider our 128-bit example above that takes 400 billion years to break by brute force. Do we care if an extra 16 bits of key put that into the quadrillions? No. What about 16 fewer bits reducing it to the 112-bit security level of Triple DES, which our example attacker could break in just over a billion years? No again, unless we're being really paranoid about safety margins.

There may be reasons of convenience in the design of the cipher to support larger keys. For example Blowfish allows up to 448 bits and RC4 up to 2048, but beyond 100-odd bits it makes no difference to practical security.

Bureau of Export Administration
see BXA

BXA
The US Commerce Department's **B**ureau of Export Administration which administers the EAR Export Administration Regulations controling the export of, among other things, cryptography.

CA
Certification Authority, an entity in a public key infrastructure that can certify keys by signing them. Usually CAs form a hierarchy. The top of this hierarchy is called the root CA.

See Web of Trust for an alternate model.

CAST-128
A block cipher using 64-bit blocks and 128-bit keys, described in RFC 2144 and used in products such as Entrust and recent versions of PGP.

This is not required by the IPSEC RFCs and not currently used in Linux FreeS/WAN.

CAST-256
: Entrust's candidate cipher for the AES standard, largely based on the CAST-128 design.

CBC mode
: Cipher Block Chaining mode, a method of using a block cipher in which for each block except the first, the result of the previous encryption is XORed into the new block before it is encrypted. CBC is the mode used in IPSEC.

An initialisation vector (IV) must be provided. It is XORed into the first block before encryption. The IV need not be secret but should be different for each message and unpredictable.

Certification Authority
: see CA

Cipher Modes
: Different ways of using a block cipher when encrypting multiple blocks.

Four standard modes were defined for DES in FIPS 81. They can actually be applied with any block cipher.

ECB	Electronic CodeBook	encrypt each block independently
CBC	Cipher Block Chaining	XOR previous block ciphertext into new block plaintext before encrypting new block
CFB	Cipher FeedBack	
OFB	Output FeedBack	

IPSEC uses CBC mode since this is only marginally slower than ECB and is more secure. In ECB mode the same plaintext always encrypts to the same ciphertext, unless the key is changed. In CBC mode, this does not occur.

Various other modes are also possible, but none of them are used in IPSEC.

Challenge-response authentication
: An authentication system in which one player generates a random number, encrypts it and sends the result as a challenge. The other player decrypts and sends back the result. If the result is correct, that proves to the first player that the second player knew the appropriate secret, required for the decryption.

Variations on this technique exist using public key or symmetric cryptography. Some provide two-way authentication, assuring each player of the other's identity.

Because the random number is different each time, this defeats simple eavesdropping and replay attacks. Of course an attacker might still try to break the cryptographic algorithm used, or the random number generator.

Ciphertext
: The encrypted output of a cipher, as opposed to the unencrypted plaintext input.

Cisco
: A vendor of routers, hubs and related products. Their IPSEC products interoperate with Linux FreeS/WAN; see our compatibility document.

Conventional cryptography
    See symmetric cryptography
Collision resistance
    The property of a message digest algorithm which makes it hard for an attacker to find or construct two inputs which hash to the same output.
Copyleft
    see GNU General Public License
CSE
    Communications Security Establishment, the Canadian organisation for signals intelligence.
DARPA (sometimes just ARPA)
    The US government's **D**efense **A**dvanced **R**esearch **P**rojects **A**gency. Projects they have funded over the years have included the Arpanet which evolved into the Internet, the TCP/IP protocol suite (as a replacement for the original Arpanet suite), the Berkeley 4.x BSD Unix projects, and Secure DNS.

    For current information, see their web site.

Denial of service (DOS) attack
    An attack that aims at denying some service to legitimate users of a system, rather than providing a service to the attacker.

- One variant is a flooding attack, overwhelming the system with too many packets, to much email, or whatever.
- A closely related variant is a resource exhaustion attack. For example, consider a "TCP SYN flood" attack. Setting up a TCP connection involves a three-packet exchange:
    - Initiator: Connection please (SYN)
    - Responder: OK (ACK)
    - Initiator: OK here too

  If the attacker puts bogus source information in the first packet, such that the second is never delivered, the responder may wait a long time for the third to come back. If responder has already allocated memory for the connection data structures, and if many of these bogus packets arrive, the responder may run out of memory.
- Another variant is to feed the system undigestible data, hoping to make it sick. For example, IP packets are limited in size to 64K bytes and a fragment carries information on where it starts within that 64K and how long it is. The "ping of death" delivers fragments that say, for example, that they start at 60K and are 20K long. Attempting to re-assemble thse without checking for overflow can be fatal.

    The two example attacks discussed were both quite effective when first discovered, capable of crashing or disabling many operating systems. They were also well-publicised, and today far fewer systems are vulnerable to them.
DES
    The **D**ata **E**ncryption **S**tandard, a block cipher with 64-bit blocks and a 56-bit key. Probably the most widely used symmetric cipher ever devised. DES has been a US government standard for their own use (only for unclassified data), and for some regulated industries such as banking, since the late 70's.

    DES is seriously insecure against current attacks.

    Linux FreeS/WAN includes DES since the RFCs require it, but our default configuration refuses to negotiate a connection using it. **We strongly recommend that single DES not be used.**

    See also 3DES and DESX, stronger ciphers based on DES.

DESX

An improved DES suggested by Ron Rivest of RSA Data Security. It XORs extra key material into the text before and after applying the DES cipher.

This is not required by the IPSEC RFCs and not currently used in Linux FreeS/WAN. DESX would be the easiest additional transform to add; there would be very little code to write. It would be much faster than 3DES and almost certainly more secure than DES. However, since it is not in the RFCs other IPSEC implementations cannot be expected to have it.

DH

see Diffie-Hellman

Diffie-Hellman (DH) key exchange protocol

A protocol that allows two parties without any initial shared secret to create one in a manner immune to eavesdropping. Once they have done this, they can communicate privately by using that shared secret as a key for a block cipher or as the basis for key exchange.

The protocol is secure against all passive attacks, but it is not at all resistant to active man-in-the-middle attacks. If a third party can impersonate Bob to Alice and vice versa, then no useful secret can be created. Authentication is a prerequisite for safe Diffie-Hellman key exchange.

IPSEC can use any of several authentication mechanisims. Those supported by FreeS/WAN are discussed in our configuration document.

Digital signature

Take a message digest of a document and encrypt it with your private key for some public key cryptosystem. I can decrypt with your public key and verify that the result matches the digest I calculate. This proves that the encrypted digest was created with your private key.

Such an encrypted message digest can be treated as a signature since it cannot be created without *both* the document *and* the private key which only you should possess. The legal issues are complex, but several countries are moving in the direction of legal recognition for digital signatures.

DNS

Domain Name Service, a distributed database through which names are associated with numeric addresses and other information in the Internet Protocol Suite. See also BIND, the Berkeley Internet Name Daemon which implements DNS services and Secure DNS. See our bibliography for a useful reference on both.

DOS attack

see Denial Of Service attack

EAR

The US government's Export Administration Regulations, administered by the Bureau of Export Administration. These have replaced the earlier ITAR regulations as the controls on export of cryptography.

ECB mode

Electronic CodeBook mode, the simplest way to use a block cipher. See Cipher Modes.

EDE

The sequence of operations normally used in either the three-key variant of triple DES used in IPSEC or the two-key variant used in some other systems.

The sequence is:

- Encrypt with key1
- Decrypt with key2
- Encrypt with key3

For the two-key version, key1=key3.

The "advantage" of this EDE order of operations is that it makes it simple to interoperate with older devices offering only single DES. Set key1=key2=key3 and you have the worst of both worlds, the overhead of triple DES with the security of single DES. Since single DES is insecure, this is a rather dubious "advantage".

The EDE two-key variant can also interoperate with the EDE three-key variant used in IPSEC; just set k1=k3.

Entrust
: A Canadian company offerring enterprise PKI products using CAST-128 symmetric crypto, RSA public key and X.509 directories.

EFF
: Electronic Frontier Foundation, an advocacy group for civil rights in cyberspace.

Encryption
: Techniques for converting a readable message (plaintext) into apparently random material (ciphertext) which cannot be read if intercepted. A key is required to read the message.

Major variants include symmetric encryption in which sender and receiver use the same secret key and public key methods in which the sender uses one of a matched pair of keys and the receiver uses the other. Many current systems, including IPSEC, are hybrids combining the two techniques.

ESP
: Encapsulated Security Payload, the IPSEC protocol which provides encryption. It can also provide authentication service and may be used with null encryption (which we do not recommend). For details see our IPSEC Overview document and/or RFC 2406.

Extruded subnet
: A situation in which something IP sees as one network is actually in two or more places.

For example, the Internet may route all traffic for a particular company to that firm's corporate gateway. It then becomes the company's problem to get packets to various machines on their subnets in various departments. They may decide to treat a branch office like a subnet, giving it IP addresses "on" their corporate net. This becomes an extruded subnet.

Packets bound for it are delivered to the corporate gateway, since as far as the outside world is concerned, that subnet is part of the corporate network. However, instead of going onto the corporate LAN (as they would for, say, the accounting department) they are then encapsulated and sent back onto the Internet for delivery to the branch office.

For information on doing this with Linux FreeS/WAN, look in our Configuration file.

Exhaustive search
: See brute force attack.

FIPS

Federal Information Processing Standard, the US government's standards for products it buys. These are issued by NIST. Among other things, DES and SHA are defined in FIPS documents. NIST have a FIPS home page.

Free Software Foundation (FSF)
An organisation to promote free software, free in the sense of these quotes from their web pages

> "Free software" is a matter of liberty, not price. To understand the concept, you should think of "free speech", not "free beer."
>
> "Free software" refers to the users' freedom to run, copy, distribute, study, change and improve the software.

See also GNU, GNU General Public License, and the FSF site.

FreeSWAN
see Linux FreeS/WAN

FSF
see Free software Foundation

GCHQ
Government Communications Headquarters, the British organisation for signals intelligence.

GILC
Global Internet Liberty Campaign, an international organisation advocating, among other things, free availability of b cryptography. They have a campaign to remove cryptographic software from the Wassenaar Arrangement.

Global Internet Liberty Campaign
see GILC.

Global Trust Register
An attempt to create something like a root CA for PGP by publishing both as a book and on the web the fingerprints of a set of verified keys for well-known users and organisations.

GMP
The GNU Multi-Precision library code, used in Linux FreeS/WAN by Pluto for public key calculations.

GNU
GNU's Not Unix, the Free Software Foundation's project aimed at creating a free system with at least the capabilities of Unix. Linux uses GNU utilities extensively.

GPG
see GNU Privacy Guard

GNU General Public License (GPL, copyleft)
The license developed by the Free Software Foundation under which Linux, Linux FreeS/WAN and many other pieces of software are distributed. The license allows anyone to redistribute and modify the code, but forbids anyone from distributing executables without providing access to source code. For more details see the file COPYING included with GPLed source distributions, including ours, or the GNU site's GPL page.

GNU Privacy Guard
An open source implementation of Open PGP as defined in RFC 2440.

GPL
see GNU General Public License.

Hash
see message digest

Hashed Message Authentication Code (HMAC)
using keyed message digest functions to authenticate a message. This differs from other uses of these functions:

- In normal usage, the hash function's internal variable are initialised in some standard way. Anyone can reproduce the hash to check that the message has not been altered.
- For HMAC usage, you initialise the internal variables from the key. Only someone with the key can reproduce the hash. A successful check of the hash indicates not only that the message is unchanged but also that the creator knew the key.

The exact techniques used in IPSEC are defined in RFC 2104. They are referred to as HMAC-MD5-96 and HMAC-SHA-96 because they output only 96 bits of the hash. This makes some attacks on the hash functions harder.

HMAC
: see Hashed Message Authentication Code

HMAC-MD5-96
: see Hashed Message Authentication Code

HMAC-SHA-96
: see Hashed Message Authentication Code

Hybrid cryptosystem
: A system using both public key and symmetric cipher techniques. This works well. Public key methods provide key management and digital signature facilities which are not readily available using symmetric ciphers. The symmetric cipher, however, can do the bulk of the encryption work much more efficiently than public key methods.

IAB
: Internet Architecture Board.

ICMP
: Internet Control Message Protocol. This is used for various IP-connected devices to manage the network.

IDEA
: International Data Encrypion Algorithm, developed in Europe as an alternative to exportable American ciphers such as DES which were too weak for serious use. IDEA is a block cipher using 64-bit blocks and 128-bit keys, and is used in products such as PGP.

  IDEA is not required by the IPSEC RFCs and not currently used in Linux FreeS/WAN.

  IDEA is patented and, with strictly limited exceptions for personal use, using it requires a license from Ascom.

IESG
: Internet Engineering Steering Group.

IETF
: Internet Engineering Task Force, the umbrella organisation whose various working groups make most of the technical decisions for the Internet. The IETF IPSEC working group wrote the RFCs we are implementing.

IKE
: Internet Key Exchange, based on the Diffie-Hellman key exchange protocol. IKE is implemented in Linux FreeS/WAN by the Pluto daemon.

Initialisation Vector (IV)
: Some cipher modes, including the CBC mode which IPSEC uses, require some extra data at the beginning. This data is called the initialisation vector. It need not be secret, but should be different for each message. Its function is to prevent messages which begin with the same text from encrypting to the same ciphertext. That might give an analyst an opening, so it is best prevented.

IP
: Internet Protocol.

IP masquerade

A method of allowing multiple machines to communicate over the Internet when only one IP address is available for their use. See the Linux masquerade resource page for details.

The client machines are set up with reserved non-routable IP addresses defined in RFC 1918. The masquerading gateway, the machine with the actual link to the Internet, rewrites packet headers so that all packets going onto the Internet appear to come from one IP address, that of its Internet interface. It then gets all the replies, does some table lookups and more header rewriting, and delivers the replies to the appropriate client machines.

To use masquerade with Linux FreeS/WAN, you must set leftfirewall=yes and/or rightfirewall=yes in the connection description in /etc/ipsec.conf.

IPng
: "IP the Next Generation", see IPv6.

IPv4
: The current version of the Internet protocol suite.

IPv6 (IPng)
: Version six of the Internet protocol suite, currently being developed. It will replace the current version four. IPv6 has IPSEC as a mandatory component.

  See this web site for more details.

IPSEC
: Internet Protocol SECurity, security functions (authentication and encryption) implemented at the IP level of the protocol stack. It is optional for IPv4 and mandatory for IPv6.

  This is the standard Linux FreeS/WAN is implementing. For more details, see our IPSEC Overview. For the standards, see RFCs listed in our RFCs document.

ISAKMP
: Internet Security Association and Key Management Protocol, defined in RFC 2408.

ITAR
: International Traffic in Arms Regulations, US regulations administered by the State Department which until recently limited export of, among other things, cryptographic technology and software. ITAR still exists, but the limits on cryptography have now been transferred to the Export Administration Regulations under the Commerce Department's Bureau of Export Administration.

IV
: see Initialisation vector

Keyed message digest
: See HMAC.

Key length
: see brute force attack

KLIPS
: Kernel IP Security, the Linux FreeS/WAN project's changes to the Linux kernel to support the IPSEC protocols.

LDAP
: Lightweight Directory Access Protocol, defined in RFCs 1777 and 1778, a method of accessing information stored in directories. LDAP is used by several PKI implementations, often with X.501 directories and X.509 certificates. It may also be used by IPSEC to obtain key certifications from those PKIs. This is not yet implemented in Linux FreeS/WAN.

LIBDES
A publicly available library of DES code, written by Eric Young, which Linux FreeS/WAN uses in both KLIPS and Pluto.

Linux
A freely available Unix-like operating system based on a kernel originally written for the Intel 386 architecture by (then) student Linus Torvalds. Once his 32-bit kernel was available, the GNU utilities made it a usable system and contributions from many others led to explosive growth.

Today Linux is a complete Unix replacement available for several CPU architectures -- Intel, DEC/Compaq Alpha, Power PC, both 32-bit SPARC and the 64-bit UltraSPARC, SrongARM, . . . -- with support for multiple CPUs on some architectures.

Linux FreeS/WAN is intended to run on all CPUs supported by Linux and is currently (February 1999) known to work on Intel, Alpha and StrongARM. See our compatibility document for details.

Linux FreeS/WAN
Our implementation of the IPSEC protocols, intended to be freely redistributable source code with a GNU GPL license and no constraints under US or other export laws. Linux FreeS/WAN is intended to interoperate with other IPSEC implementations. The name is partly taken, with permission, from the S/WAN multi-vendor IPSEC compatability effort. Linux FreeS/WAN has two major components, KLIPS (KerneL IPSEC Support) and the Pluto daemon which manages the whole thing.

See our IPSEC Overview for more detail. For the code see our primary distribution site or one of the mirror sites on this list.

Mailing list
The Linux FreeS/WAN project has an open public email list for bug reports and software development discussions. The list address is **linux-ipsec@clinet.fi**. To subscribe, send mail to majordomo@clinet.fi with a one-line message body "subscribe linux-ipsec". For more information, send majordomo the one-line message "help".

NOTE: **US citizens or residents are asked not to post code to the list, not even one-line bug fixes.** The project cannot accept code which might entangle it in US export restrictions.

For more detail, see our document on this and other mailing lists.

Man-in-the-middle attack
An active attack in which the attacker impersonates each of the legitimate players in a protocol to the other.

For example, if Alice and Bob are negotiating a key via the Diffie-Hellman key agreement, and are not using authentication to be certain they are talking to each other, then an attacker able to insert himself in the communication path can deceive both players.

Call the attacker Mallory. For Bob, he pretends to be Alice. For Alice, he pretends to be Bob. Two keys are then negotiated, Alice-to-Mallory and Bob-to-Mallory. Alice and Bob each think the key they have is Alice-to-Bob.

A message from Alice to Bob then goes to Mallory who decrypts it, reads it and/or saves a copy, re-encrypts using the Bob-to-Mallory key and sends it along to Bob. Bob decrypts successfully and sends a reply which Mallory decrypts, reads, re-encrypts and forwards to Alice.

To make this attack effective, Mallory must

- subvert some part of the network in some way that lets him carry out the deception
  possible targets: DNS, router, Alice or Bob's machine, mail server, ...
- beat any authentication mechanism Alice and Bob use
  strong authentication defeats the attack entirely; this is why IKE requires authentication
- work in real time, delivering messages without noticable delay
  not hard if Alice and Bob are using email; quite difficult in some situations.

If he manages it, however, it is devastating. He not only gets to read all the messages; he can alter messages, inject his own, forge anything he likes, . . . In fact, he controls the communication completely.

Manual keying

An IPSEC mode in which the keys are provided by the administrator. In FreeS/WAN, they are stored in /etc/ipsec.conf. The alternative, automatic keying, is preferred in most cases.

MD4

Message Digest Algorithm Four from Ron Rivest of RSA. MD4 was widely used a few years ago, but is now considered obsolete. It has been replaced by its descendants MD5 and SHA.

MD5

Message Digest Algorithm Five from Ron Rivest of RSA, an improved variant of his MD4. Like MD4, it produces a 128-bit hash. For details see RFC 1321.

MD5 is one of two message digest algorithms available in IPSEC. The other is SHA. SHA produces a longer hash and is therefore more resistant to birthday attacks, but this is not a concern for IPSEC. The HMAC method used in IPSEC is secure even if the underlying hash is not particularly strong against this attack.

Meet-in-the-middle attack

A divide-and-conquer attack which breaks a cipher into two parts, works against each separately, and compares results. Probably the best known example is an attack on double DES. This applies in principle to any pair of block ciphers, e.g. to an encryption system using, say, CAST-128 and Blowfish, but we will describe it for double DES.

Double DES encryption and decryption can be written:

```
        C = E(k2,E(k1,P))
        P = D(k1,D(k2,C))
```

Where C is ciphertext, P is plaintext, E is encryption, D is decryption, k1 is one key, and k2 is the other key. If we know a P, C pair, we can try and find the keys with a brute force attack, trying all possible k1, k2 pairs. Since each key is 56 bits, there are 2^{112} such pairs and this attack is painfully inefficient.

The meet-in-the middle attack re-writes the equations to calculate a middle value M:

```
        M = E(k1,P)
        M = D(k2,C)
```

Now we can try some large number of D(k2,C) decryptions with various values of k2 and store the results in a table. Then start doing E(k1,P) encryptions, checking each result to see if it is in the table.

With enough table space, this breaks double DES with 2^{57} work. The memory requirements of such attacks can be prohibitive, but there is a whole body of research literature on methods of reducing them.

Message Digest Algorithm
An algorithm which takes a message as input and produces a hash or digest of it, a fixed-length set of bits which depend on the message contents in some highly complex manner. Design criteria include making it extremely difficult for anyone to counterfeit a digest or to change a message without altering its digest. One essential property is collision resistance. The main applications are in message authentication and digital signature schemes. Widely used algorithms include MD5 and SHA. In IPSEC, message digests are used for HMAC authentication of packets.

MTU
Maximum Transmission Unit, the largest size of packet that can be sent over a link. This is determined by the underlying network, but must be taken account of at the IP level.

IP packets, which can be up to 64K bytes each, must be packaged into lower-level packets of the appropriate size for the underlying network(s) and re-assembled on the other end. When a packet must pass over multiple networks, each with its own MTU, and many of the MTUs are unknown to the sender, this becomes a fairly complex problem. See path MTU discovery for details.

Often the MTU is a few hundred bytes on serial links and 1500-odd on Ethernet. There are, however, serial link protocols which use a larger MTU to avoid packet packet fragmentation at the ethernet/serial boundary, and newer (especially gigabit) Ethernet networks sometimes support much larger packets because these are more efficient in some applications.

NAI
Network Associates, a conglomerate formed from PGP Inc., TIS, Macaffee Anti-virus products and several others. Among other things, they offer an IPSEC-based VPN.

NAT
Network Address Translation.

NIST
The US National Institute of Standards and Technology, responsible for FIPS standards including DES and its replacement, AES.

Nonce
A random value used in an authentication protocol.

Non-routable IP address
An IP address not normally allowed in the "to" or "from" IP address field header of IP packets.

Almost invariably, the phrase "non-routable address" means one of the addresses reserved by RFC 1918 for private networks:

- 10.anything
- 172.x.anything with 16 <= x <= 31
- 192.168.anything

These addresses are commonly used on private networks, e.g. behind a Linux machines doing IP masquerade. Machines within the private network can address each other with these addresses. All packets going outside that network, however, have these addresses replaced before they reach the Internet.

If any packets using these addresses do leak out, they do not go far. Most routers automatically discard all such packets.

Various other addresses -- the 127.0.0.0/8 block reserved for local use, 0.0.0.0, various broadcast and network addresses -- cannot be routed over the Internet, but are not normally included in the meaning when the phrase "non-routable address" is used.

NSA

The US National Security Agency, the American organisation for signals intelligence, the protection of US government messages and the interception and analysis of other messages. For details, see Bamford's "The Puzzle Palace".

Some history of NSA documents were declassified in response to a FOIA (Freedom of Information Act) request.

Oakley

A key determination protocol, defined in RFC 2412.

One time pad

A cipher in which the key is:

- as long as the total set of messages to be enciphered
- absolutely random
- never re-used

Given those three conditions, it can easily be proved that the cipher is perfectly secure, in the sense that an attacker with intercepted message in hand has no better chance of guessing the message than an attacker who only knows the message length. No such proof exists for any other cipher.

There are, however, several problems with this "perfect" cipher.

- It is wildly impractical for many applications. Key management is difficult or impossible.
- It is *extremely* fragile. Small changes which violate the conditions listed above do not just weaken the cipher a bit; quite often they destroy its security completely.
    - Re-using the pad weakens it to the point where it can be broken with pencil and paper. With a computer, the attack is trivially easy.
    - Using computer-generated pseudo-random numbers instead of a really random pad completely invalidates the security proof. Depending on random number generator used, this may also give an extremely weak cipher.
- If an attacker knows the plaintext and has an intercepted message, he can discover the pad. This does not matter if the attacker is just a passive eavesdropper. It gives him no plaintext he didn't already know and we don't care that he learns a pad which we'll never re-use. However, knowing the pad lets an active attacker perform a man-in-the-middle attack, replacing your message with whatever he chooses.

Outrageous marketing claims about the "unbreakable" security of various products which somewhat resemble one-time pads are common. They are a sure sign of cryptographic snake oil.

See also the one time pad FAQ.

Opportunistic encryption

A situation in which any two IPSEC-aware machines can secure their communications, without a pre-shared secret and without a common PKI. This is a long-term goal of the Linux FreeS/WAN project which we expect to acheive using Secure DNS.

P1363 standard
: An IEEE standard for public key cryptography.

Passive attack
: An attack in which the attacker only eavesdrops and attempts to analyse intercepted messages, as opposed to an active attack in which he diverts messages or generates his own.

Path MTU discovery
: The process of discovering the largest packet size which all links on a path can handle without fragmentation -- that is, without any router having to break the packet up into smaller pieces to match the MTU of its outgoing link.

This is done as follows:

- originator sends the largest packets allowed by MTU of the first link, setting the DF (don't fragment) bit in the packet header
- any router which cannot send the packet on (outgoing MTU is too small for it, and DF prevents fragmenting it to match) sends back an ICMP packet reporting the problem
- originator looks at ICMP message and tries a smaller size
- eventually, you settle on a size that can pass all routers
- thereafter, originator just sends that size and no-one has to fragment

Since this requires co-operation of many systems, and since the next packet may travel a different path, this is one of the trickier areas of IP programming. Bugs that have shown up over the years have included:

- malformed ICMP messages
- hosts that ignore or mishandle these ICMP messages
- firewalls blocking the ICMP messages so host does not see them

Since IPSEC adds a header, it increases packet size and may require fragmentation even where incoming and outgoing MTU are equal.

Perfect forward secrecy (PFS)
: A property of systems such as Diffie-Hellman key exchange which use a long-term key (the shared secret in IKE) and generate short-term keys as required. If an attacker who acquires the long-term key *provably* can

- *neither* read previous messages which he may have archived
- *nor* read future messages without performing additional successful attacks

then the system has PFS. The attacker needs the short-term keys in order to read the trafiic and merely having the long-term key does not allow him to infer those. Of course, it may allow him to conduct another attack (such as man-in-the-middle) which gives him some short-term keys, but he does not automatically get them just by acquiring the long-term key.

PFS
: see Perfect Forward Secrecy

PGP
: Pretty Good Privacy, a personal encryption system for email based on public key technology, written by Phil Zimmerman.

The 2.xx versions of PGP used the RSA public key algorithm and used IDEA as the symmetric cipher. These versions are described in RFC 1991 and in Garfinkel's book. They are freely available. There is a US version and an International version . The differences are questions of licensing; the two are fully compatible.

Since version 5, the products from PGP Inc. have used Diffie-Hellman public key methods and IDEA or CAST-128 symmetric encryption. These can verify signatures from the 2.xx versions, but cannot exchange encryted messages with them. Some 5.x and 6.x products are free for personal use. Information on all products and downloads of the free ones are available from PGP Inc. The free versions are also on the US and International sites listed above.

An IETF working group has issued RFC 2440 for an "Open PGP" standard, similar to the 5.x versions. PGP Inc. staff were among the authors. A free Gnu Privacy Guard based on that standard is now available.

PGP Inc.
: A company founded by Zimmerman, the author of PGP, now a division of NAI. See the corporate website.

Their PGP 6.5 product includes PGPnet, an IPSEC client for Macintosh or for Windows 95/98/NT.

Photuris
: Another key negotiation protocol, an alternative to IKE, described in RFCs 2522 and 2523.

PPTP
: Point-to-Point Tunneling Protocol.

PKI
: Public Key Infrastructure, the things an organisation or community needs to set up in order to make public key cryptographic technology a standard part of their operating procedures.

There are several PKI products on the market. Typically they use a hierarchy of Certification Authorities (CAs). Often they use LDAP access to X.509 directories to implement this.

See Web of Trust for a different sort of infrastructure.

PKIX
: PKI eXchange, an IETF standard that allows PKIs to talk to each other.

This is required, for example, when users of a corporate PKI need to communicate with people at client, supplier or government organisations, any of which may have a different PKI in place. I should be able to talk to you securely whenever:

- your organisation and mine each have a PKI in place
- you and I are each set up to use those PKIs
- the two PKIs speak PKIX
- the configuration allows the conversation

At time of writing (March 1999), this is not yet widely implemented but is under quite active development by several groups.

Plaintext
: The unencrypted input to a cipher, as opposed to the encrypted ciphertext output.

Pluto
: The Linux FreeS/WAN daemon which handles key exchange via the IKE protocol, connection negotiation, and other higher-level tasks. Pluto calls the KLIPS kernel code as required. For details, see the manual page ipsec_pluto(8).

Public Key Cryptography
: In public key cryptography, keys are created in matched pairs. Encrypt with one half of a pair and only the matching other half can decrypt it. This contrasts with symmetric or secret key cryptography in which a single key known to both parties is used for both encryption and decryption.

One half of each pair, called the public key, is made public. The other half, called the private key, is kept secret. Messages can then be sent by anyone who knows the public key to the holder of the private key. Encrypt with the public key and you know only someone with the matching private key can decrypt.

Public key techniques can be used to create digital signatures and to deal with key management issues, perhaps the hardest part of effective deployment of symmetric ciphers. The resulting hybrid cryptosystems use public key methods to manage keys for symmetric ciphers.

Many organisations are currently creating PKIs, public key infrastructures to make these benefits widely available.

Public Key Infrastructure

see PKI

Random

A remarkably tricky term, far too much so for me to attempt a definition here. Quite a few cryptosystems have been broken via attacks on weak random number generators, even when the rest of the system was sound.

See RFC 1750 for the theory. It will be available locally if you have downloaded our RFC bundle (which is described here). Or read it on the net.

See the manual pages for ipsec_ranbits(8) and random(4) for details of what we use.

There has recently been discussion on several mailing lists of the limitations of Linux /dev/random and of whether we are using it correctly. Those discussions are archived on the /dev/random support page.

Raptor

A firewall product for Windows NT offerring IPSEC-based VPN services. Linux FreeS/WAN interoperates with Raptor; see our Compatibility document for details. Raptor have recently merged with Axent.

RC4

Rivest Cipher four, designed by Ron Rivest of RSA and widely used. Believed highly secure with adequate key length, but often implemented with inadequate key length to comply with export restrictions.

RC6

Rivest Cipher six, RSA's AES candidate cipher.

Replay attack

An attack in which the attacker records data and later replays it in an attempt to deceive the recipient.

RFC

Request For Comments, an Internet document. Some RFCs are just informative. Others are standards.

Our list of IPSEC and other security-related RFCs is here, along with information on methods of obtaining them.

RIPEMD

A message digest algorithm. The current version is RIPEMD-160 which gives a 160-bit hash.

Root CA
: The top level Certification Authority in a hierachy of such authorities.

Routable IP address
: Most IP addresses can be used as "to" and "from" addresses in packet headers. These are the routable addresses; we expect routing to be possible for them. If we send a packet to one of them, we expect (in most cases; there are various complications) that it will be delivered if the address is in use and will cause an ICMP error packet to come back to us if not.

  There are also several classes on non-routable IP addresses.

RSA algorithm
: Rivest Shamir Adleman public key encryption method, named for its three inventors. Patented (expires in Sept. 2000) with licenses available from RSA Data Security. Widely used.

RSA Data Security
: A company founded by the inventors of the RSA public key algorithm.

SA
: Security Association, the channel negotiated by the higher levels of an IPSEC implementation and used by the lower. SAs are unidirectional; you need a pair of them for two-way communication.

  An SA is defined by three things -- the destination, the protocol (AH orESP) and the SPI, security parameters index. It is used to index other things such as session keys and intialisation vectors.

  For more detail, see our IPSEC Overview and/or RFC 2401.

Secure DNS
: A version of the DNS or Domain Name Service enhanced with authentication services. This is being designed by the IETF DNS security working group. The BIND 8.2 implementation is available for download. Another site has more information.

  IPSEC can use this plus Diffie-Hellman key exchange to bootstrap itself. This would allow opportunistic encryption. Any pair of machines which could authenticate each other via DNS could communicate securely, without either a pre-existing shared secret or a shared PKI.

  Linux FreeS/WAN will support this in a future release.

Secret key cryptography
: See symmetric cryptography

Security Association
: see SA

Sequence number
: A number added to a packet or message which indicates its position in a sequence of packets or messages. This provides some security against replay attacks.

  For automatic keying mode, the IPSEC RFCs require that the sender generate sequence numbers for each packet, but leave it optional whether the receiver does anything with them.

SHA
: Secure Hash Algorithm, a message digest algorithm developed by the NSA for use in the Digital Signature standard, FIPS number 186 from NIST. SHA is an improved variant of MD4 producing a 160-bit hash.

SHA is one of two message digest algorithms available in IPSEC. The other is MD5. Some people do not trust SHA because it was developed by the NSA. There is, as far as we know, no cryptographic evidence that SHA is untrustworthy, but this does not prevent that view from being strongly held.

Signals intelligence (SIGINT)
: Activities of government agencies from various nations aimed at protecting their own communications and reading those of others. Cryptography, cryptanalysis, wiretapping, interception and monitoring of various sorts of signals. The players include the American NSA, British GCHQ and Canadian CSE.

SKIP
: Simple Key management for Internet Protocols, an alternative to IKE developed by Sun and being marketed by their Internet Commerce Group.

Snake oil
: Bogus cryptography. See the Snake Oil FAQ or this paper by Schneier.

SPI
: Security Parameter Index, an index used within IPSEC to keep connections distinct. A Security Association (SA) is defined by destination, protocol and SPI. Without the SPI, two connections to the same gateway using the same protocol could not be distinguished.

For more detail, see our IPSEC Overview and/or RFC 2401.

SSH
: Secure SHell, an encrypting replacement for the insecure Berkeley commands whose names begin with "r" for "remote": rsh, rlogin, etc. Web site.

SSH Communications Security
: A company founded by the authors of SSH. Offices are in Finland and California. They have a toolkit for developers of IPSEC applications.

SSL
: Secure Sockets Layer, a set of encryption and authentication services for web browsers, developed by Netscape. Widely used in Internet commerce. Also known as TLS.

SSLeay
: A free implementation of SSL by Eric Young (eay) and others. Developed in Australia; not subject to US export controls.

Stream cipher
: A symmetric cipher which produces a stream of output which can be combined (often using XOR or bytewise addition) with the plaintext to produce ciphertext. Contrasts with block cipher.

IPSEC does not use stream ciphers. Their main application is link-level encryption, for example of voice, video or data streams on a wire or a radio signal.

subnet
: A group of IP addresses which are logically one network, typically (but not always) assigned to a group of physically connected machines. The range of addresses in a subnet is described using a subnet mask. See next entry.

subnet mask
: A method of indicating the addresses included in a subnet. Here are two equivalent examples:

- 101.101.101.0/24
- 101.101.101.0 with mask 255.255.255.0

The '24' is shorthand for a mask with the top 24 bits one and the rest zero. This is exactly the same as 255.255.255.0 which has three all-ones bytes and one all-zeros byte.

These indicate that, for this range of addresses, the top 24 bits are to be treated as naming a network (often referred to as "the 101.101.101.0/24 subnet") while most combinations of the low 8 bits can be used to designate machines on that network. Two addresses are reserved; 101.101.101.0 refers to the subnet rather than a specific machine while 101.101.101.255 is a broadcast address. 1 to 254 are available for machines.

It is common to find subnets arranged in a hierarchy. For example, a large company might have a /16 subnet and allocate /24 subnets within that to departments. An ISP might have a large subnet and allocate /26 subnets (64 addresses, 62 usable) to business customers and /29 subnets (8 addresses, 6 usable) to residential clients.

S/WAN

Secure Wide Area Network, a project involving RSA Data Security and a number of other companies. The goal is to ensure that all their IPSEC implementations will interoperate so that their customers can communicate with each other securely.

Symmetric cryptography

Symmetric cryptography, also referred to as conventional or secret key cryptography, relies on a *shared secret key*, identical for sender and receiver. Sender encrypts with that key, receiver decrypts with it. The idea is that an eavesdropper without the key be unable to read the messages. There are two main types of symmetric cipher, block ciphers and stream ciphers.

Symmetric cryptography contrasts with public key or asymmetric systems where the two players use different keys.

The great difficulty in symmetric cryptography is, of course, key management. Sender and receiver *must* have identical keys and those keys *must* be kept secret from everyone else. Not too much of a problem if only two people are involved and they can conveniently meet privately or employ a trusted courier. Quite a problem, though, in other circumstances.

It gets much worse if there are many people. An application might be written to use only one key for communication among 100 people, for example, but there would be serious problems. Do you actually trust all of them that much? Do they trust each other that much? Should they? What is at risk if that key is compromised? How are you going to distribute that key to everyone without risking its secrecy? What do you do when one of them leaves the company? Will you even know?

On the other hand, if you need unique keys for every possible connection between a group of 100, then each user must have 99 keys. You need either 99*100/2 = 4950 *secure* key exchanges between users or a central authority that *securely* distributes 100 key packets, each with a different set of 99 keys.

Either of these is possible, though tricky, for 100 users. Either becomes an administrative nightmare for larger numbers. Moreover, keys *must* be changed regularly, so the problem of key distribution comes up again and again. If you use the same key for many messages then an attacker has more text to work with in an attempt to crack that key. Moreover, one successful crack will give him or her the text of all those messages.

In short, the *hardest part of conventional cryptography is key management*. Today the standard solution is to build a hybrid system using public key techniques to manage keys.

TIS

Trusted Information Systems, a firewall vendor now part of NAI. Their Gauntlet product offers IPSEC VPN services. TIS implemented the first version of Secure DNS on a DARPA contract.

TLS

Transport Layer Security, a newer name for SSL.

Traffic analysis

Deducing useful intelligence from patterns of message traffic, without breaking codes or reading the messages. In one case during World War II, the British knew an attack was coming because all German radio traffic stopped. The "radio silence" order, intended to preserve security, actually gave the game away.

In an industrial espionage situation, one might deduce something interesting just by knowing that company A and company B were talking, especially if one were able to tell which departments were involved, or if one already knew that A was looking for acquisitions and B was seeking funds for expansion.

IPSEC itself does not defend against this, but carefully thought out systems using IPSEC can do so. In particular, one might want to encrypt more traffic than was strictly necessary, route things in odd ways, or even encrypt dummy packets, to confuse the analyst.

Transport mode

An IPSEC application in which the IPSEC gateway is the destination of the protected packets, a machine acts as its own gateway. Contrast with tunnel mode.

Triple DES

see 3DES

Tunnel mode

An IPSEC application in which an IPSEC gateway provides protection for packets to and from other systems. Contrast with transport mode.

Two-key Triple DES

A variant of triple DES or 3DES in which only two keys are used. As in the three-key version, the order of operations is EDE or encrypt-decrypt-encrypt, but in the two-key variant the first and third keys are the same.

3DES with three keys has 3*56 = 168 bits of key but has only 112-bit strength against a meet-in-the-middle attack, so it is possible that the two key version is just as strong. Last I looked, this was an open question in the research literature.

RFC 2451 defines triple DES for IPSEC as the three-key variant. The two-key variant should not be used and is not implemented directly in Linux FreeS/WAN. It cannot be used in automatically keyed mode without major fiddles in the source code. For manually keyed connections, you could make Linux FreeS/WAN talk to a two-key implementation by setting two keys the same in /etc/ipsec.conf.

Virtual Interface

A Linux feature which allows one physical network interface to have two or more IP addresses. See the *Linux Network Administrator's Guide* in book form or on the web for details.

Virtual Private Network

see VPN

VPN

Virtual Private Network, a network which can safely be used as if it were private, even though some of its communication uses insecure connections. All traffic on those connections is encrypted.

IPSEC is not the only technique available for building VPNs, but it is the only method defined by RFCs and supported by many vendors. VPNs are by no means the only thing you can do with IPSEC, but they may be the most important application for many users.

VPNC
: Virtual Private Network Consortium, an association of vendors of VPN products.

Wassenaar Arrangement
: An international agreement restricting export of munitions and other tools of war. Unfortunately, cryptographic software is also restricted under the current version of the agreement.

Web of Trust
: PGP's method of certifying keys. Any user can sign a key; you decide which signatures or combinations of signatures to accept as certification. This contrasts with the hierarchy of CAs (Certification Authorities) used in many PKIs (Public Key Infrastructures).

See Global Trust Register for an interesting addition to the web of trust.

X.509
: A standard from the ITU (International Telecommunication Union), for hierarchical directories with authentication services, used in many PKI implementations.

Use of X.509 services, via the LDAP protocol, for certification of keys is allowed but not required by the IPSEC RFCs. It is not yet implemented in Linux FreeS/WAN.

Xedia
: A vendor of router and Internet access products. Their QVPN products interoperate with Linux FreeS/WAN; see our compatibility document.

Click below to go to:

- Document index file
- Table of Contents
- Beginning of this file
- FreeS/WAN home page

OIPE JC98 JUN 1 3 2002 PATENT & TRADEMARK OFFICE

#11/D
mm
PATENT 6-26-02

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

RECEIVED
JUN 2 4 2002
Technology Center 2100

In Re Application Of	
Edmund Colby MUNGER *et al.*	
Serial No.:	09/504,783
Filed:	February 15, 2000
For:	IMPROVEMENTS TO AN AGILE NETWORK PROTOCOL FOR SECURE COMMUINCATIONS WITH ASSURED SYSTEM AVAILABILITY

Group Art Unit: 2153

Examiner: K. Lim

Atty. Dkt. No. 00479.85672

AMENDMENT AND RESPONSE UNDER 37 C.F.R. § 1.111

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

In response to the Office Action mailed March 13, 2002, Applicants respectfully request the application be amended as follows. No fee is believed to be due with this Request. However, if a fee is due the Office is authorized to charge any required fees for consideration of this paper to our Deposit Account No. 19-0733.

IN THE CLAIMS

Please cancel claims 72-81.

Remarks

Applicants are in receipt of the Office Action mailed March 13, 2002, indicating that claims 28-39 and 67-81 are pending, claims 72-81 are withdrawn from consideration, claims 28-37 and 67-

69 stand rejected, and claims 38, 39, 70 and 71 are objected to. Applicants thank the Examiner for the indication of allowable subject matter in claims 38, 39, 70, and 71.

Submitted concurrently herewith are formal drawings in substitution for the informal drawings submitted with the application as filed. Applicants respectfully request that the official draftsman reviews the formal drawings at his earliest convenience.

Second Preliminary Amendment and IDS

A Second Preliminary Amendment adding claims 82-91 was submitted on February 22, 2002, but this amendment was not reflected in the Office Action mailed on March 13, 2002. Applicants respectfully request that the Second Preliminary Amendment be entered as of the date of its receipt by the Office, and that the claims submitted in the Second Preliminary Amendment be considered simultaneously with the requested reconsideration of the pending claims.

A Supplemental Information Disclosure Statement was also submitted February 22, 2002, but was not reflected in the Office Action mailed on March 13, 2002. Applicants respectfully request that the references cited in the Supplemental Information Disclosure Statement be considered and acknowledged at the Examiner's earliest convenience.

On the Merits

The Office Action restricted newly added claims 72-81 (group IV) as being drawn to an independent or distinct invention from the originally claimed invention in claims 28-39 and 67-71 (group II), and constructively elected group II for prosecution on the merits. By the present amendment, Applicants cancel claims 72-81.

The Office Action rejected claims 28-37 and 67-69 under 35 U.S.C. § 103(a) as being unpatentable over *Boden et al.* (U.S. Pat. No. 6,330,562, hereinafter "Boden") in view of *Risley et al.* (U.S. Pat. No. 6,332,158, hereinafter "Risley"). Applicants respectfully traverse this rejection based on the following arguments.

In order to reject a claim as obvious under § 103(a), three criteria must exist: 1) there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combined reference teachings; 2) there must be a reasonable expectation of success; and 3) the prior art reference(s) must teach or suggest all the claim limitations. See MPEP § 706.02 (j); *In re Vaeck*, 947 F.2d 488 (Fed. Cir. 1991).

First, Applicants submit that there is no motivation or suggestion to combine the Boden and Risley references. Boden discloses a data model for abstracting customer-defined VPN security policy information (Boden, Abstract). The system in Boden addresses the need to enable connection filter rules to be generated and loaded dynamically at negotiation time, due to remote initiating hosts having *dynamically assigned IP addresses*. Boden, col. 2, lines 38-41 (emphasis added). As cited in the Office Action, Boden allows for "dynamically establishing VPN connections with different security policies and other attributes, based solely on an unfixed IP address (e.g. [sic] *a user ID*)...." Boden, col. 3, lines 14-16 (emphasis added). Boden does not disclose establishing a VPN based on a DNS request for an IP address.

Risley discloses a DNS lookup system that allows intelligent correction of domain name searches by providing alternative suggestions of possible intended domain names when a DNS lookup was unsuccessful. Risley, Abstract. That is, when a user submits a domain name query, if the domain name exists, the domain name server (DNS) provides the corresponding machine address back to the user, as is known in the art. However, if the domain name does not exist, the Risley domain name server returns a machine address for a machine that will help the user identify the desired domain name. Subsequently, the machine to which the user has been redirected suggests possible intended domain names based on heuristics such as common misspellings, phonetic errors, and the like. Risley, Abstract. Risley does not teach or suggest establishing a VPN based on a DNS request, nor establishing any sort of secure communications channel over a network.

The Office Action states that establishing a secure connection between computers with the use of VPN would have been a desired feature in the art as suggested by Boden at col. 1, lines 41-55. However, Boden at col. 1, lines 41-55, discusses a general need for computer security, not a specific suggestion to incorporate the VPN techniques disclosed in Boden, or any other security technique, with a DNS lookup assistant as disclosed by Risley. In addition, there are many ways in which to create a VPN, and Boden at best only discloses a single specific security solution that may be used to establish a VPN. Boden does not include any suggestion or motivation to alter a DNS request scheme to create a VPN (in fact, there is only one instance of the acronym DNS in the entire Boden specification, col. 10, line 3, and no instances of the phrase "domain name service"). Indeed, Boden specifically states that "no verification is made via DNS or similar that [the mapping of ID to IP address] is correct." *Id.*

The Office Action also states that "the system that made it easier to remember, access, and convey the location information in order to access information would have been also a desired feature in the art as suggested by [Risley col. 1, lines 46-52]." However, Risley at col. 1, lines 46-52, discusses the general notion that users prefer using domain names (e.g., coolsite.com) rather than IP addresses (e.g., 199.227.249.232) when remembering, accessing, and conveying information. Risley does not provide a specific suggestion that its DNS service would benefit from the use of a VPN (or any other type of security). Risley only discloses that users prefer to use domain names over IP addresses when remembering, accessing, and conveying information, and provides a system for helping a user identify an intended domain name.

The Office Action concludes that it would have been obvious to combine the references in order to have "an easier to use and secure network connection because the teachings of these two references are complemented each other for easier to use and for securing network connection in a computer network." While two patents may ostensibly complement each other, this does not provide the necessary suggestion to combine the two references. In light of the fact that neither reference includes a specific suggestion to combine the references, the mere fact that two references are complementary does not provide the required suggestion or motivation. Risley does not teach or suggest establishing a VPN using its domain name resolution technique, nor does Boden teach using domain name resolution to establish a VPN.

To allow the combination of Boden and Risley would allow the hindsight combination of almost any two references as long as they had something in common, e.g., they both relate to the Internet. The Federal Circuit has repeatedly stated that the limitations of a claim in a pending application cannot be used as a blueprint to piece together prior art in hindsight, *In re Dembiczak*, 50 U.S.P.Q.2d 1614 (Fed. Cir. 1999), and that the Patent Office should *rigorously* apply the requirement that a teaching or motivation to combine prior art references needs to be provided. *Id.* (emphasis added). Thus, Applicants respectfully submit that that there is no motivation or suggestion to combine Risley, which discloses a modified DNS lookup system, with Boden, which discloses a specific VPN technique.

Second, even if the Boden and Risley references were combined, the combination would not teach or suggest all the limitations of any pending claim. The Office Action uses claim 37 as an exemplary claim, which requires:

> a DNS proxy server that receives a request from the client computer to look up an IP address for a domain name, wherein the DNS proxy server returns the IP address for the requested domain name if it is determined that access to a non-secure web site has been requested, and wherein the DNS proxy server generates a request to create the VPN between the client computer and the secure target computer if it is determined that access to a secure web site has been requested; and
>
> a gatekeeper computer that allocates resources for the VPN between the client computer and the secure web computer in response to the request by the DNS proxy server.

At a minimum, neither Boden nor Risley discloses a DNS proxy server that "generates a request to create the VPN between the client computer and the secure target computer if it is determined that access to a secure web site has been requested..." Neither Risley nor Boden teach or suggest triggering the creation of a VPN in response to a DNS request. Instead, Risley discloses a modified DNS lookup, whereby when a DNS request is received that is unsuccessful, Risley redirects the requestor to a domain name resolver to assist the user with locating an intended domain name. Risley does not disclose generating a request to create a VPN, as is required by claim 37, nor does Risley determine whether access to a secure web site has been requested. Likewise, Boden does not disclose these limitation, as is admitted in the Office Action at page 5, para. 11.

In addition, the Office Action does not indicate that either Boden or Risley includes a gatekeeper computer as is required by claim 37.

Based at least on the above arguments, Applicants respectfully traverse the rejection of claim 37 and its dependent claims.

The Office Action also rejected claims 28-36 and 67-69 for the same reasons set forth with respect to claim 37 because the claims are similar in scope. Applicants submit that each claim presents an individually patentable scope, and that these claims are allowable for at least the same reasons as claim 37.

In addition, with respect to claim 31, none of the cited references teach or suggest, upon determining that a client computer is not authorized to establish a VPN with a secure web site, returning an error from the DNS request.

With respect to claim 32, none of the cited references teach or suggest, upon determining that a client computer is not authorized to resolve addresses of non-secure target computers, returning an error from the DNS request.

With respect to claim 33, none of the cited references teach or suggest establishing the VPN by creating an IP address hopping scheme between the client computer and the target computer. (see, e.g., allowable subject matter in claim 38).

With respect to claim 34, none of the cited references teach or suggest using a gatekeeper computer that allocates VPN resources for communicating between the client computer and the target computer.

With respect to claim 35, none of the cited references teach or suggest that step (2) is performed in a DNS proxy server that passes through the request to a DNS server if it is determined in step (3) that access is not being requested to a secure target web site.

With respect to claim 68, none of the cited references teach or suggest communicating according to a scheme by which at least one field in a series of data packets is periodically changed according to a known sequence.

With respect to claim 69, none of the cited references teach or suggest comparing an Internet Protocol (IP) address in a header of each data packet to a table of valid IP addresses maintained in a table in the second computer.

Based on the aforementioned Applicants respectfully submit that all pending claims are in condition for allowance, and Applicants request that the subject application be reconsidered and passed to issue at the Examiner's earliest possible convenience.

If the Examiner has any questions or wishes to discuss this amendment, the Examiner is invited to telephone the undersigned representative at the number set forth below.

Respectfully submitted,

BANNER & WITCOFF, LTD.

Date: June 13, 2002 By: [illegible]

Bradley C. Wright **Reg. No. 49,024**
Registration No. 38,061
1001 G Street N.W., 11th Floor
Washington, D.C. 20001
(202) 508-9100

JUN 1 3 2002 JC98 PATENT & TRADEMARK OFFICE

2153

Please type a plus sign (+) inside this box ⟶ [+]

PTO/SB/21 (08-00)
Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/504,783
Filing Date	February 15, 2000
First Named Inventor	Edmund Colby Munger
Group Art Unit	2153
Examiner Name	K. Lim
Attorney Docket Number	000479.85672
Total Number of Pages in This Submission	

RECEIVED JUN 2 4 2002 Technology Center 2100

ENCLOSURES *(check all that apply)*

- ☐ Fee Transmittal Form
  - ☐ Fee Attached
- ☒ Amendment / Response
  - ☐ After Final
  - ☐ Affidavits/declaration(s)
- ☐ Extension of Time Request
- ☐ Express Abandonment Request
- ☐ Information Disclosure Statement
- ☐ Certified Copy of Priority Document(s)
- ☐ Response to Missing Parts/ Incomplete Application
  - ☐ Response to Missing Parts under 37 CFR 1.52 or 1.53
- ☐ Assignment Papers *(for an Application)*
- ☒ Drawing(s)
- ☐ Licensing-related Papers
- ☐ Petition
- ☐ Petition to Convert to a Provisional Application
- ☐ Power of Attorney, Revocation Change of Correspondence Address
- ☐ Terminal Disclaimer
- ☐ Request for Refund
- ☐ CD, Number of CD(s) _____
- ☐ After Allowance Communication to Group
- ☐ Appeal Communication to Board of Appeals and Interferences
- ☐ Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*
- ☐ Proprietary Information
- ☐ Status Letter
- ☒ Other Enclosure(s) *(please identify below)*:
  Submission of Formal Drawings to Official Draftsman

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm or Individual name	Bradley C. Wright, Reg. No. 38,061
Signature	[illegible] **Reg. No. 49,024**
Date	June 13, 2002

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231 on this date:

Typed or printed name			
Signature		Date	

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be send to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

JUN 1 '02
PATENT & TRADEMARK OFFICE

#18

#12

RECEIVED
JUN 24 2002
Technology Center 2100

PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	Group Art Unit: 2153
Edmond Colby Munger et al.	Examiner: K. Lim
Serial No. 09/504,783	Attorney Docket No. 00479.85672
Filed: February 15, 2000	

For: IMPROVEMENTS TO AN AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS WITH ASSURED SYSTEM AVAILABILITY

SUBMISSION OF FORMAL DRAWINGS TO OFFICIAL DRAFTSMAN

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Please substitute the attached 35 sheets of formal drawings depicting Figures 1-32 for the informal drawings filed with the patent application on February 15, 2000, in this matter. Applicant respectfully requests the Official Draftsman to review these drawings and advise the undersigned of any objections thereto.

It is believed that no fee is required. However, if a fee is required, please charge our Deposit Account No. 19-0733.

Respectfully submitted,

Date: June 13, 2002

By: [illegible]
for Bradley C. Wright
Registration No. 38,061 **Reg. No. 49,024**

BANNER & WITCOFF, LTD
1001 G Street, N.W.
Eleventh Floor
Washington, D.C. 20001
(202) 508-9100
RAD/mmd

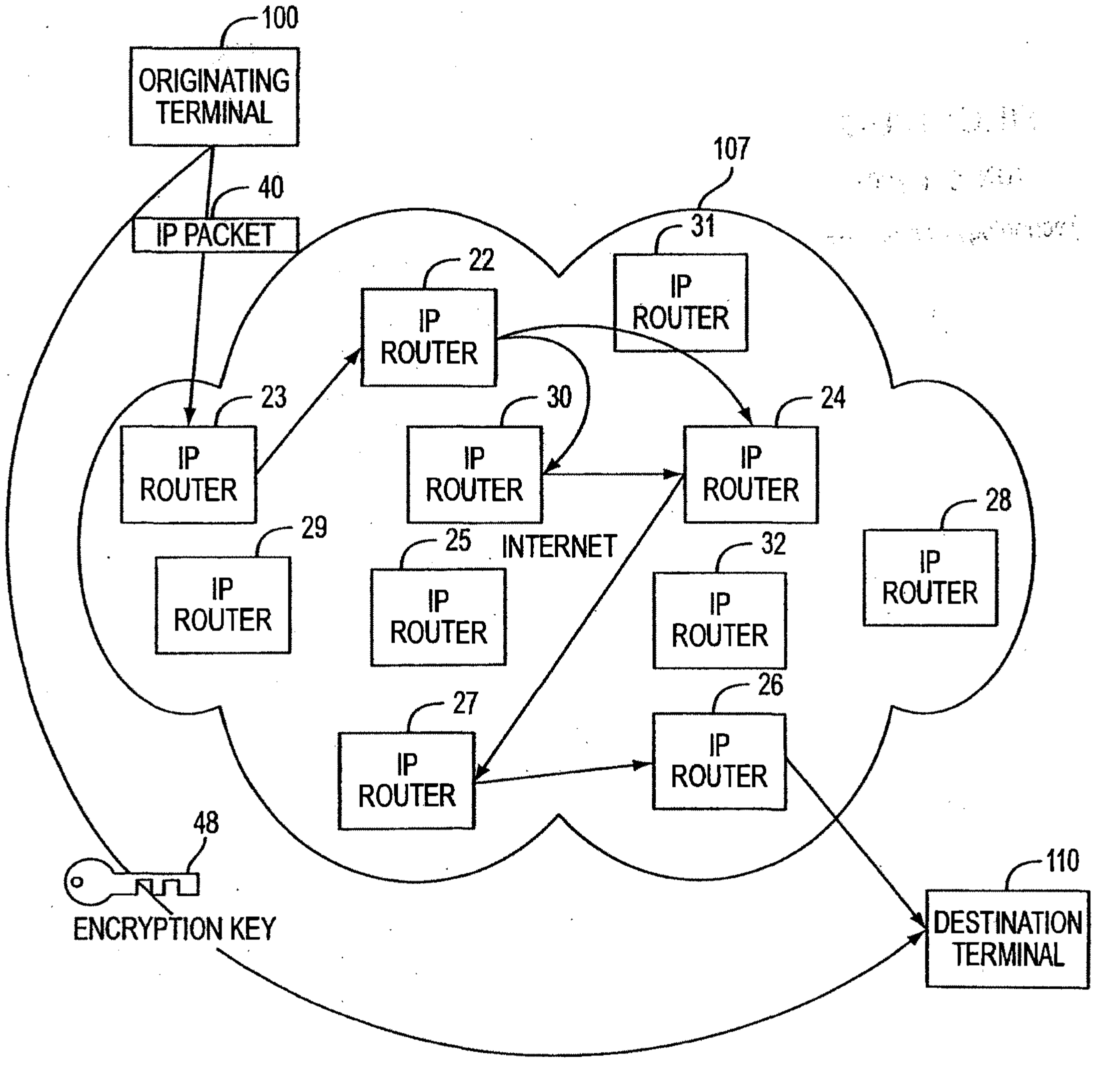

100
ORIGINATING TERMINAL
40
IP PACKET
107
31
IP ROUTER
22
IP ROUTER
23
IP ROUTER
30
IP ROUTER
24
IP ROUTER
28
IP ROUTER
29
IP ROUTER
25
IP ROUTER
INTERNET
32
IP ROUTER
27
IP ROUTER
26
IP ROUTER
48
ENCRYPTION KEY
110
DESTINATION TERMINAL

FIG. 1

OIPE JC98
JUN 1 3 2002
PATENT & TRADEMARK OFFICE

RECEIVED
JUN 2 4 2002
Technology Center 2100

VNET00221431

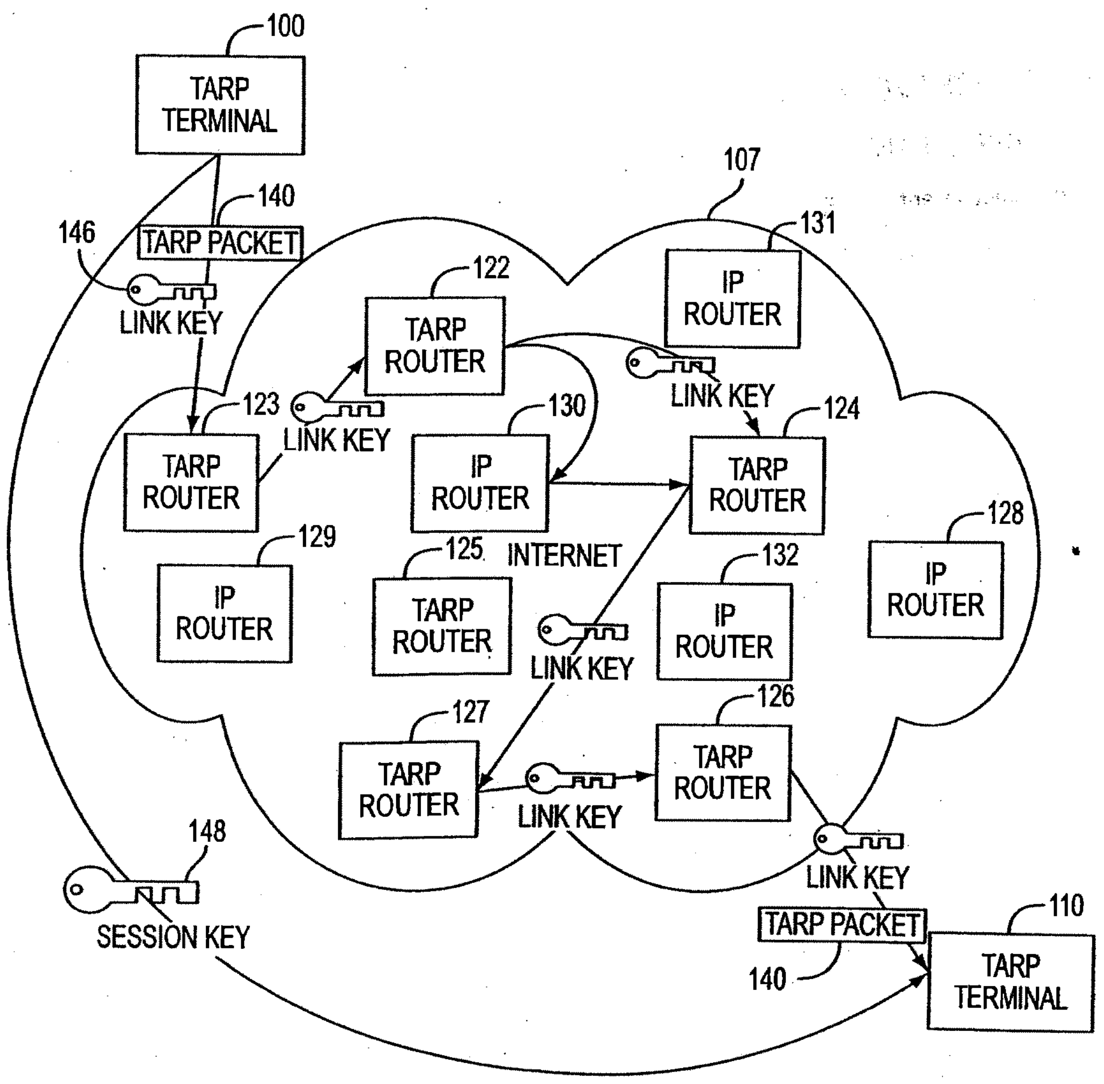
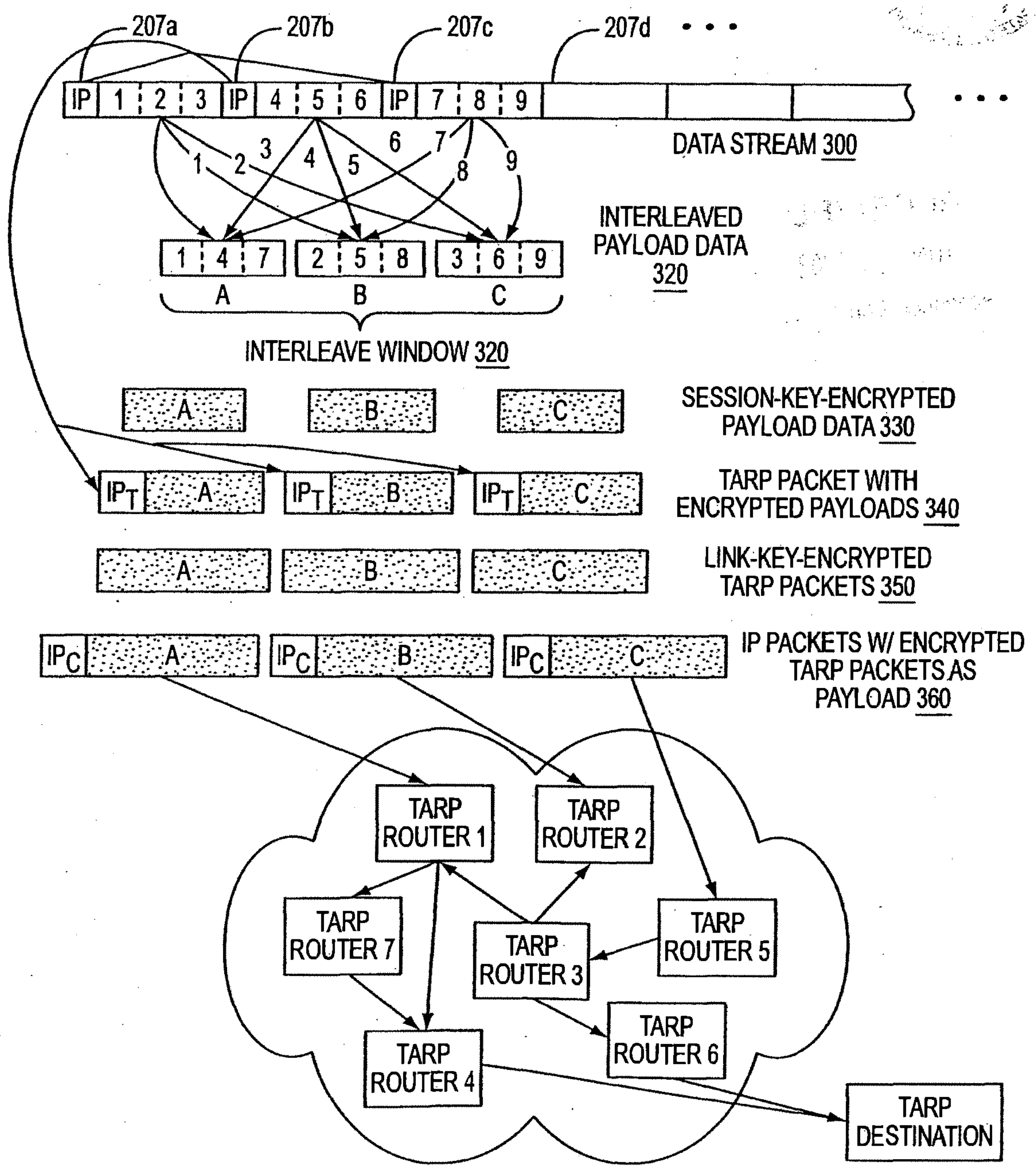

100
TARP TERMINAL
140
TARP PACKET
146
LINK KEY
107
131
IP ROUTER
122
TARP ROUTER
LINK KEY
123
TARP ROUTER
LINK KEY
130
IP ROUTER
124
TARP ROUTER
128
IP ROUTER
129
IP ROUTER
125
TARP ROUTER
INTERNET
132
IP ROUTER
LINK KEY
127
TARP ROUTER
126
TARP ROUTER
LINK KEY
LINK KEY
148
SESSION KEY
TARP PACKET
140
110
TARP TERMINAL

FIG. 2

OIPE JC80
JUN 1 3 2002
PATENT & TRADEMARK OFFICE

RECEIVED
JUN 2 4 2002
Technology Center 2100

VNET00221433



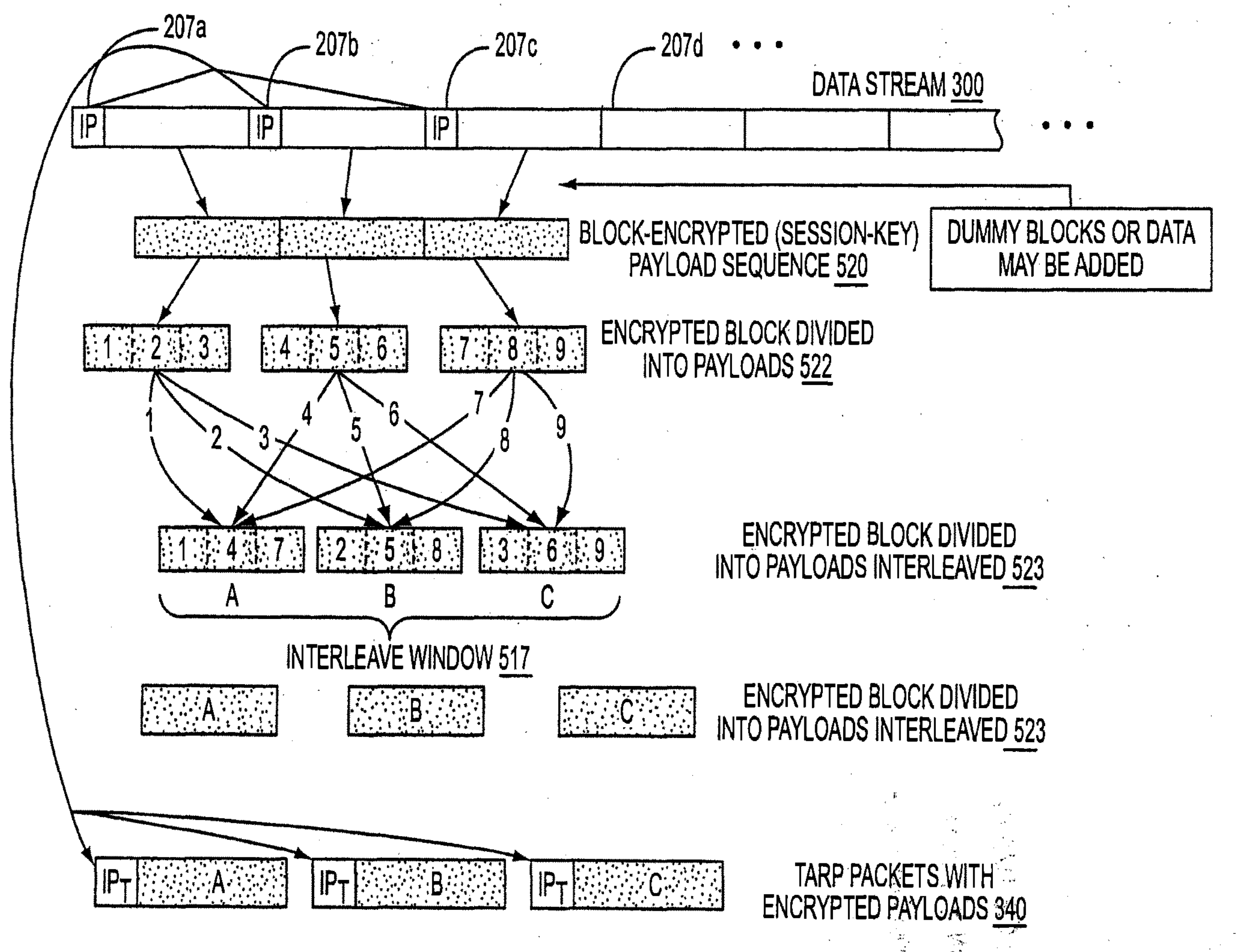
207a
207b
207c
207d
IP 1 2 3 IP 4 5 6 IP 7 8 9
1 2 3 4 5 6 7 8 9
DATA STREAM 300
1 4 7
2 5 8
3 6 9
A B C
INTERLEAVED PAYLOAD DATA 320
INTERLEAVE WINDOW 320
SESSION-KEY-ENCRYPTED PAYLOAD DATA 330
IPT A
IPT B
IPT C
TARP PACKET WITH ENCRYPTED PAYLOADS 340
LINK-KEY-ENCRYPTED TARP PACKETS 350
IPC A
IPC B
IPC C
IP PACKETS W/ ENCRYPTED TARP PACKETS AS PAYLOAD 360
TARP ROUTER 1
TARP ROUTER 2
TARP ROUTER 7
TARP ROUTER 3
TARP ROUTER 5
TARP ROUTER 6
TARP ROUTER 4
TARP DESTINATION

FIG. 3A

OIPE JC98
JUN 1 3 2002
PATENT & TRADEMARK OFFICE

RECEIVED
JUN 2 4 2002
Technology Center 2100

VNET00221435



207a
207b
207c
207d
DATA STREAM 300
IP
BLOCK-ENCRYPTED (SESSION-KEY) PAYLOAD SEQUENCE 520
DUMMY BLOCKS OR DATA MAY BE ADDED
ENCRYPTED BLOCK DIVIDED INTO PAYLOADS 522
1 2 3 4 5 6 7 8 9
ENCRYPTED BLOCK DIVIDED INTO PAYLOADS INTERLEAVED 523
1 4 7
2 5 8
3 6 9
A B C
INTERLEAVE WINDOW 517
ENCRYPTED BLOCK DIVIDED INTO PAYLOADS INTERLEAVED 523
IPT
TARP PACKETS WITH ENCRYPTED PAYLOADS 340

FIG. 3B

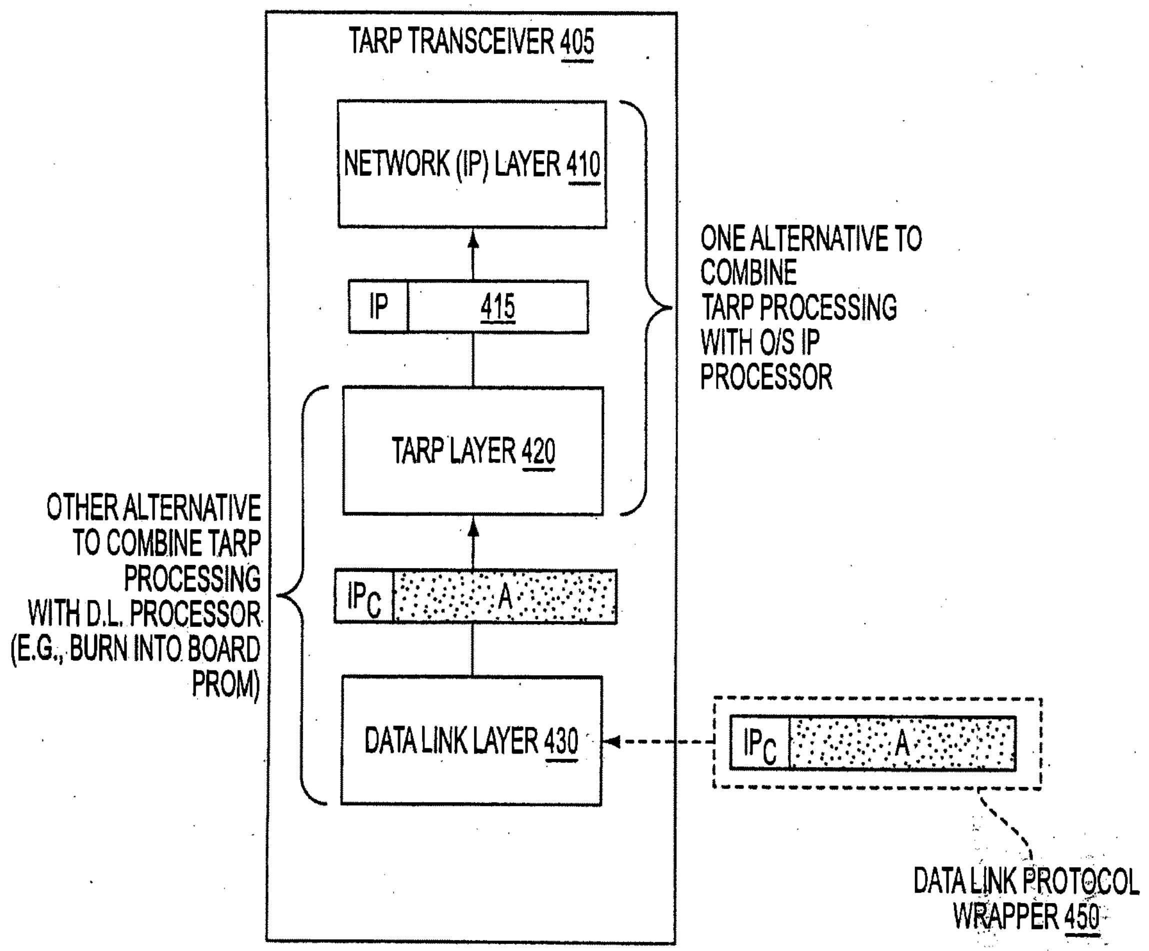
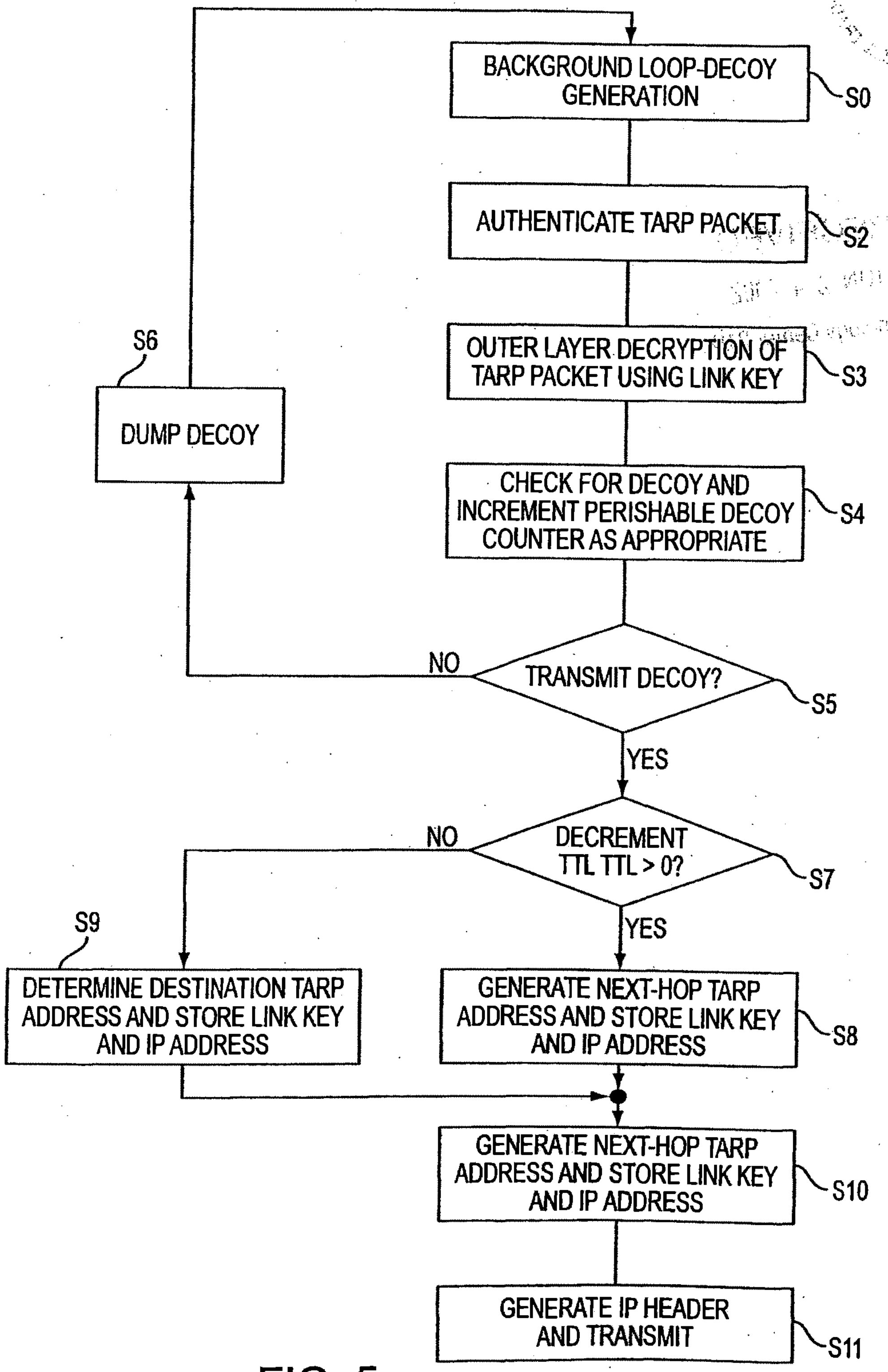

TARP TRANSCEIVER 405
NETWORK (IP) LAYER 410
IP
415
TARP LAYER 420
IPC
A
DATA LINK LAYER 430
ONE ALTERNATIVE TO COMBINE TARP PROCESSING WITH O/S IP PROCESSOR
OTHER ALTERNATIVE TO COMBINE TARP PROCESSING WITH D.L. PROCESSOR (E.G., BURN INTO BOARD PROM)
DATA LINK PROTOCOL WRAPPER 450

FIG. 4

OIPE JC98
JUN 1 3 2002
PATENT & TRADEMARK OFFICE

RECEIVED
JUN 2 4 2002
Technology Center 2100

VNET00221439



BACKGROUND LOOP-DECOY GENERATION
S0
AUTHENTICATE TARP PACKET
S2
OUTER LAYER DECRYPTION OF TARP PACKET USING LINK KEY
S3
CHECK FOR DECOY AND INCREMENT PERISHABLE DECOY COUNTER AS APPROPRIATE
S4
TRANSMIT DECOY?
S5
NO
YES
S6
DUMP DECOY
DECREMENT TTL TTL > 0?
S7
NO
YES
S9
DETERMINE DESTINATION TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
GENERATE NEXT-HOP TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
S8
GENERATE NEXT-HOP TARP ADDRESS AND STORE LINK KEY AND IP ADDRESS
S10
GENERATE IP HEADER AND TRANSMIT
S11

FIG. 5

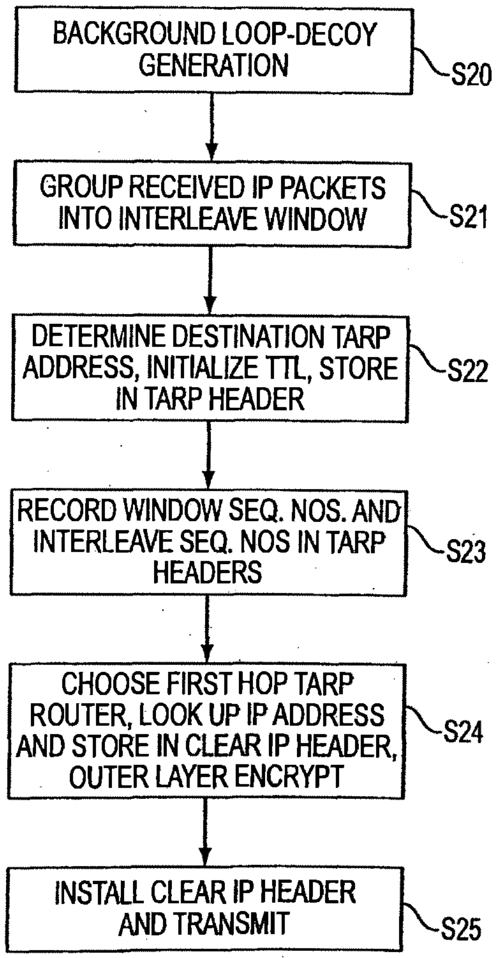

BACKGROUND LOOP-DECOY GENERATION
S20
GROUP RECEIVED IP PACKETS INTO INTERLEAVE WINDOW
S21
DETERMINE DESTINATION TARP ADDRESS, INITIALIZE TTL, STORE IN TARP HEADER
S22
RECORD WINDOW SEQ. NOS. AND INTERLEAVE SEQ. NOS IN TARP HEADERS
S23
CHOOSE FIRST HOP TARP ROUTER, LOOK UP IP ADDRESS AND STORE IN CLEAR IP HEADER, OUTER LAYER ENCRYPT
S24
INSTALL CLEAR IP HEADER AND TRANSMIT
S25

FIG. 6

OIPE JC98 JUN 1 3 2002 PATENT & TRADEMARK

RECEIVED
JUN 2 4 2002
Technology Center 2100

VNET00221443

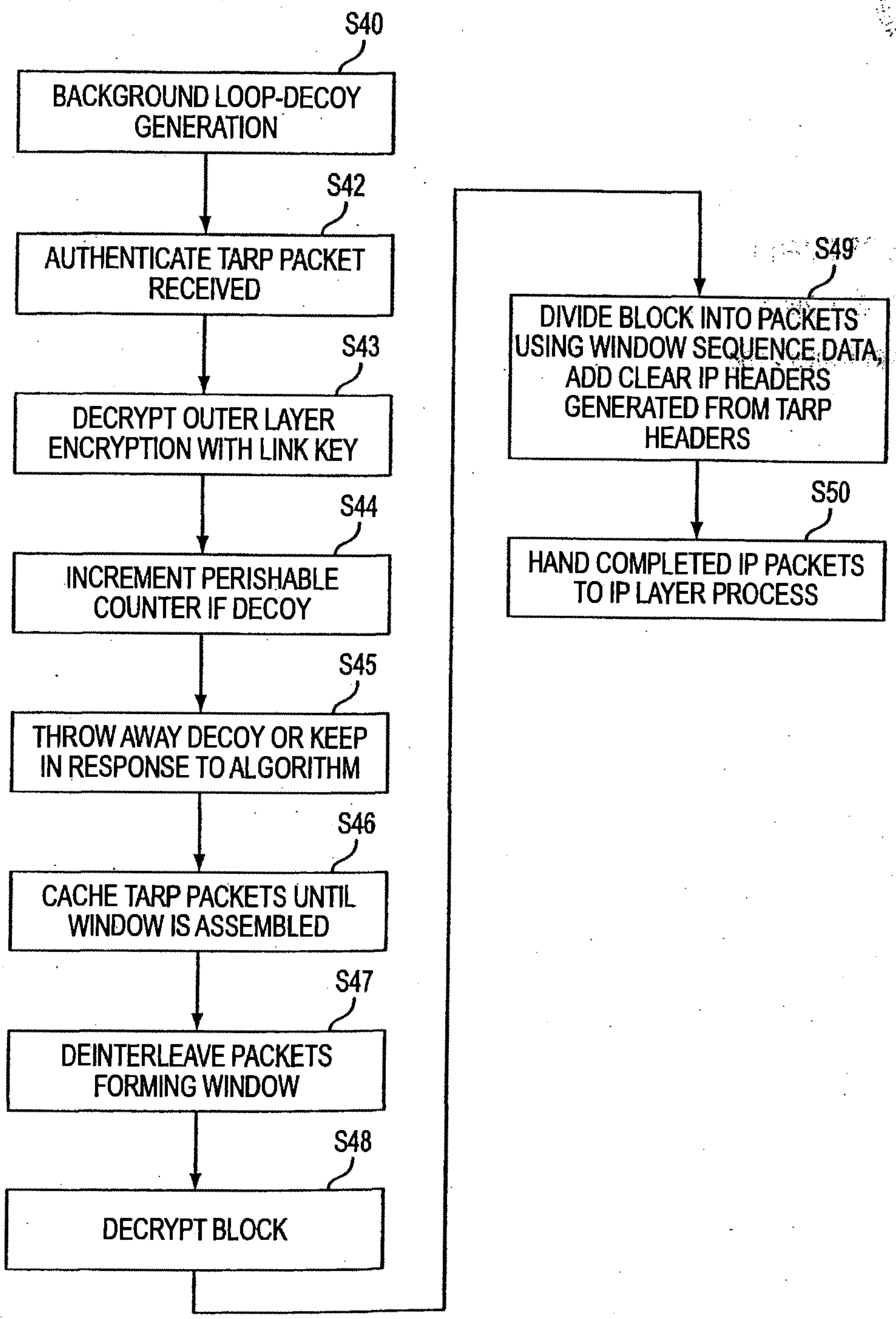

S40
BACKGROUND LOOP-DECOY GENERATION
S42
AUTHENTICATE TARP PACKET RECEIVED
S43
DECRYPT OUTER LAYER ENCRYPTION WITH LINK KEY
S44
INCREMENT PERISHABLE COUNTER IF DECOY
S45
THROW AWAY DECOY OR KEEP IN RESPONSE TO ALGORITHM
S46
CACHE TARP PACKETS UNTIL WINDOW IS ASSEMBLED
S47
DEINTERLEAVE PACKETS FORMING WINDOW
S48
DECRYPT BLOCK
S49
DIVIDE BLOCK INTO PACKETS USING WINDOW SEQUENCE DATA, ADD CLEAR IP HEADERS GENERATED FROM TARP HEADERS
S50
HAND COMPLETED IP PACKETS TO IP LAYER PROCESS

FIG. 7

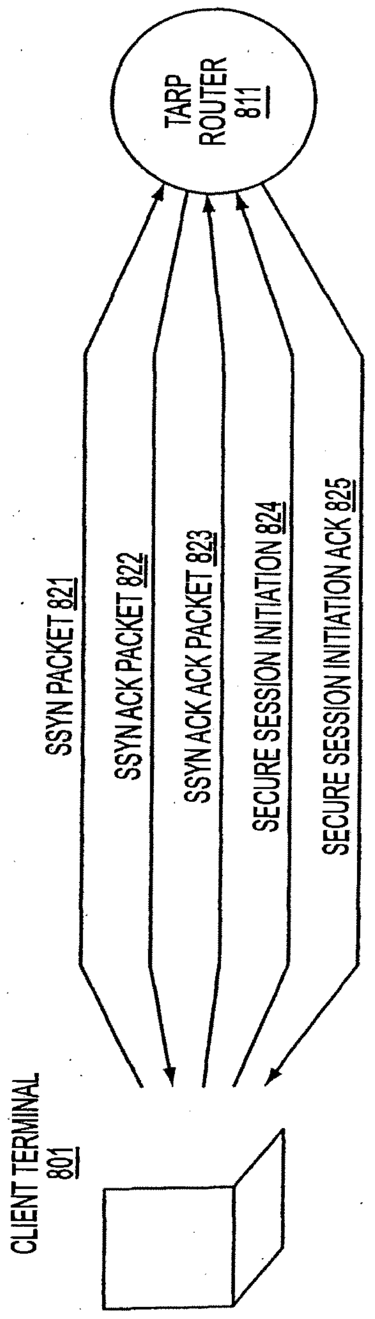
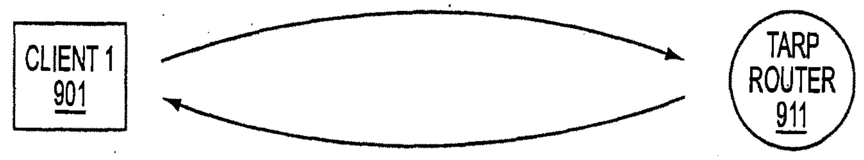

TARP ROUTER 811
SSYN PACKET 821
SSYN ACK PACKET 822
SSYN ACK ACK PACKET 823
SECURE SESSION INITIATION 824
SECURE SESSION INITIATION ACK 825
CLIENT TERMINAL 801

FIG. 8

OIPE JC98 JUN 1 3 2002 PATENT & TRADEMARK OFFICE

RECEIVED
JUN 2 4 2002
Technology Center 2100



TRANSMIT TABLE 921

131.218.204.98	,	131.218.204.65
131.218.204.221	,	131.218.204.97
131.218.204.139	,	131.218.204.186
131.218.204.12	,	131.218.204.55
⋮		⋮

RECEIVE TABLE 924

131.218.204.98	,	131.218.204.65
131.218.204.221	,	131.218.204.97
131.218.204.139	,	131.218.204.186
131.218.204.12	,	131.218.204.55
⋮		⋮

RECEIVE TABLE 922

131.218.204.161	,	131.218.204.89
131.218.204.66	,	131.218.204.212
131.218.204.201	,	131.218.204.127
131.218.204.119	,	131.218.204.49
⋮		⋮

TRANSMIT TABLE 923

131.218.204.161	,	131.218.204.89
131.218.204.66	,	131.218.204.212
131.218.204.201	,	131.218.204.127
131.218.204.119	,	131.218.204.49
⋮		⋮

FIG. 9

OIPE JC98
JUN 1 3 2002
PATENT & TRADEMARK OFFICE

RECEIVED
JUN 2 4 2002
Technology Center 2100

VNET00221449

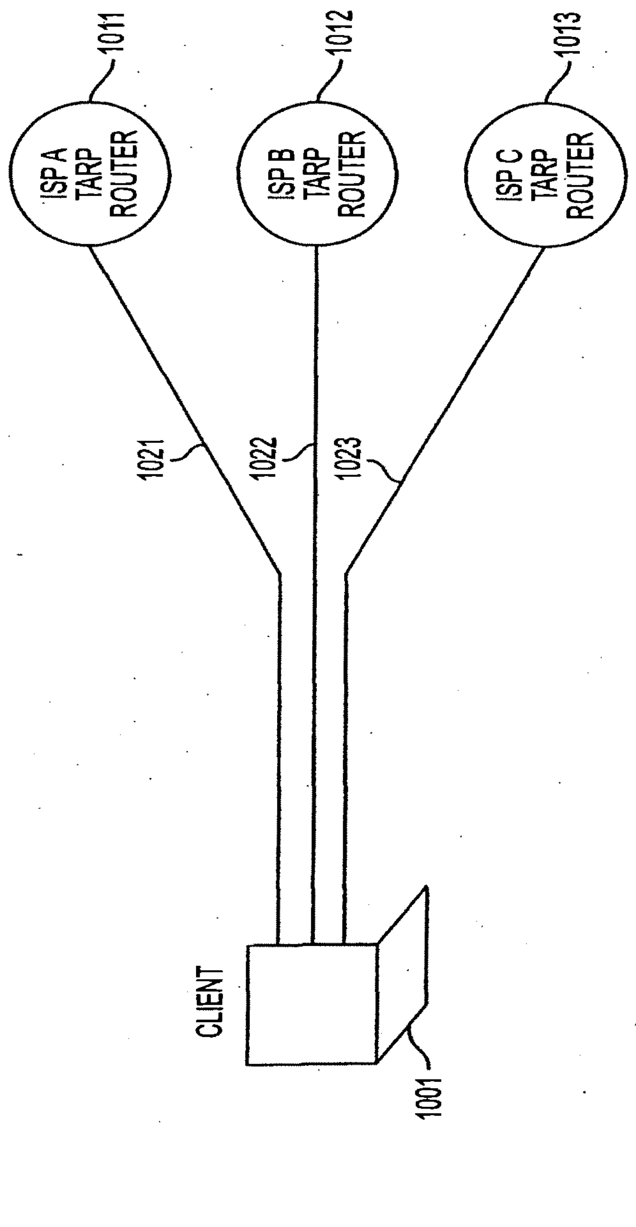

ISP A TARP ROUTER
1011
ISP B TARP ROUTER
1012
ISP C TARP ROUTER
1013
1021
1022
1023
CLIENT
1001

FIG. 10

OIPE JC98 JUN 1 3 2002 PATENT & TRADEMARK OFFICE

RECEIVED
JUN 2 4 2002
Technology Center 2100

VNET00221451

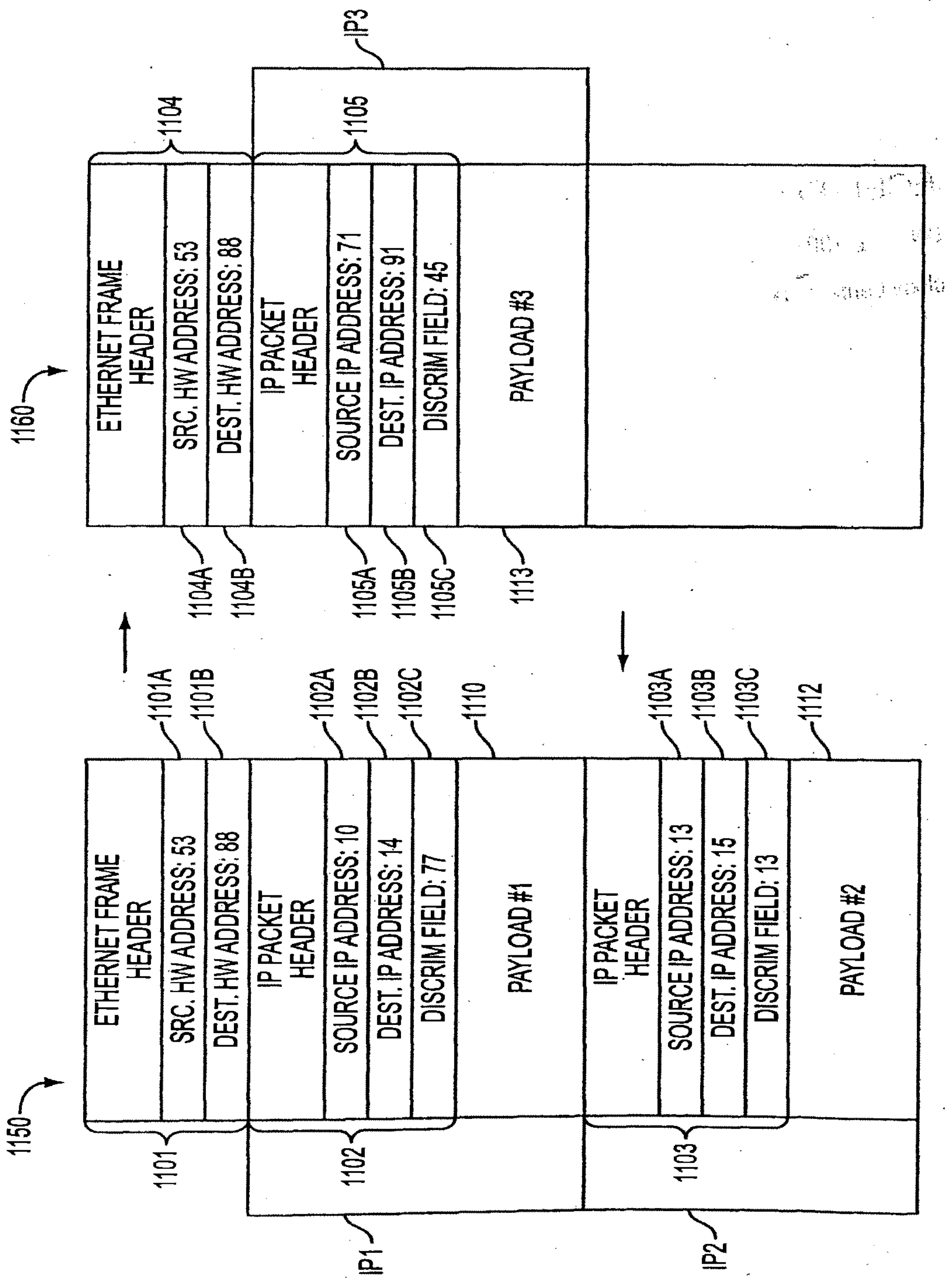
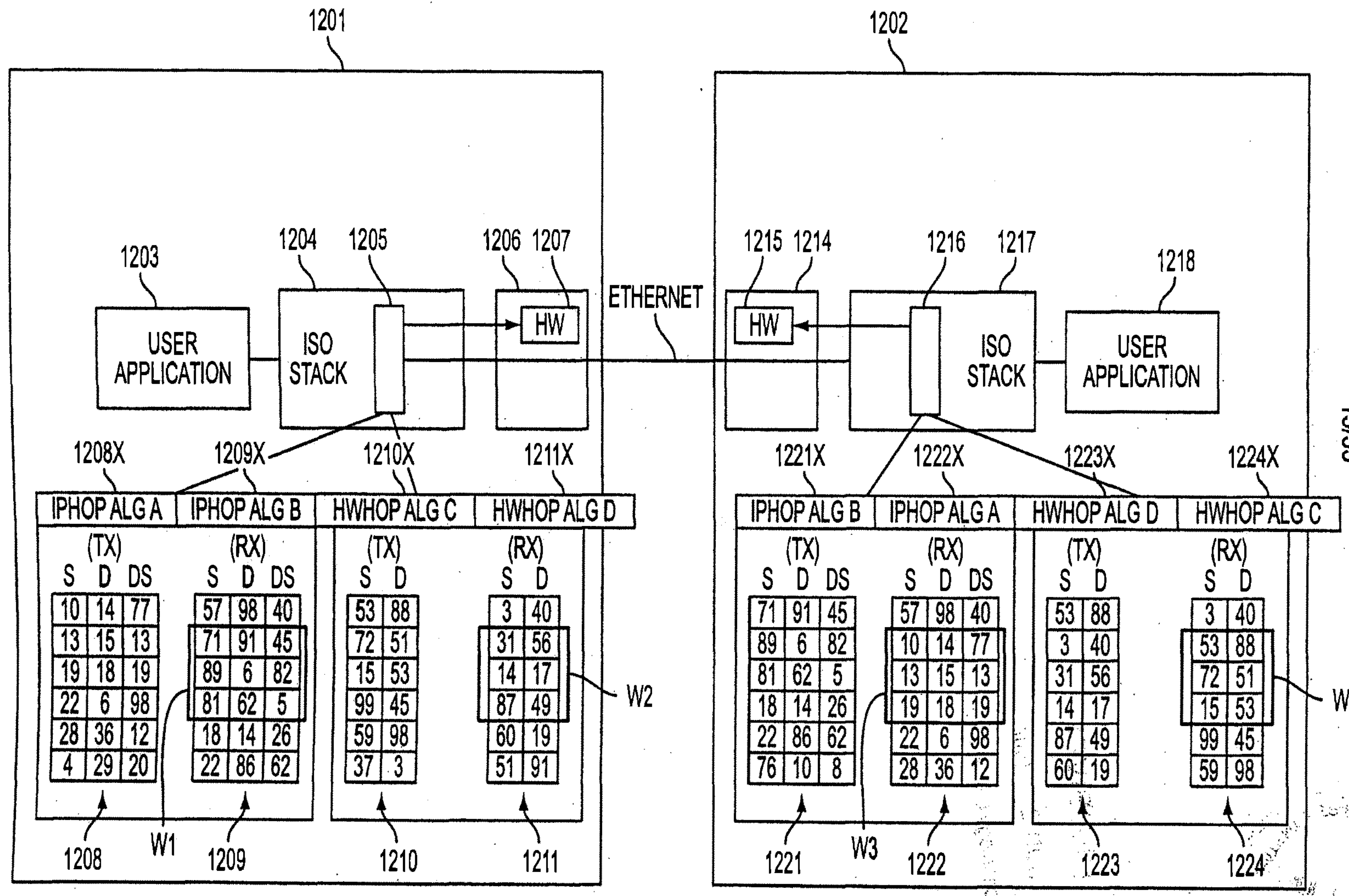

1150
1101
1102
IP1
1103
IP2
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
IP PACKET HEADER
SOURCE IP ADDRESS: 10
DEST. IP ADDRESS: 14
DISCRIM FIELD: 77
PAYLOAD #1
IP PACKET HEADER
SOURCE IP ADDRESS: 13
DEST. IP ADDRESS: 15
DISCRIM FIELD: 13
PAYLOAD #2
1101A
1101B
1102A
1102B
1102C
1110
1103A
1103B
1103C
1112
1160
1104
1105
IP3
ETHERNET FRAME HEADER
SRC. HW ADDRESS: 53
DEST. HW ADDRESS: 88
IP PACKET HEADER
SOURCE IP ADDRESS: 71
DEST. IP ADDRESS: 91
DISCRIM FIELD: 45
PAYLOAD #3
1104A
1104B
1105A
1105B
1105C
1113

FIG. 11

O I P E JC98
JUN 1 3 2002
PATENT & TRADEMARK OFFICE

RECEIVED
JUN 2 4 2002
Technology Center 2100

VNET00221453



1208 IPHOP ALG A (TX)

S	D	DS
10	14	77
13	15	13
19	18	19
22	6	98
28	36	12
4	29	20

1209 IPHOP ALG B (RX)

S	D	DS
57	98	40
71	91	45
89	6	82
81	62	5
18	14	26
22	86	62

1210 HWHOP ALG C (TX)

S	D
53	88
72	51
15	53
99	45
59	98
37	3

1211 HWHOP ALG D (RX)

S	D
3	40
31	56
14	17
87	49
60	19
51	91

1221 IPHOP ALG B (TX)

S	D	DS
71	91	45
89	6	82
81	62	5
18	14	26
22	86	62
76	10	8

1222 IPHOP ALG A (RX)

S	D	DS
57	98	40
10	14	77
13	15	13
19	18	19
22	6	98
28	36	12

1223 HWHOP ALG D (TX)

S	D
53	88
3	40
31	56
14	17
87	49
60	19

1224 HWHOP ALG C (RX)

S	D
3	40
53	88
72	51
15	53
99	45
59	98

FIG. 12A

MODE OR EMBODIMENT	HARDWARE ADDRESSES	IP ADDRESSES	DISCRIMINATOR FIELD VALUES
1. PROMISCUOUS	SAME FOR ALL NODES OR COMPLETELY RANDOM	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
2. PROMISCUOUS PER VPN	FIXED FOR EACH VPN	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC
3. HARDWARE HOPPING	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC	CAN BE VARIED IN SYNC

FIG. 12B

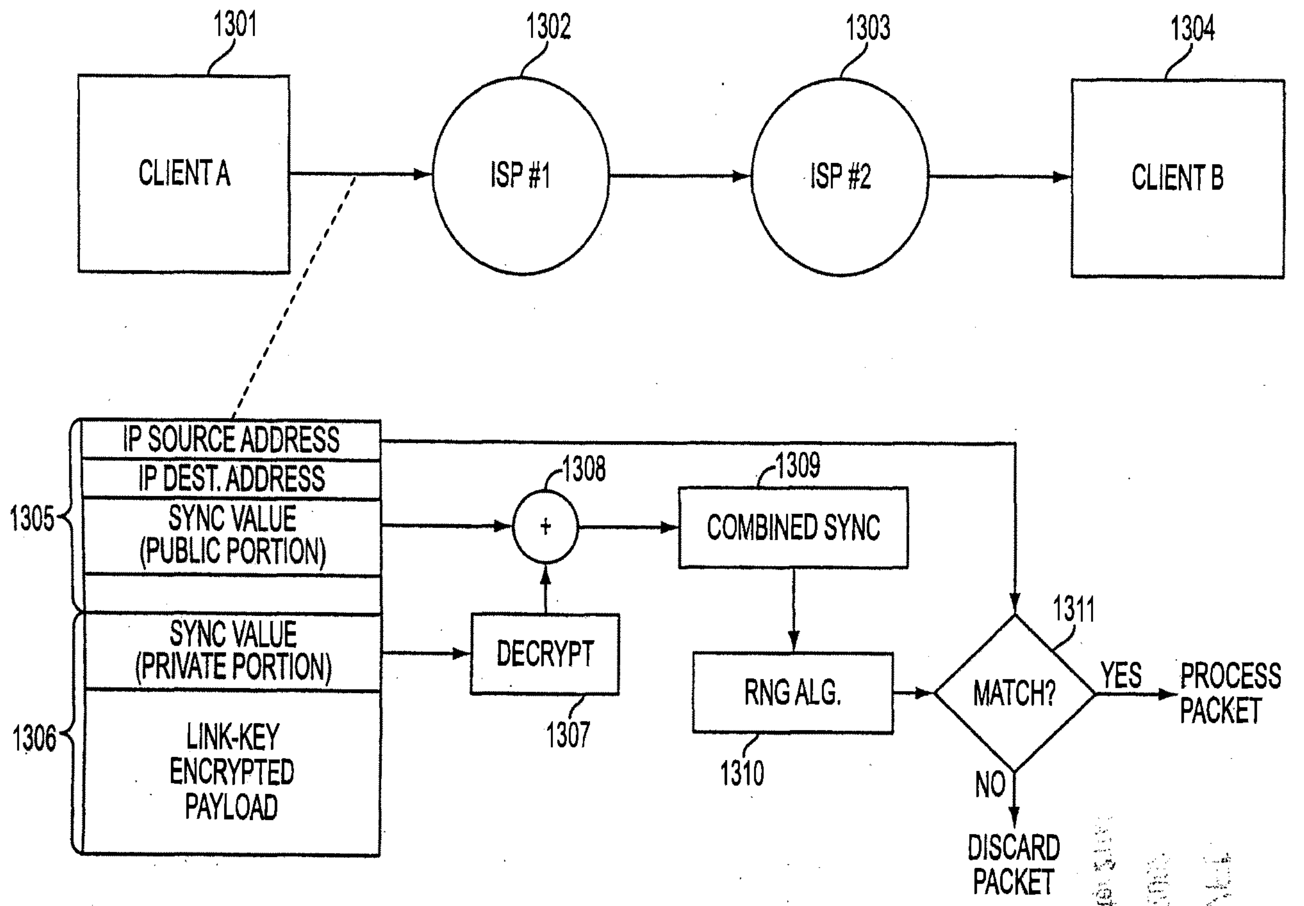

1301
CLIENT A
1302
ISP #1
1303
ISP #2
1304
CLIENT B
1305
IP SOURCE ADDRESS
IP DEST. ADDRESS
SYNC VALUE (PUBLIC PORTION)
1306
SYNC VALUE (PRIVATE PORTION)
LINK-KEY ENCRYPTED PAYLOAD
1308
+
1307
DECRYPT
1309
COMBINED SYNC
1310
RNG ALG.
1311
MATCH?
YES
PROCESS PACKET
NO
DISCARD PACKET

FIG. 13

OIPE JC98 JUN 1 3 2002 PATENT & TRADEMARK OFFICE

RECEIVED
JUN 2 4 2002
Technology Center 2100

VNET00221459

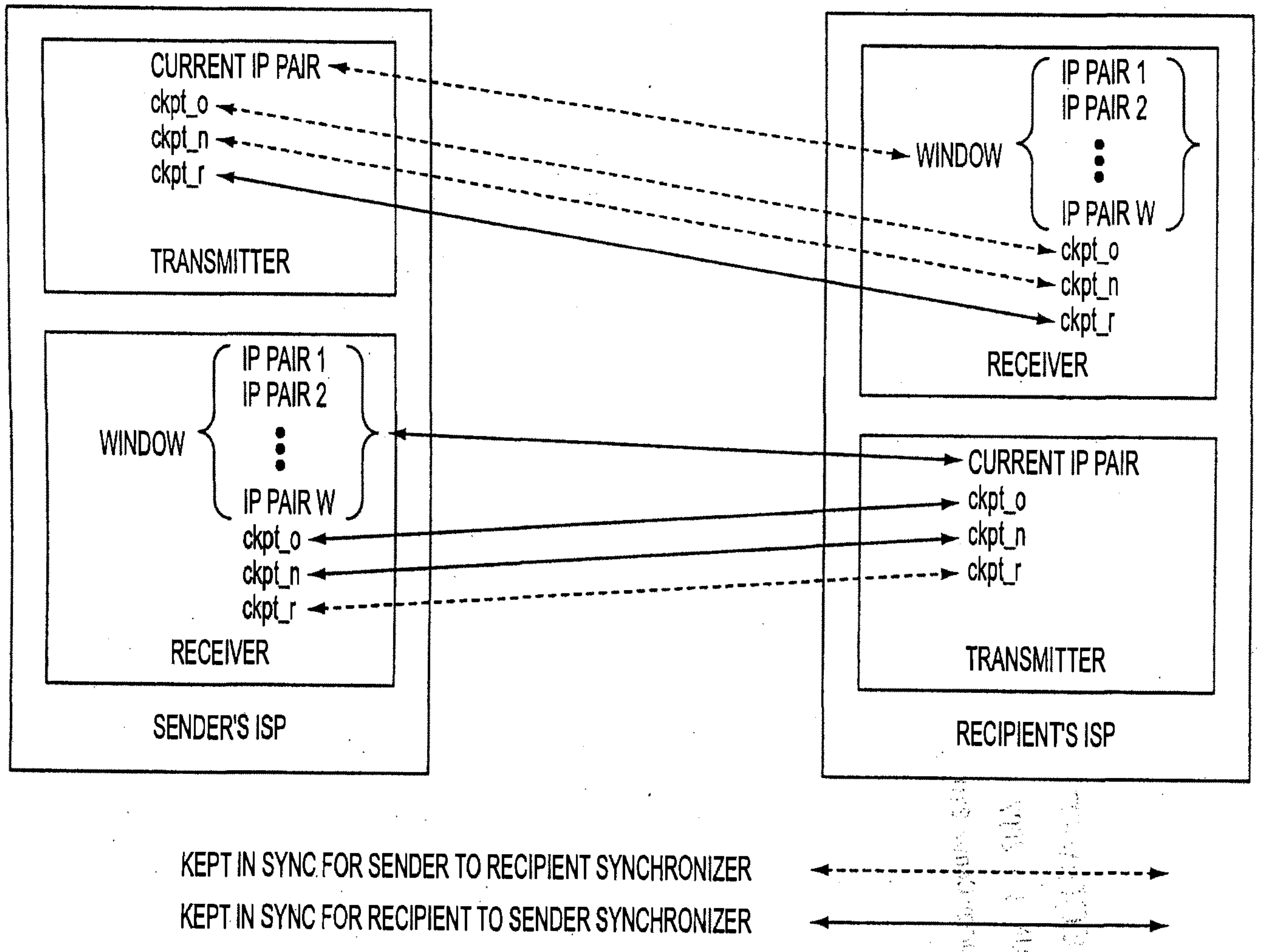
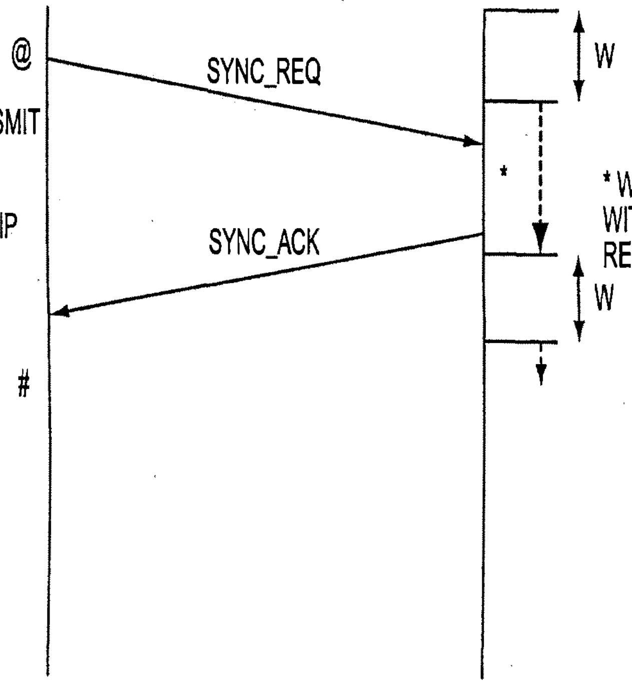

CURRENT IP PAIR
ckpt_o
ckpt_n
ckpt_r
TRANSMITTER
WINDOW
IP PAIR 1
IP PAIR 2
IP PAIR W
ckpt_o
ckpt_n
ckpt_r
RECEIVER
SENDER'S ISP
WINDOW
IP PAIR 1
IP PAIR 2
IP PAIR W
ckpt_o
ckpt_n
ckpt_r
RECEIVER
CURRENT IP PAIR
ckpt_o
ckpt_n
ckpt_r
TRANSMITTER
RECIPIENT'S ISP
KEPT IN SYNC FOR SENDER TO RECIPIENT SYNCHRONIZER
KEPT IN SYNC FOR RECIPIENT TO SENDER SYNCHRONIZER

FIG. 14



@ WHEN SYNCHRONIZATION BEGINS TRANSMIT (RETRANSMIT PERIODICALLY UNTIL ACKed) SYNC_REQ USING NEW TRANSMITTER CHECKPOINT IP PAIR ckpt_n AND GENERATE NEW RECEIVER RESPONSE CHECKPOINT ckpt_r

WHEN SYNC_ACK ARRIVES WITH INCOMING HEADER = ckpt_r: GENERATE NEW CHECKPOINT IP PAIR ckpt_n IN TRANSMITTER

* WHEN SYNC_REQ ARRIVES WITH INCOMING HEADER = RECEIVER'S ckpt_n:
- UPDATE WINDOW
- GENERATE NEW CHECKPOINT IP PAIR ckpt_n IN RECEIVER
- GENERATE NEW CHECKPOINT IP PAIR ckpt_r IN TRANSMITTER
- TRANSMIT SYNC_ACK USING NEW CHECKPOINT IP PAIR ckpt_r

FIG. 15

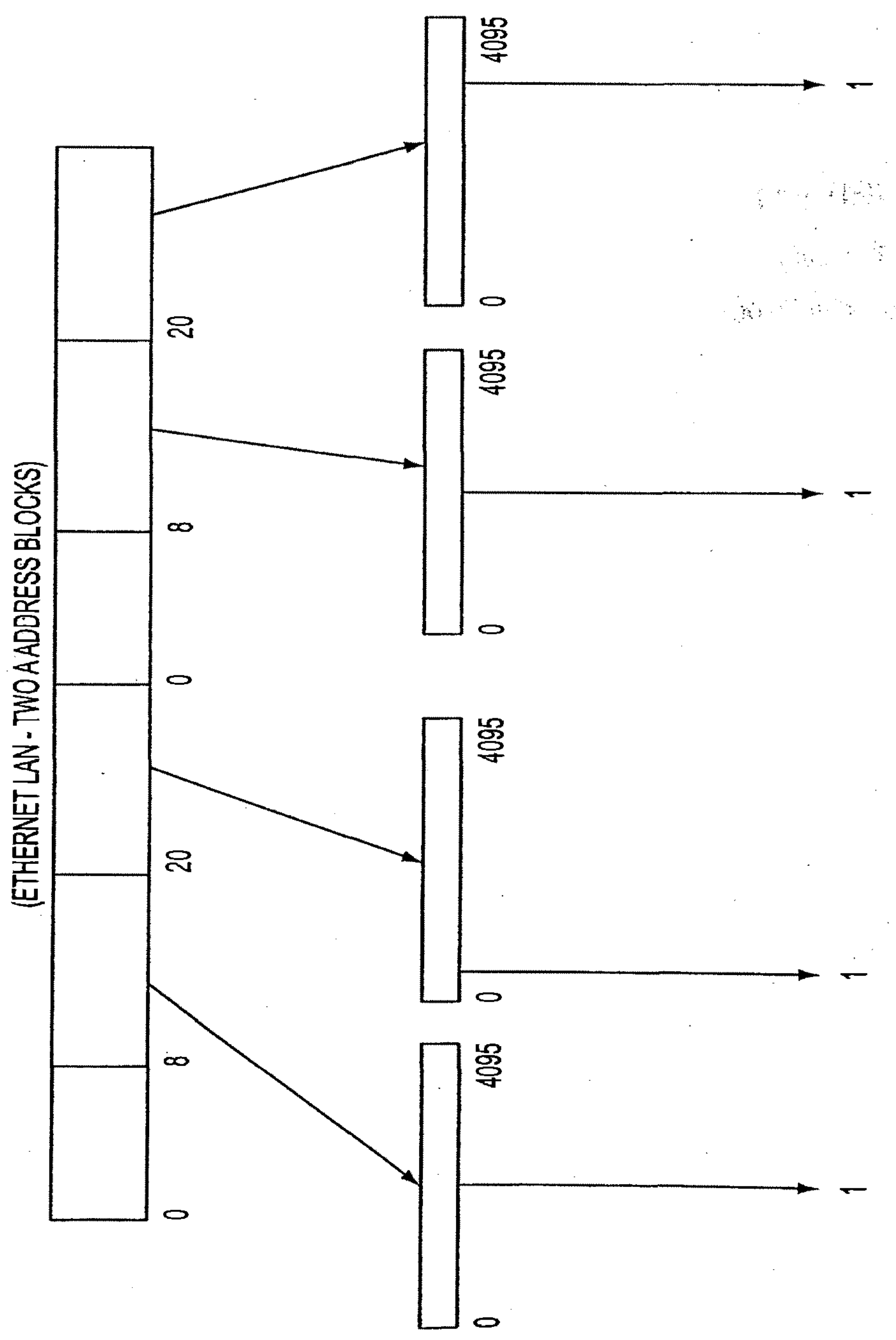

(ETHERNET LAN - TWO A ADDRESS BLOCKS)
0
8
20
0
8
20
0
4095
0
4095
0
4095
0
4095
1
1
1
1

FIG. 16

O I P E JC98
JUN 1 3 2002
PATENT & TRADEMARK OFFICE

RECEIVED
JUN 2 4 2002
Technology Center 2100

VNET00221465

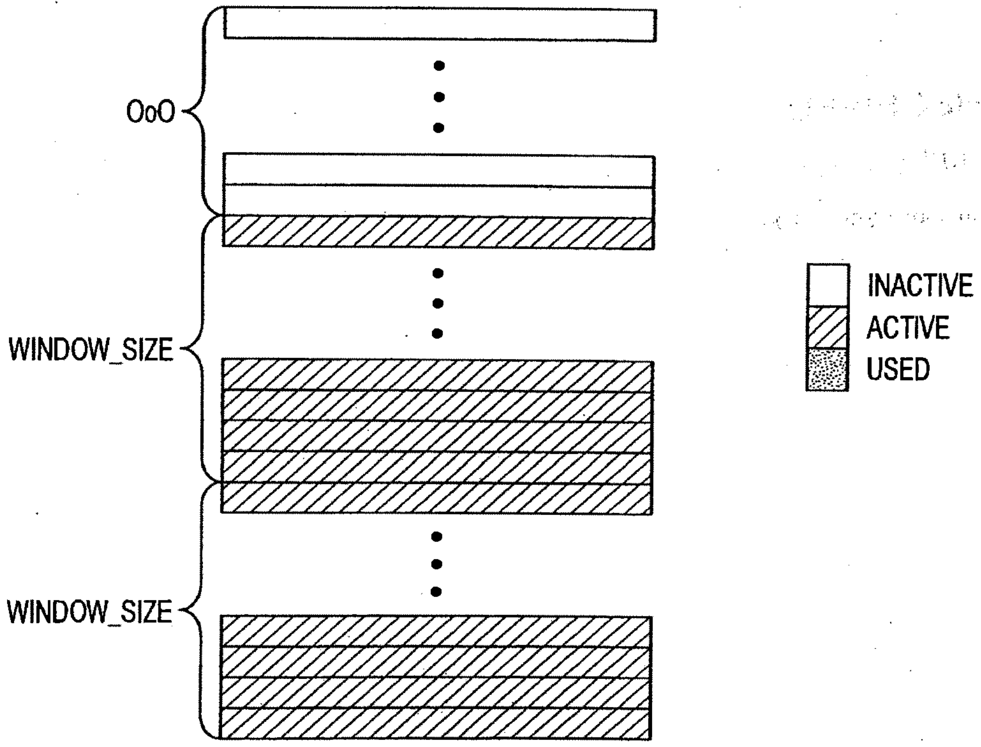


FIG. 17

OIPE JC98 JUN 1 3 2002 PATENT & TRADEMARK OFFICE

RECEIVED
JUN 2 4 2002
Technology Center 2100

VNET00221467

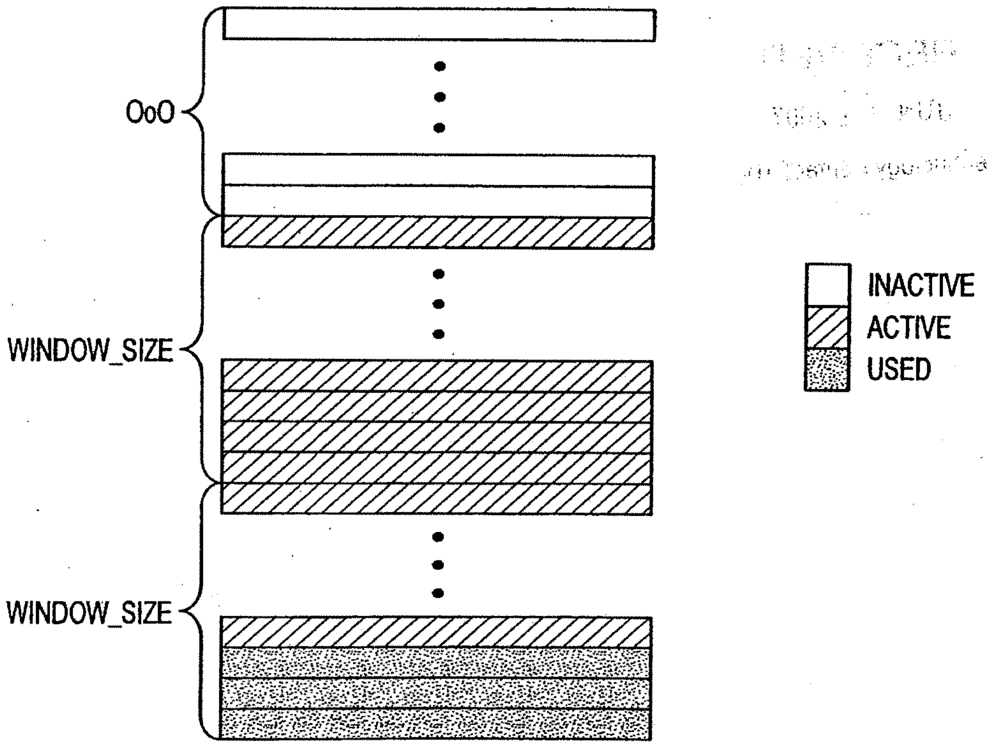


FIG. 18

O I P E JC98
JUN 1 3 2002
PATENT & TRADEMARK OFFICE

RECEIVED
JUN 2 4 2002
Technology Center 2100

VNET00221469

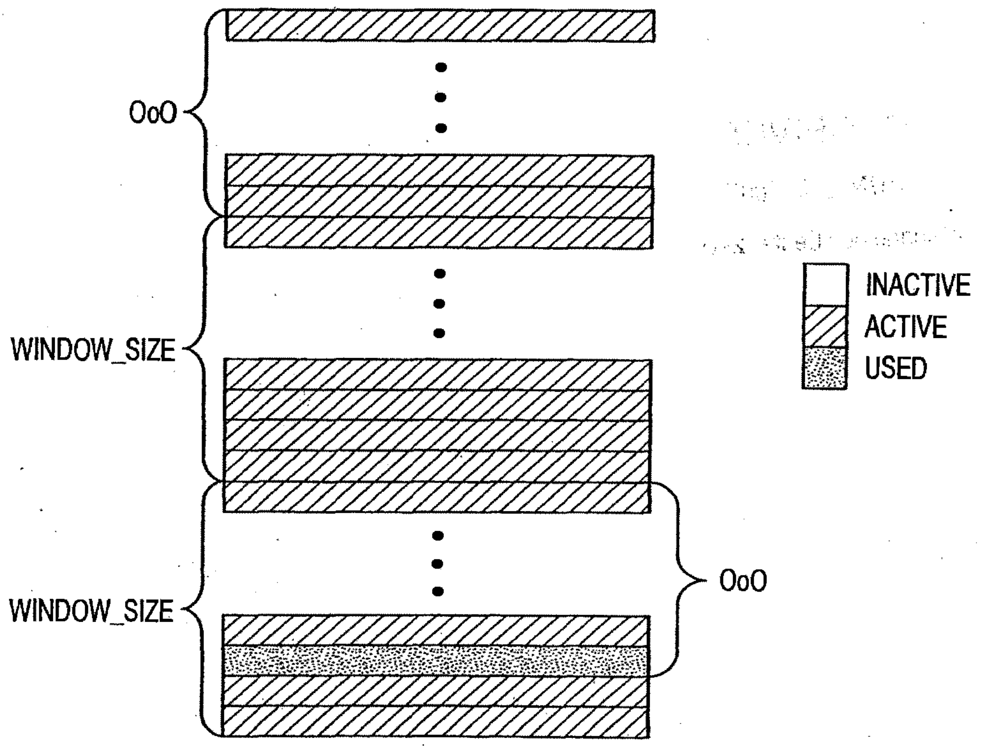
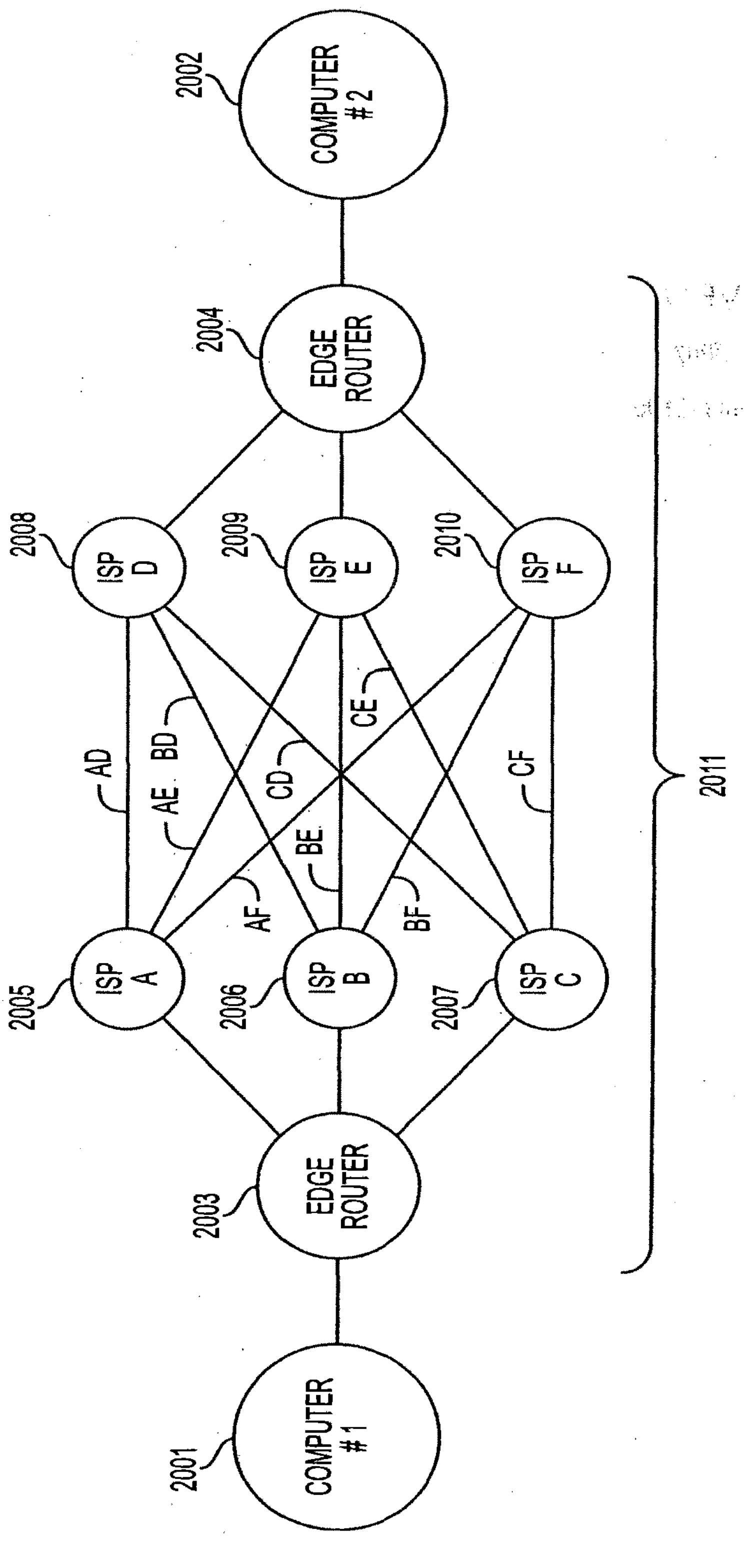


FIG. 19

OIPE JC98 JUN 1 3 2002 PATENT & TRADEMARK OFFICE

RECEIVED
JUN 2 4 2002
Technology Center 2100

VNET00221471



2002
COMPUTER #2
2004
EDGE ROUTER
2008
ISP D
2009
ISP E
2010
ISP F
AD
AE
BD
CD
CE
BE
AF
BF
CF
2005
ISP A
2006
ISP B
2007
ISP C
2003
EDGE ROUTER
2001
COMPUTER #1
2011

FIG. 20

O I P E JC98
JUN 1 3 2002
PATENT & TRADEMARK OFFICE

RECEIVED
JUN 2 4 2002
Technology Center 2100

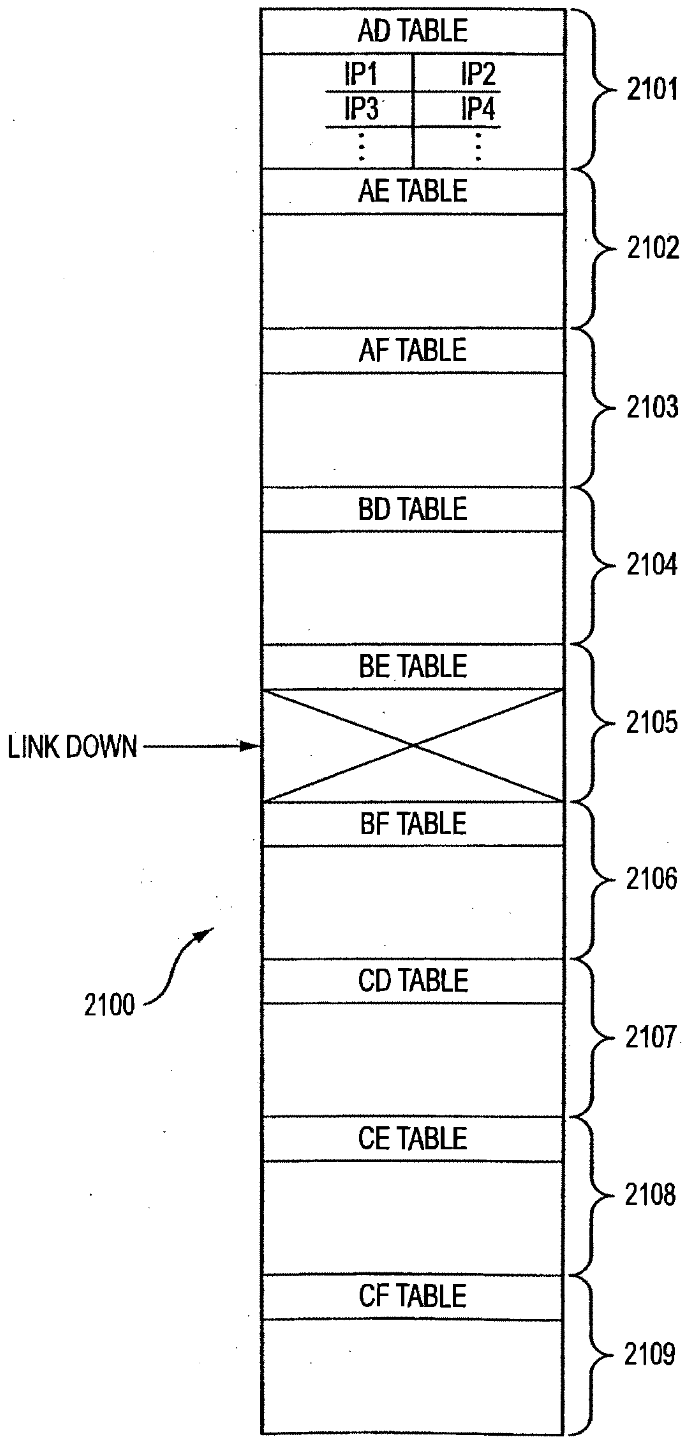
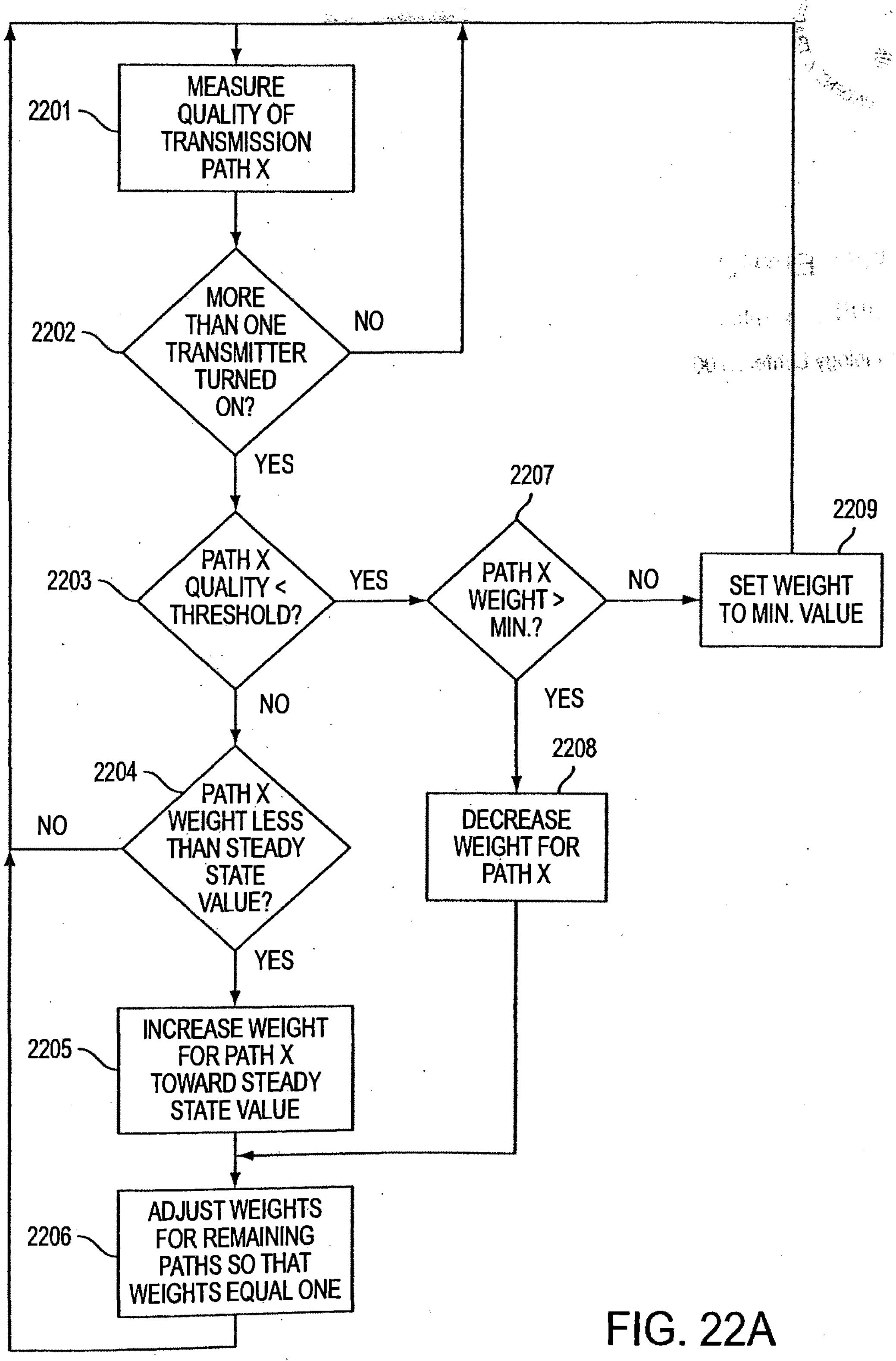

AD TABLE
IP1
IP2
IP3
IP4
AE TABLE
AF TABLE
BD TABLE
BE TABLE
BF TABLE
CD TABLE
CE TABLE
CF TABLE
LINK DOWN
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109

FIG. 21

OIPE JC98 JUN 1 3 2002 PATENT & TRADEMARK OFFICE

RECEIVED
JUN 2 4 2002
Technology Center 2100

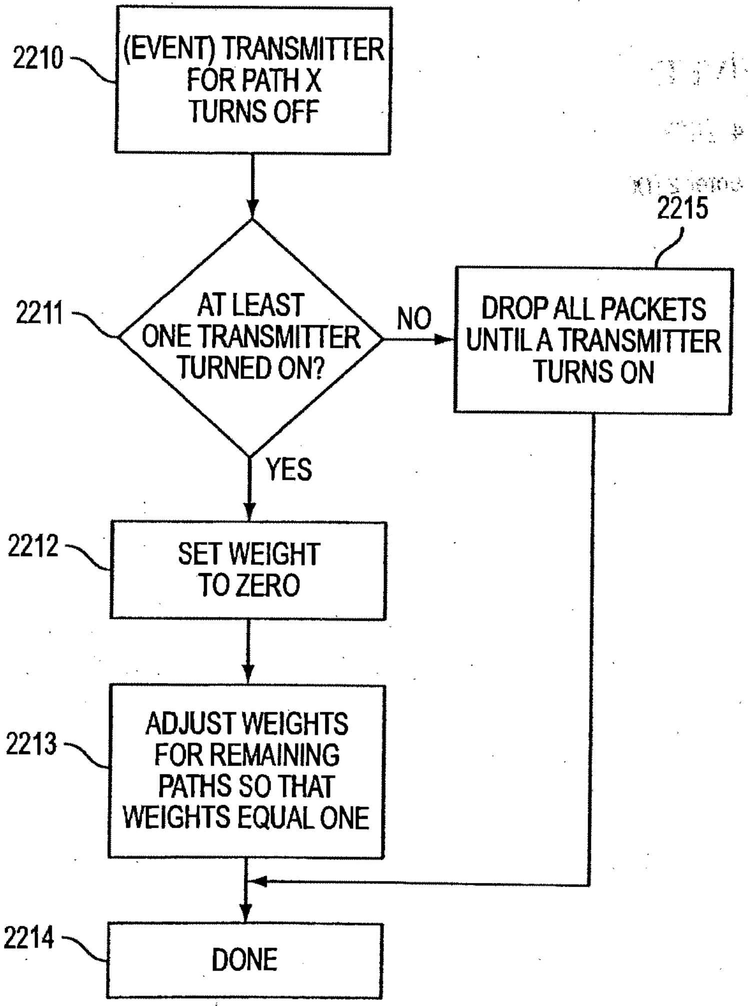


MEASURE QUALITY OF TRANSMISSION PATH X
2201
MORE THAN ONE TRANSMITTER TURNED ON?
2202
NO
YES
PATH X QUALITY < THRESHOLD?
2203
YES
NO
PATH X WEIGHT > MIN.?
2207
NO
YES
SET WEIGHT TO MIN. VALUE
2209
DECREASE WEIGHT FOR PATH X
2208
PATH X WEIGHT LESS THAN STEADY STATE VALUE?
2204
NO
YES
INCREASE WEIGHT FOR PATH X TOWARD STEADY STATE VALUE
2205
ADJUST WEIGHTS FOR REMAINING PATHS SO THAT WEIGHTS EQUAL ONE
2206

FIG. 22A

OIPE JC98 JUN 1 3 2002 PATENT & TRADEMARK OFFICE

RECEIVED
JUN 2 4 2002
Technology Center 2100



2210
(EVENT) TRANSMITTER FOR PATH X TURNS OFF
2211
AT LEAST ONE TRANSMITTER TURNED ON?
NO
2215
DROP ALL PACKETS UNTIL A TRANSMITTER TURNS ON
YES
2212
SET WEIGHT TO ZERO
2213
ADJUST WEIGHTS FOR REMAINING PATHS SO THAT WEIGHTS EQUAL ONE
2214
DONE

FIG. 22B

OIPE JC98 JUN 1 3 2002 PATENT & TRADEMARK OFFICE

RECEIVED
JUN 2 4 2002
Technology Center 2100

3. The method as defined by claim 2, wherein said key K_{ij} is derived from $\propto^{ij}$ mod p using low order key-size bits of $\propto^{ij}$ mod p.

4. The method as defined by claim 3, wherein the key K_{ij} is an implicit pair-wise shared secret used as a key for a shared key cryptosystem (SKCS).

5. The method as defined by claim 4, wherein said SKCS is DES.

6. The method as defined by claim 5, wherein said SKCS is RC2.

7. The method as defined by claim 4, wherein said data packet includes a source address, a destination address and an SKCS identifier field.

8. The method as defined by claim 7, wherein said data packet further includes a message indicator field.

9. The method as defined by claim 4, wherein $\propto$ and p are system parameters, and where p is a prime number.

10. An apparatus for encrypting data for transmission from a first data processing device (node I) to a second data processing device (node J), comprising:

node I including a first storage device for storing a secret value i, and a public value $\propto^i$ mod p;

node J including a second storage device for storing a secret value j, and a public value $\propto^j$ mod p;

node I including an encrypting device for encrypting a data packet to be transmitted to node J, said data packet being encrypted using a first Diffie-Helman (DH) certificate for node J to determine said public value $\propto^j$ mod p;

said encrypting device further computing the value of $\propto^{ij}$ mod p and deriving a key K_{ij} from said value $\propto^{ij}$ mod p;

said encrypting device encrypting a randomly generated transient key K_p from K_{ij}, and encrypting said data packet using said transient key K_p;

node I further including an interface circuit for transmitting said encrypted data packet to said node J.

11. The apparatus as defined by claim 10, wherein said node J further includes:

a receiver for receiving said encrypted data packet from node I;

a decrypting device coupled to said receiver for decrypting said data packet from node I.

12. The apparatus as defined by claim 11, wherein said decrypting device obtains a second DH certificate for said node I and determines said public value $\propto^i$ mod p, and computes the value of $\propto^{ij}$ mod p, said decrypting device further deriving said key K_{ij} from $\propto^{ij}$ mod p.

13. The apparatus as defined by claim 12, wherein said decrypting device utilizes said key K_{ij} to decrypt said transient key K_p, and decrypts said received data packet using said transient key K_p.

14. The apparatus as defined by claim 13, wherein said key K_{ij} is derived from $\propto^{ij}$ mod p using low order key-size bits of $\propto^{ij}$ mod p.

15. The apparatus as defined by claim 14, wherein said key K_{ij} is an implicit pair-wise shared secret used as a key for a shared key cryptosystem (SKCS).

16. The apparatus as defined by claim 15, wherein said data packet includes a source address, a destination address and an SKCS identifier field.

17. The apparatus as defined by claim 16, wherein said data packet further includes a message indicator field.

18. The apparatus as defined by claim 17, wherein $\propto$ and p are system parameters, and where p is a prime number.

19. The apparatus as defined by claim 15, wherein said SKCS is DES.

20. The apparatus as defined by claim 15, where said SKCS is RC2.

* * * * *

US005689566A

United States Patent [19]

Nguyen

[11] **Patent Number: 5,689,566**

[45] **Date of Patent: Nov. 18, 1997**

[54] **NETWORK WITH SECURE COMMUNICATIONS SESSIONS**

[76] Inventor: **Minhtam C. Nguyen**, 10018 Lexington Estates Blvd., Boca Raton, Fla. 33428

[21] Appl. No.: **547,346**

[22] Filed: **Oct. 24, 1995**

[51] **Int. Cl.⁶** **H04K 1/00**

[52] **U.S. Cl.** **380/25**; 380/29; 380/49

[58] **Field of Search** 380/25, 23, 24, 380/4, 46, 49, 29

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,227,253	10/1980	Ehrsam et al.	375/2
5,060,263	10/1991	Bosen et al.	380/25
5,073,852	12/1991	Siegel et al.	395/700
5,111,504	5/1992	Esserman et al.	380/21
5,136,716	8/1992	Harvey et al.	395/800
5,142,622	8/1992	Owens	395/200
5,220,655	6/1993	Tsutsui	395/325
5,226,172	7/1993	Seymour et al.	395/800
5,239,648	8/1993	Nukui	395/600
5,241,599	8/1993	Bellovin et al.	380/21
5,261,070	11/1993	Ohta	395/425
5,263,165	11/1993	Janis	395/725
5,268,962	12/1993	Abadi et al.	380/21
5,301,247	4/1994	Rasmussen et al.	380/43
5,311,593	5/1994	Carmi	380/23
5,323,146	6/1994	Glaschick	340/825.34
5,369,707	11/1994	Follendore, III	380/25
5,373,559	12/1994	Kaufman et al.	380/30
5,375,207	12/1994	Blakely et al.	395/200
5,392,357	2/1995	Bulfer et al.	380/33
5,416,842	5/1995	Aziz	380/30
5,418,854	5/1995	Kaufman et al.	380/23

OTHER PUBLICATIONS

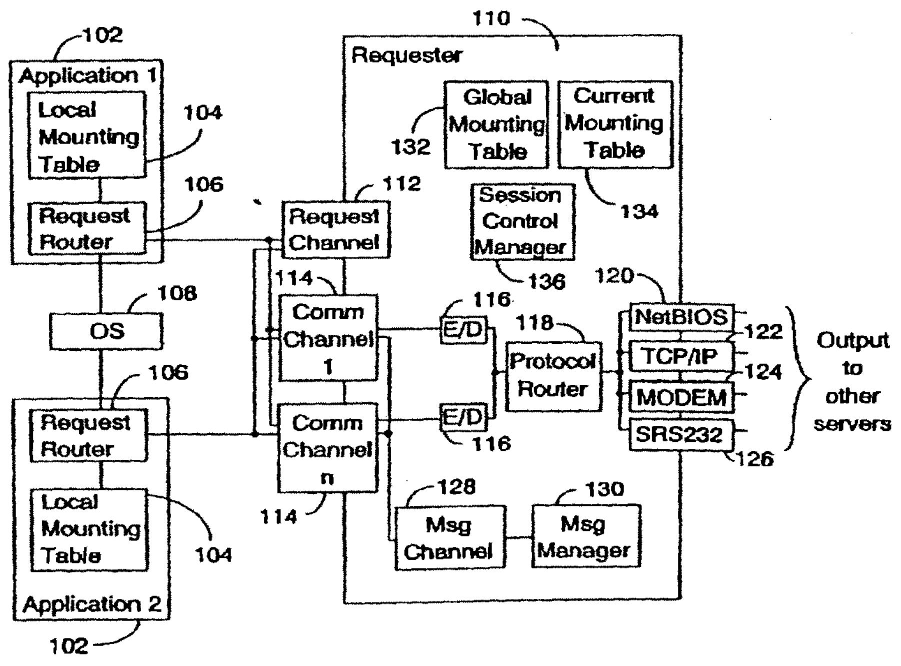
Bruce Schneier, *Applied Cryptography* (second edition), New York, NY. John Wiley & Sons, Inc. 1996, pp. 298–301.

Primary Examiner—David C. Cain
Attorney, Agent, or Firm—John C. Smith

[57] **ABSTRACT**

A system which uses three way password authentication, encrypting different portions of a logon packet with different keys based on the nature of the communications link. Nodes attached to a particular LAN can have one level of security for data transfer within the LAN while data transfers between LANs on a private network can have a second level of security and LANs connected via public networks can have a third level of security. The level of security can optionally be selected by the user. Data transfers between nodes of a network are kept in separate queues to reduce queue search times and enhance performance.

20 Claims, 13 Drawing Sheets



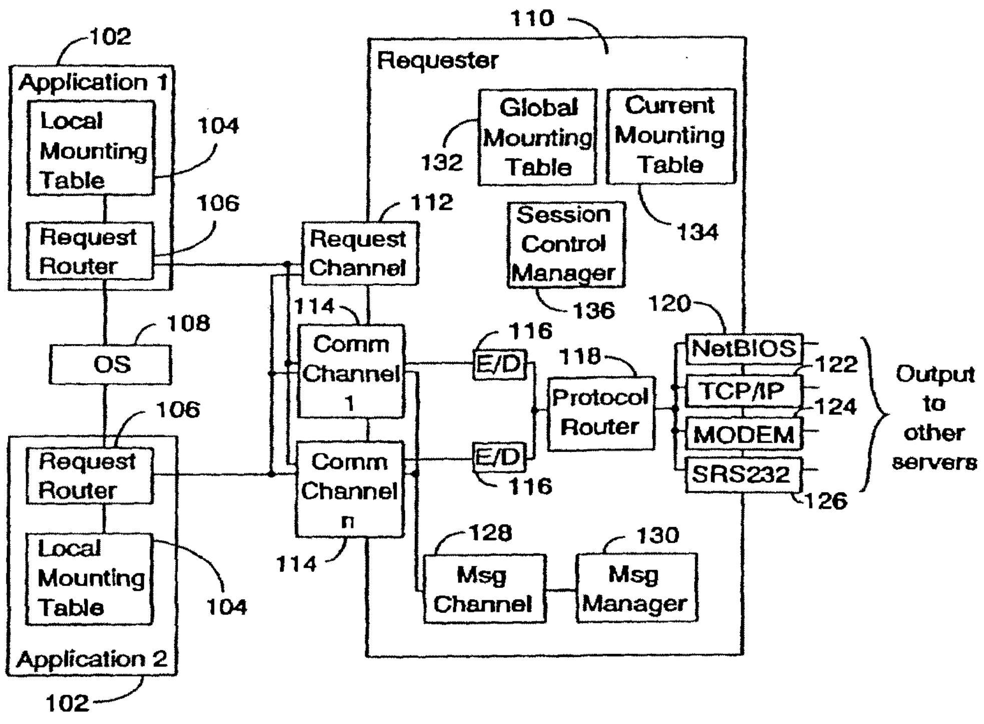

102
Application 1
Local Mounting Table
104
Request Router
106
OS
108
106
Request Router
Local Mounting Table
104
Application 2
102
110
Requester
Global Mounting Table
132
Current Mounting Table
134
112
Request Channel
Session Control Manager
136
114
Comm Channel 1
Comm Channel n
114
116
E/D
E/D
116
118
Protocol Router
120
NetBIOS
TCP/IP
MODEM
SRS232
122
124
126
Output to other servers
128
Msg Channel
130
Msg Manager

Figure 1

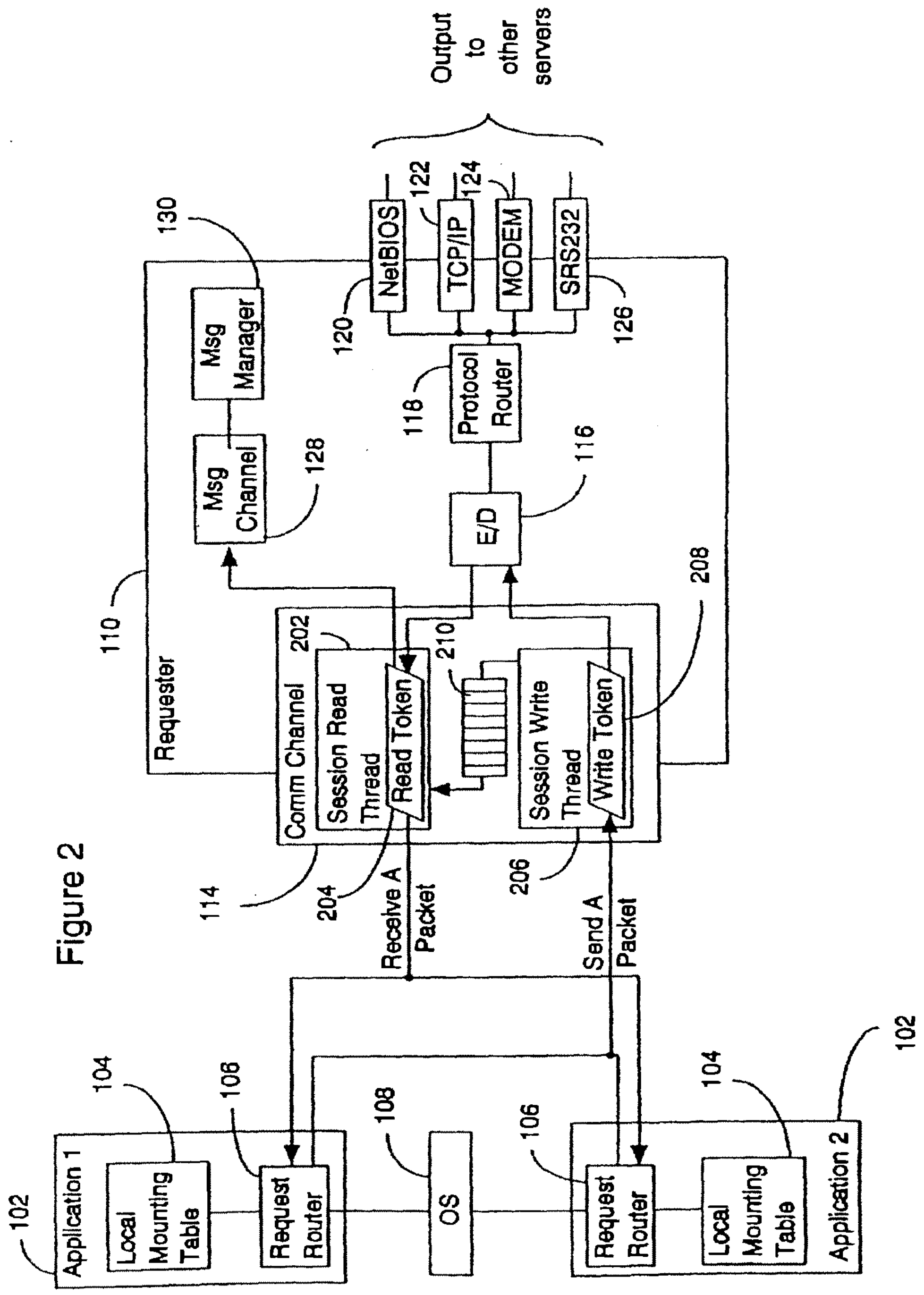

Output to other servers
NetBIOS
TCP/IP
MODEM
SRS232
120
122
124
126
130
Msg Manager
Msg Channel
128
Protocol Router
118
E/D
116
110
Requester
Comm Channel
202
Session Read Thread
Read Token
210
Session Write Thread
Write Token
208
114
204
206
Receive A Packet
Send A Packet
Application 1
Local Mounting Table
Request Router
102
104
106
OS
108
Application 2

Figure 2

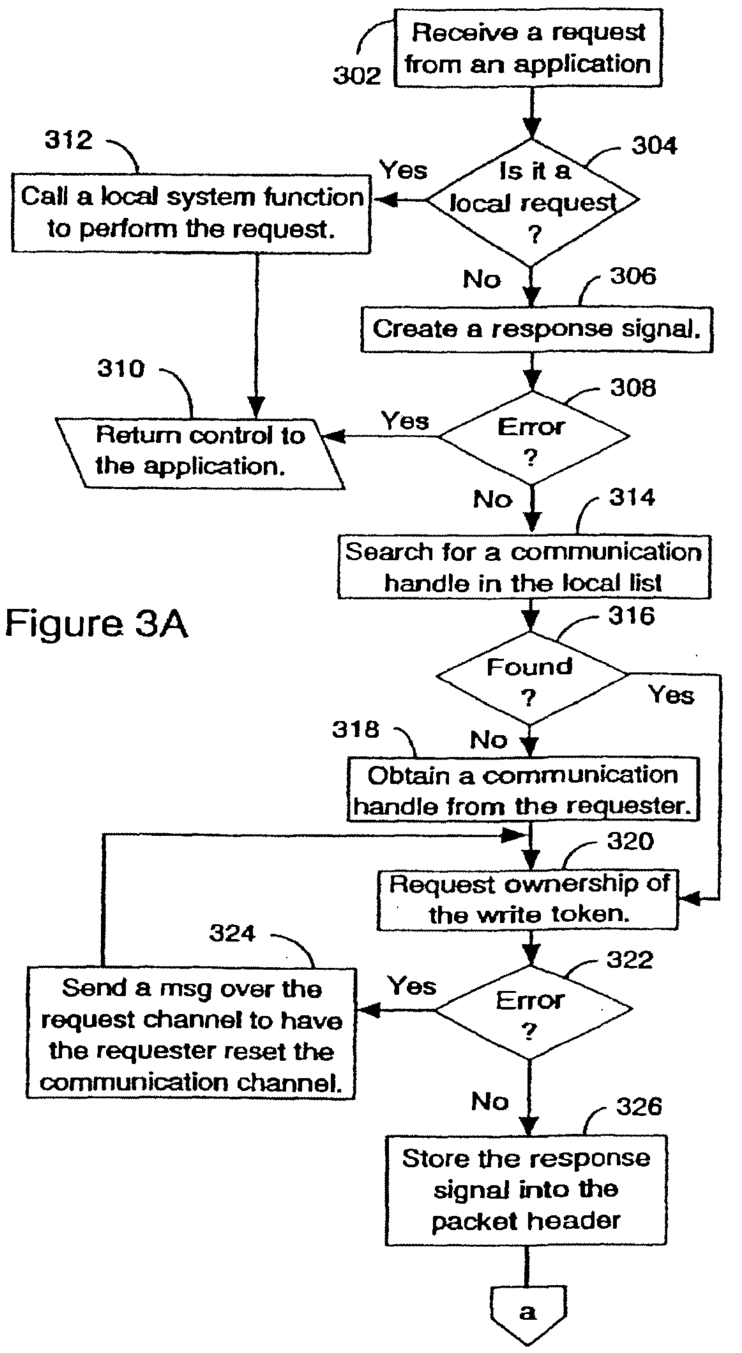
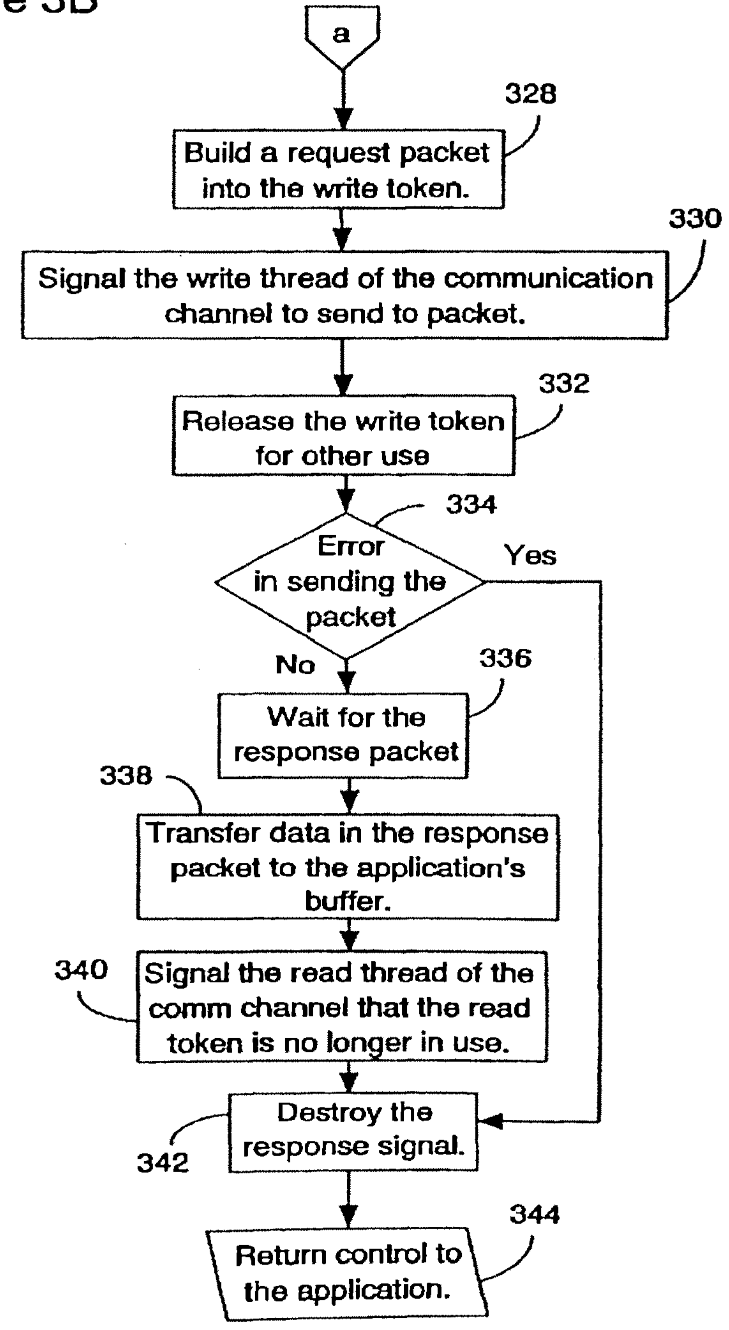

Receive a request from an application
302
Is it a local request ?
304
Yes
Call a local system function to perform the request.
312
No
Create a response signal.
306
Error ?
308
Yes
Return control to the application.
310
No
Search for a communication handle in the local list
314
Found ?
316
Yes
No
Obtain a communication handle from the requester.
318
Request ownership of the write token.
320
Error ?
322
Yes
Send a msg over the request channel to have the requester reset the communication channel.
324
No
Store the response signal into the packet header
326
a

Figure 3A

Figure 3B


a
328
Build a request packet into the write token.
330
Signal the write thread of the communication channel to send to packet.
332
Release the write token for other use
334
Error in sending the packet
Yes
No
336
Wait for the response packet
338
Transfer data in the response packet to the application's buffer.
340
Signal the read thread of the comm channel that the read token is no longer in use.
Destroy the response signal.
342
344
Return control to the application.

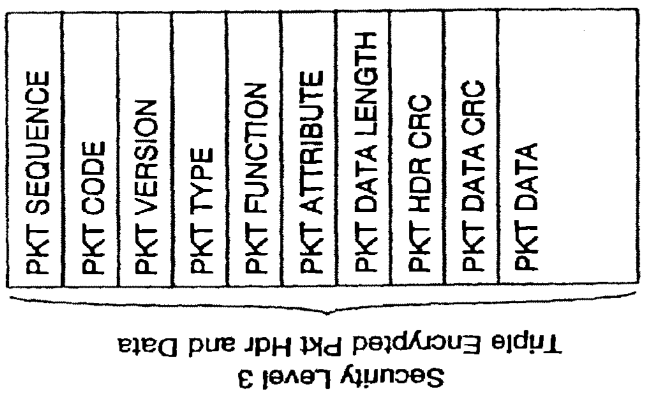


Figure 4C

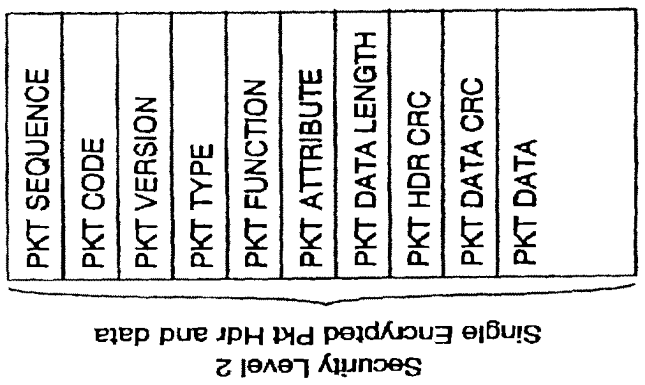


Figure 4B

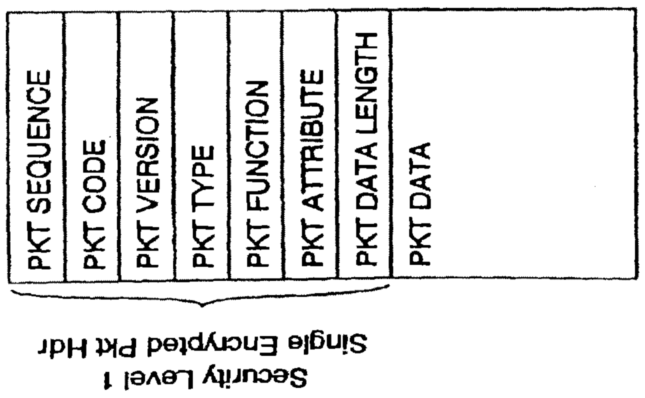


Figure 4A

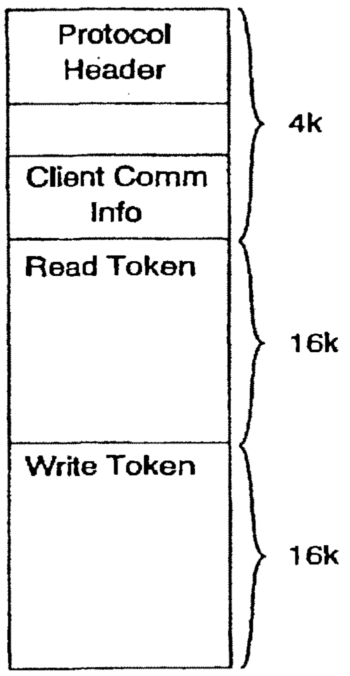


Figure 5A

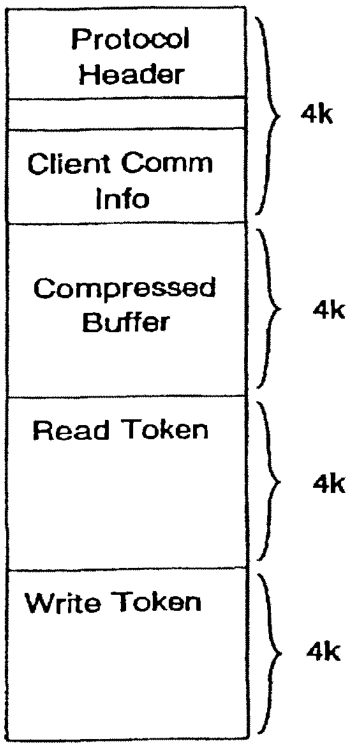


Figure 5B

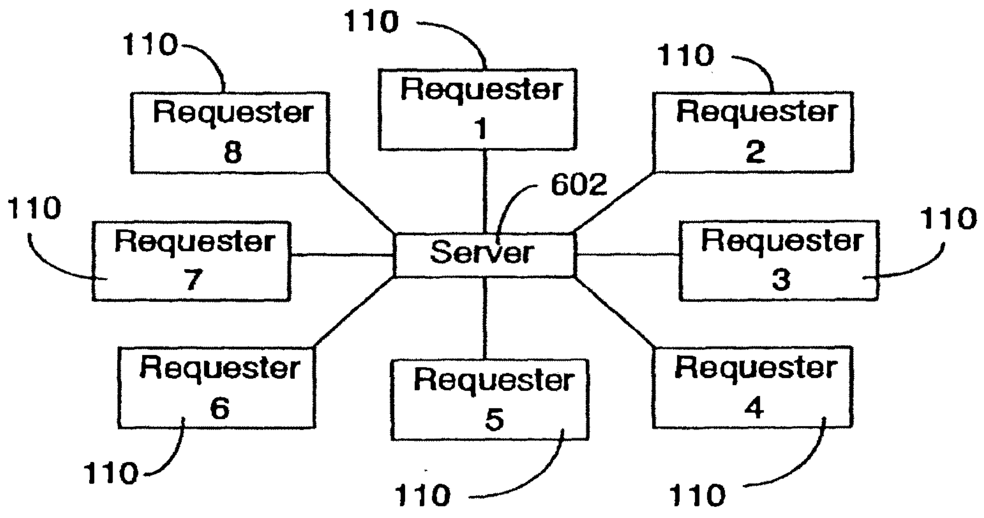


Figure 6

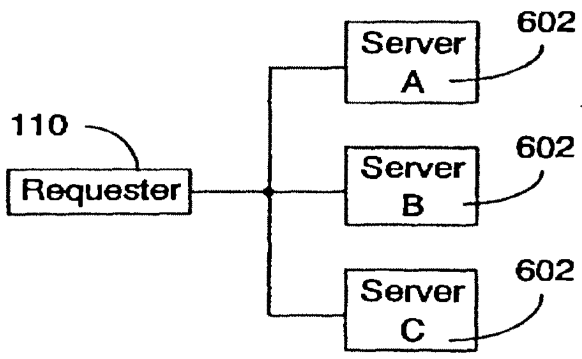


Figure 7

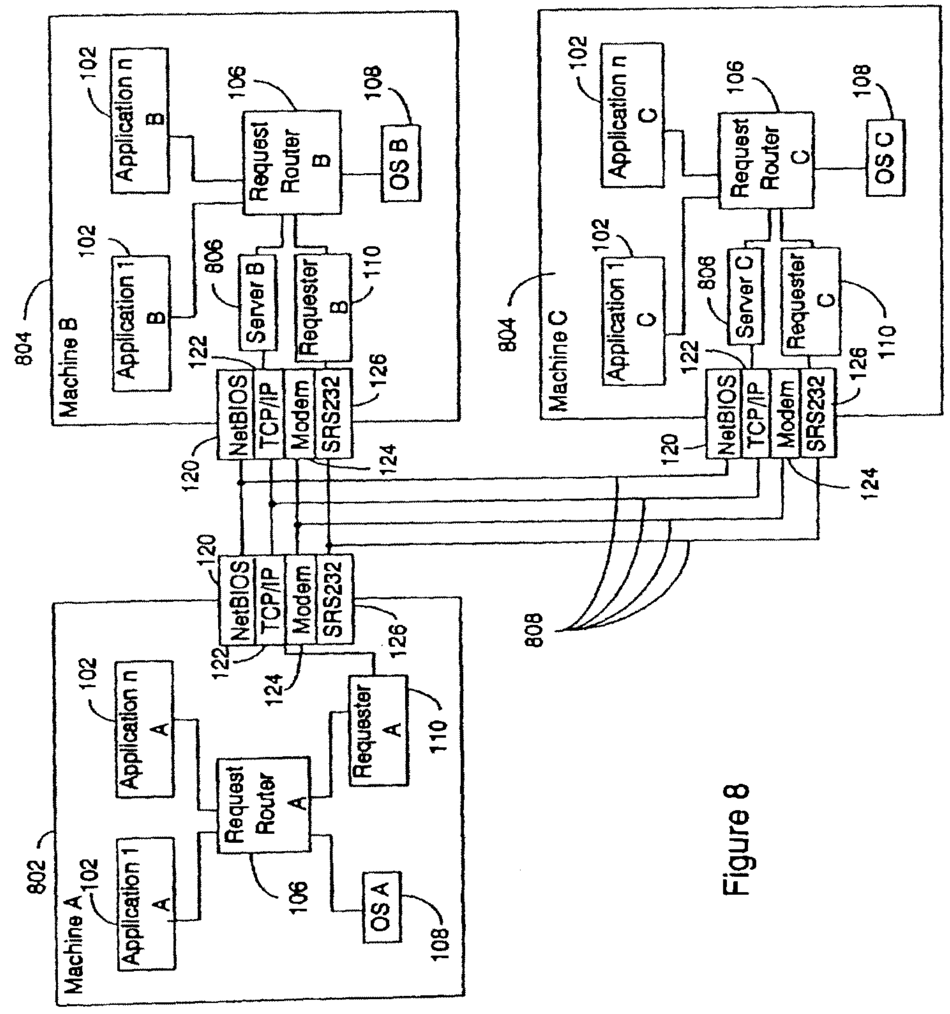

Machine A
802
Application 1 A
Application n A
Request Router A
Requester A
OS A
NetBIOS
TCP/IP
Modem
SRS232
Machine B
804
Application 1 B
Application n B
Request Router B
Server B
806
Requester B
OS B
Machine C
804
Application 1 C
Application n C
Request Router C
Server C
806
Requester C
OS C
102
106
108
110
120
122
124
126
808

Figure 8

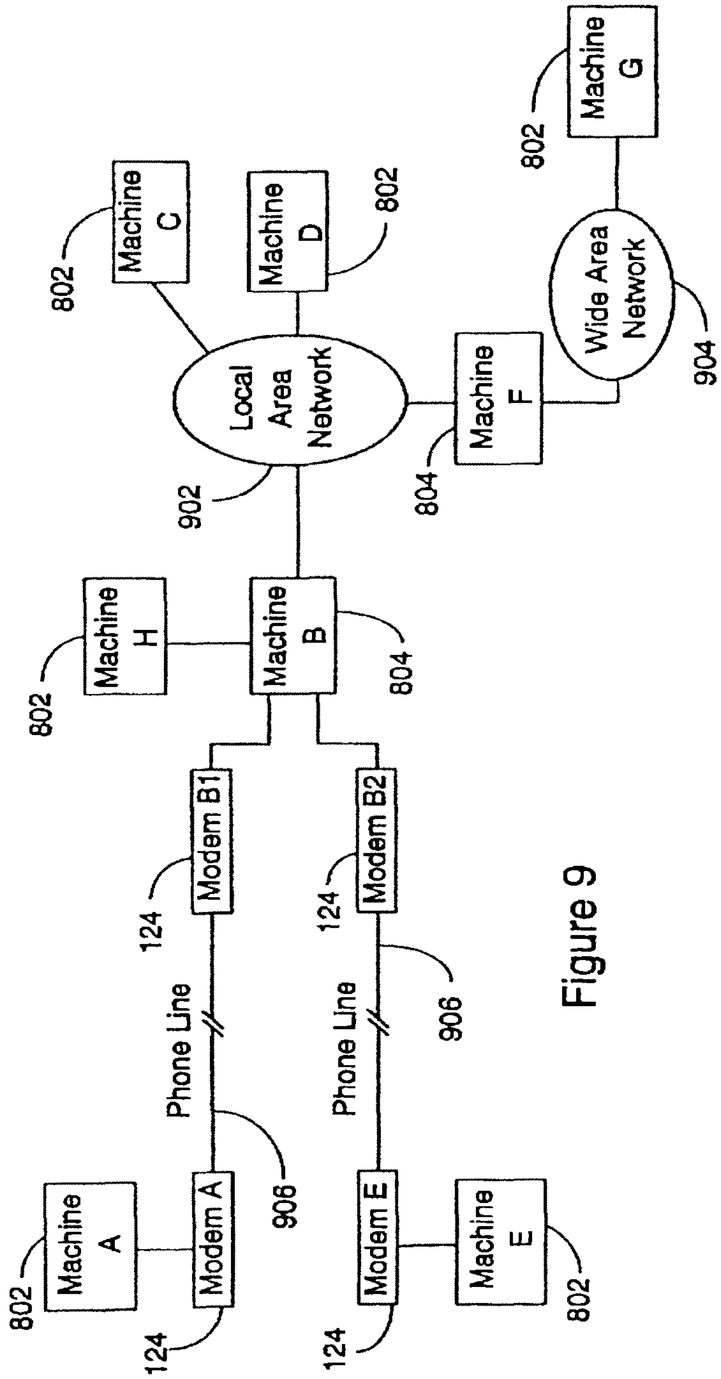

Machine A
802
Modem A
124
906
Phone Line
Modem B1
124
Machine H
802
Machine B
804
Modem B2
124
Phone Line
906
Modem E
124
Machine E
802
Local Area Network
902
Machine C
802
Machine D
802
Machine F
804
Wide Area Network
904
Machine G
802

Figure 9

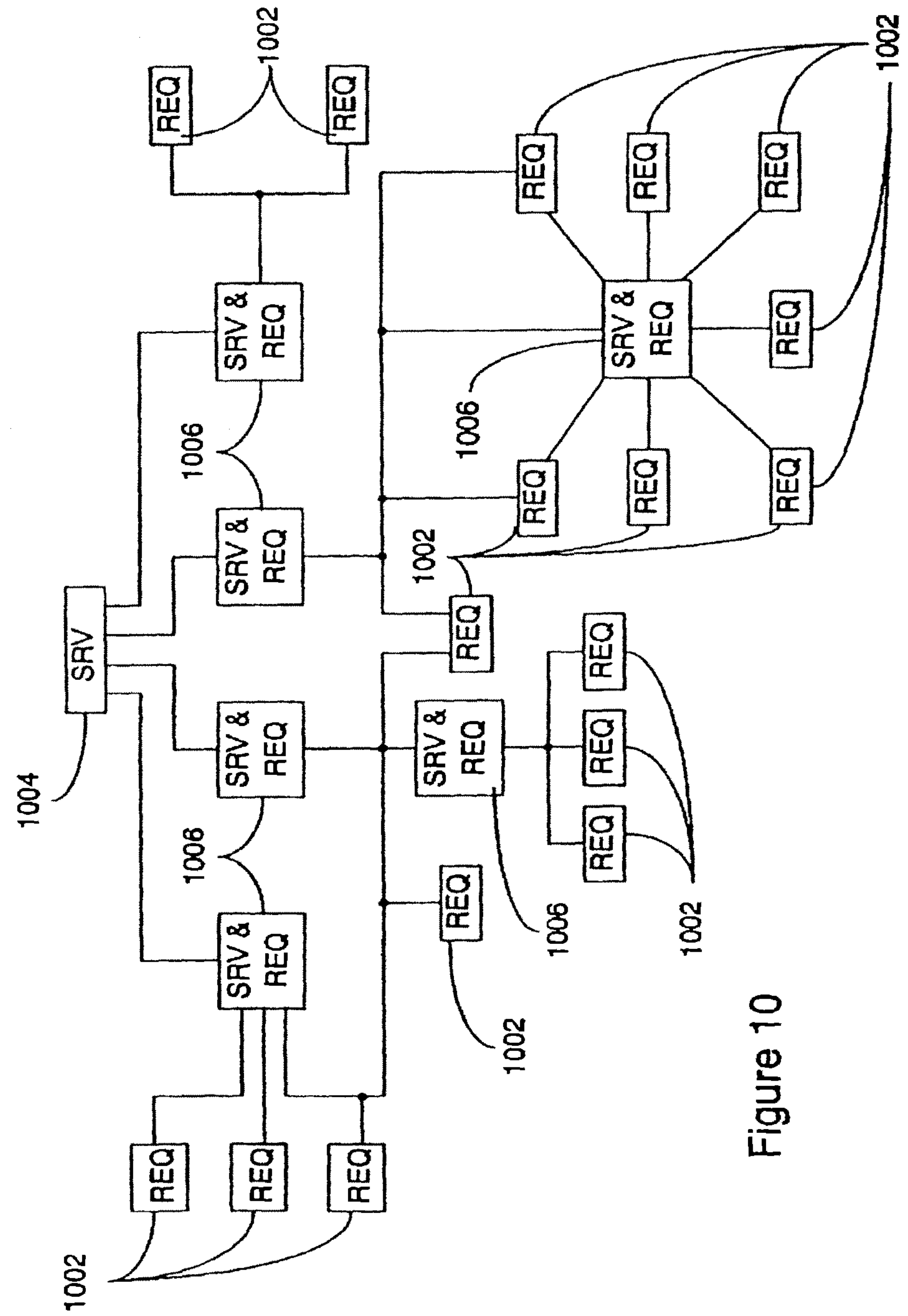


Figure 10

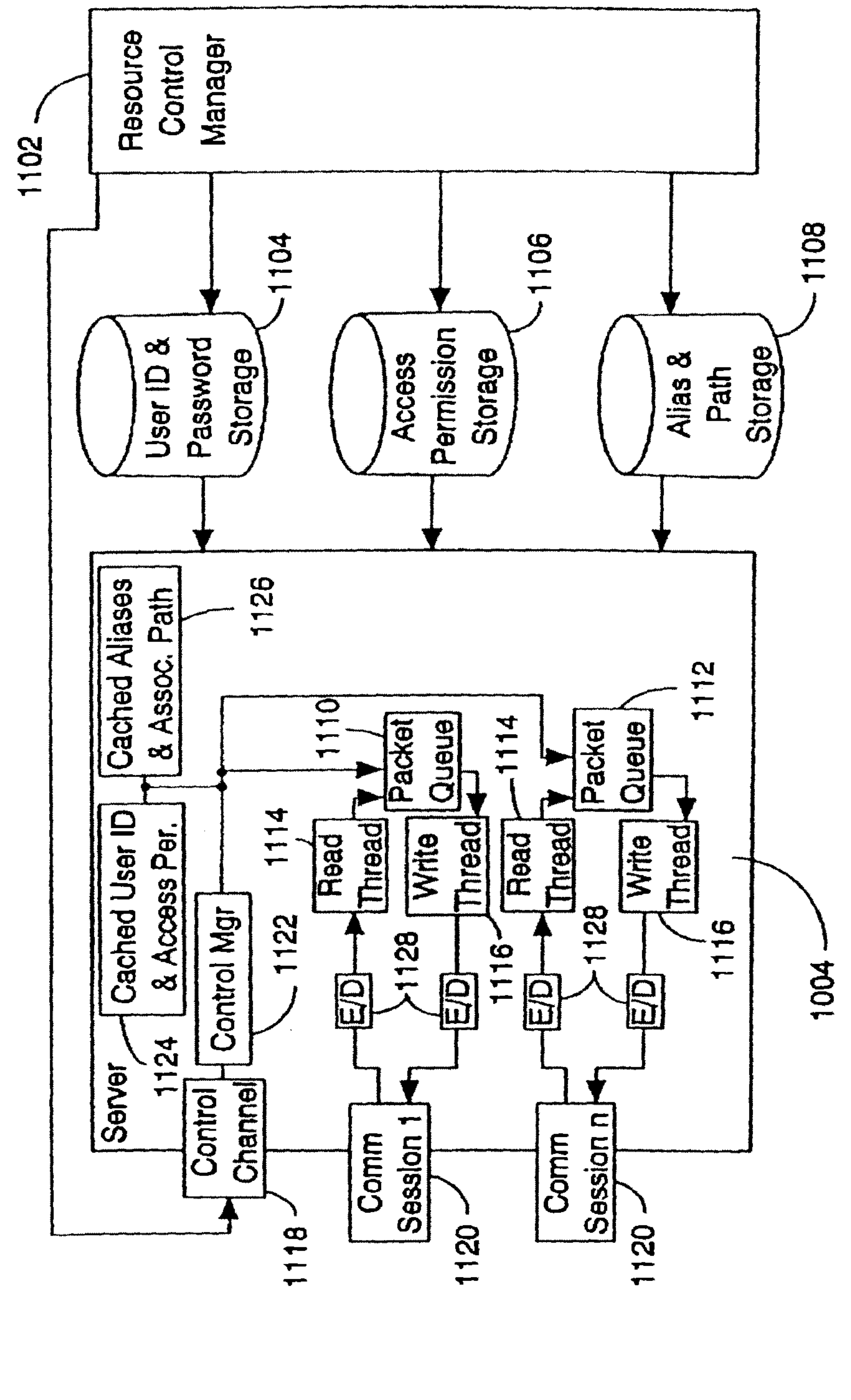

Resource Control Manager
1102
1104
User ID & Password Storage
1106
Access Permission Storage
1108
Alias & Path Storage
Cached Aliases & Assoc. Path
1126
Cached User ID & Access Per.
1124
Control Mgr
1122
Server
Control Channel
1118
1110
Packet Queue
1112
Packet Queue
1114
Read Thread
Write Thread
1116
Read Thread
Write Thread
1128
E/D
E/D
E/D
E/D
Comm Session 1
Comm Session n
1120
1120
1004

Figure 11

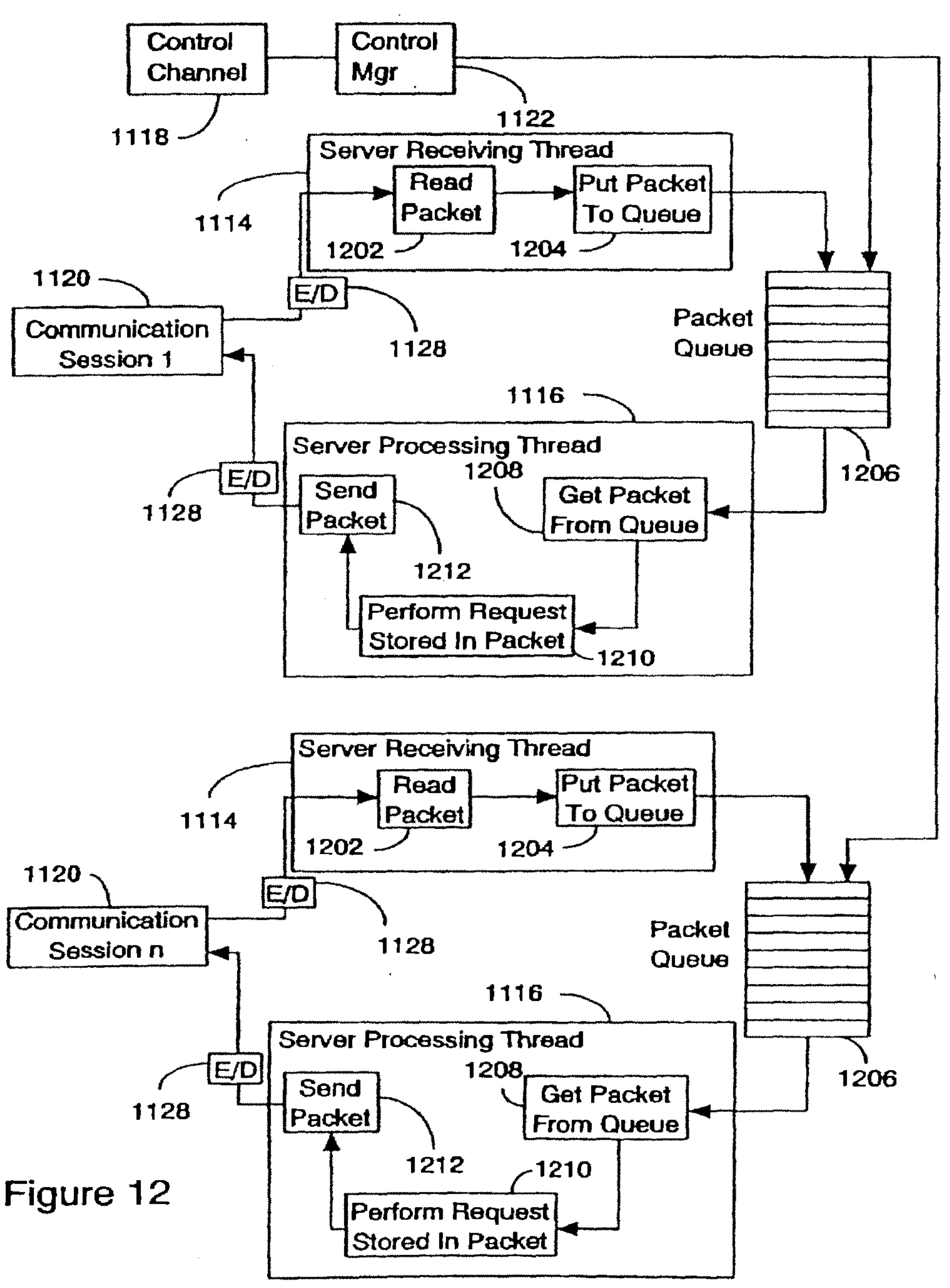

Control Channel
Control Mgr
1118
1122
Server Receiving Thread
Read Packet
Put Packet To Queue
1114
1202
1204
1120
E/D
Communication Session 1
1128
Packet Queue
1116
Server Processing Thread
1208
Send Packet
Get Packet From Queue
1206
1212
Perform Request Stored In Packet
1210
Server Receiving Thread
Read Packet
Put Packet To Queue
1114
1202
1204
1120
E/D
Communication Session n
1128
Packet Queue
1116
Server Processing Thread
1208
Send Packet
Get Packet From Queue
1206
1212
1210
Perform Request Stored In Packet

Figure 12

Figure 13A

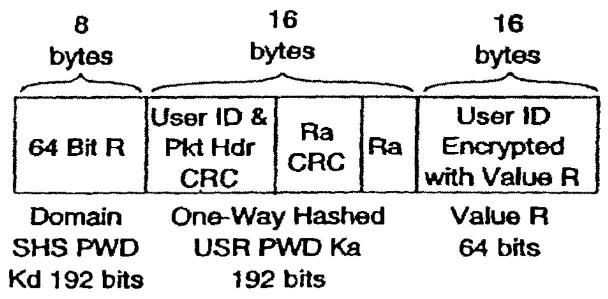
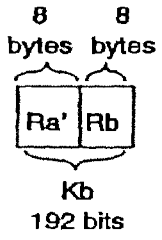

8 bytes
16 bytes
16 bytes
64 Bit R
User ID & Pkt Hdr CRC
Ra CRC
Ra
User ID Encrypted with Value R
Domain SHS PWD Kd 192 bits
One-Way Hashed USR PWD Ka 192 bits
Value R 64 bits

Figure 13B



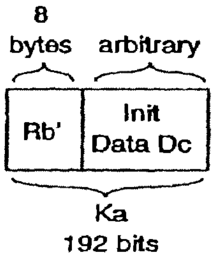


Figure 13C

Figure 13D

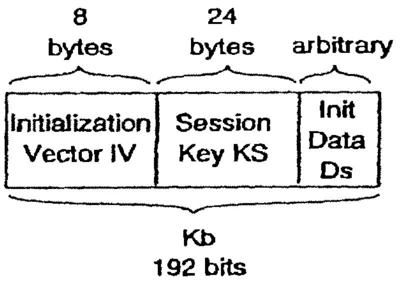
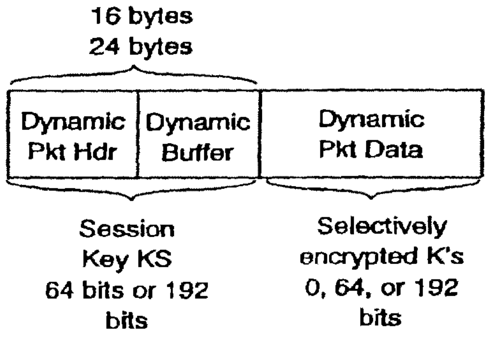


Figure 13E


16 bytes
24 bytes
Dynamic Pkt Hdr
Dynamic Buffer
Dynamic Pkt Data
Session Key KS 64 bits or 192 bits
Selectively encrypted K's 0, 64, or 192 bits

NETWORK WITH SECURE COMMUNICATIONS SESSIONS

BACKGROUND OF THE INVENTION

1. Technical Field

The present invention relates to computer network security. In particular, it relates to networks which use dynamic packet headers and multiple levels of packet encryption to transfer data to and from a remote server or to and from another node in the local network.

2. Background Art

The development of small independent systems such as personal computers has provided several benefits to users. By providing each user with their own processor and data storage, personal computers provide consistent performance and data security. A cost of these benefits is the inconvenience which results from the inability to easily access data by other members of an organization.

The use of mainframe systems, and the later development of alternative systems such as LANs (Local Area Networks) and servers reduces the inconvenience of making data available to all members of an organization, but results in unpredictable performance, and more importantly results in exposure of sensitive data to unauthorized parties. The transmission of data is commonly done via packet based systems which have user ID and password information in a header section. Interception of a packet with header information allows the intercepter to learn the user ID and password which will in turn allow future penetration of the user's system and unauthorized access to the user's data. It would be desirable to transmit user identification and password information in a manner which would be indecipherable to an unauthorized interceptor.

Data security is endangered not only by access by outside parties such as hackers, industrial spies, etc, but also to inadvertent disclosure of data to unauthorized members of the organization. For example, data exchange at certain levels of management may cause problems should the information be disclosed to the general employee population. Likewise, the transmission of personal information such as banking codes over networks has exposed individuals using online financial systems to the possibility of fraudulent access to their funds by third parties.

In addition to data security, the use of network systems such as LANs has created performance problems due to the queuing of requests from multiple locations and the unpredictable delays associated with queuing fluctuations. It would be advantageous if a system could provide not only data security, but also more consistent performance.

The prior art has failed to provide network systems which ensure that access to data is restricted to authorized parties while at the same time providing more consistent performance.

SUMMARY OF THE INVENTION

The present invention solves the foregoing problems by providing a system which uses three way password authentication, encrypting different portions of a logon packet with different keys based on the nature of the communications link. Nodes attached to a particular LAN can have one level of security for data transfer within the LAN while data transfers between LANs on a private network can have a second level of security and LANs connected via public networks can have a third level of security. The level of security can optionally be selected by

the user. Data transfers between nodes of a network are kept in separate queues to reduce queue search times and enhance performance.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagram showing the connection between applications and the requester in a local system.

FIG. 2 is the diagram of FIG. 1 with a more detailed view of the requester.

FIGS. 3A–B are a flow diagram illustrating data transfer between the application and requester of the preferred embodiment.

FIGS. 4A–C are diagrams of the memory layout of packet headers used in the preferred embodiment.

FIGS. 5A–B are diagrams showing the memory layout of entries in the packet queue. FIG. 5A is the memory layout used for TCP/IP and NetBIOS. FIG. 5B is the memory layout used by SMODEM or SRS232 communications systems.

FIG. 6 is a diagram of a multi-requester system with a single server.

FIG. 7 is a diagram illustrating a single requester attached to three servers.

FIG. 8 is a diagram showing a requester (machine A) interconnected with two servers (machines B–C).

FIG. 9 is a diagram illustrating multiple requesters connected to servers via local area networks (LANs) and wide area networks and public telephone networks.

FIG. 10 is a diagram illustrating multiple requesters connected to servers and server/requester systems.

FIG. 11 is a diagram illustrating the server used in the preferred embodiment.

FIG. 12 is a diagram illustrating the read/write threads and packet queues used by the server of FIG. 11.

FIGS. 13A–D are diagrams illustrating the packet headers used in the logon procedure of the preferred embodiment.

FIG. 13E are diagrams illustrating the packet headers used during data transfer in the preferred embodiment.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Prior to a detailed description of the figures, a general discussion of the operation of the preferred embodiment follows. A network can take a variety of forms. For example, it can be two personal computers communicating via modem; it can be a single LAN system within a particular facility; it can be a remote server or mainframe system with communications links to individual terminals or personal computers; it can be a network of LANs or other servers each communicating with one another or through one another; or it can be any of the foregoing systems which use not only dedicated communications lines, but also nondedicated communications (i.e. public networks such as the Internet) through a "firewall". The use of the term firewall herein refers to the requirement for increased levels of security to avoid the possibility of unauthorized data access by parties outside of the organization. Likewise, a machine in the network can act as a client or a server depending on the nature of the data transfer.

In the preferred embodiment, communication between a client and a server is as follows. The server waits for connection requests from clients on the network. The server can be started with one or more supported protocols to enable support of a variety of client types on the network.

For example, the server protocols can include, among others, NetBIOS, TCP/IP, SMODEM and SRS232. All of the foregoing protocols are well known in the art.

When a user on a client machine wishes to initiate a data transfer or other function, the client application activates a requester to access resources in the network. When the server receives a request from a client application, it activates a thread to process the request. A thread is an execution unit of an operating system. Operating systems used for this type of system are Microsoft Windows 95 (trademark of Microsoft Corporation), Microsoft Windows NT (trademark of Microsoft Corporation), IBM OS/2 (trademark of IBM Corporation). These systems may use multiple session protocols such as NetBIOS and TCP/IP or single session protocols such as SMODEM or SRS232.

In single session protocols such as SMODEM and SRS232, the same thread is used to process the request from a client since a serial port can act as a server or client, but cannot simultaneously act as a server and client. Multiple session protocols create a new thread, referred to as an original thread, and wait for a request from a client. When a request is received, the thread is referred to as a server processing thread which is used to process the client logon.

After the logon is successfully completed, the server processing thread creates a packet queue and a packet thread to receive incoming packets and place them in the packet queue. The server then waits for packets to arrive. On the client side, the client creates a session write thread to initiate contact with the server. In addition, the client creates a second thread which is referred to as the session read thread. This thread is used to receive packets sent from the server to the client.

To use resources on the network, users must first logon the server to prove their identity. A logon request is sent from the client's logon application to the requester on the client computer. Before logon data can be exchanged between the applications and the requester, a command manager is created by the requester to accept application requests. The command manager is responsible for housekeeping requests within the client computer.

In the preferred embodiment, the logon procedure uses a three way authentication to prevent the password from being transferred over the computer and also to allow both the client and the server to authenticate each other. In addition, the authentication procedure prevents unauthorized penetration of the system security by detecting the replaying of packets by third parties.

The three way authentication system encrypts the very first logon packet with different keys for each part of the packet as follows.

The first step takes place at the client computer as follows.

1—The client generates a 32 bit random number value which is concatenated to a predefined 32 bit constant to form a 64 bit value R.

2—The CRC signature C1 of the 64 bit value R and the user ID is calculated. This signature value allows detection of packet manipulation.

3—The 64 bit value R is used as a DES key to encrypt the user ID. This makes the user ID look random for each logon packet.

4—The client generates a 192 bit key K from the server name to encrypt the 64 bit value R.

5—The client generates a key Ka from the user ID and password using a one way hash function such as the Secure Hash Standard (SHS) specified in the Federal Information Processing Standards Publication 180 (FIPS PUB 180).

6—The client generates a random number Ra, calculates its CRC signature C2, and encrypts them with the signature C1 using the key Ka. This signature is used to validate the key Ka by the server.

The second step in the process takes place at the server. When the server receives the first logon packet it decrypts the packet as follows.

1—The server generates a key K2 from its machine name and the SHS to decrypt the packet header for identification. If the packet header does not contain the predefined constant, the user is unauthorized. This occurs when an unauthorized user tries to access the server over the phone line but does not know the server name (since the phone number is a public record but the server name is private).

2—If the user is authorized, the server uses the decrypted 64 bit value R in the packet header as a key to decrypt the user ID.

3—The server then uses the user ID to search a database for an access record. If the access record cannot be found, the user has entered an invalid ID and the session is terminated. If the access record is found, the server verifies if the user is allowed access to network resources at this date and time.

4—If access date and time are verified, the server retrieves an associated one way hashed password Kb from an encrypted password file to decrypt the random number Ra and the CRC signatures. The password file is encrypted with a key Kk which is selected by the system administrator at installation.

5—The random numbers Ra and the CRC signatures are then decrypted. The server calculates the CRC signature of the packet header, the user ID and the random number Ra. If the calculated signatures match the decrypted signatures C1 and C2 stored in the packet, and if password Ka matches Kb, the server manipulates the client random number Ra with a predefined formula, generates a random number Rb, and encrypts both random numbers Ra and Rb with the password Kb before sending the first logon response packet to the client.

The third step in the process takes place at the client computer as follows.

1—The client decrypts the first logon response packet.

2—The client manipulates the random number Ra with the predefined formula and compares it with the one returned from the server. If the numbers match, the client knows that it is connected to the correct server, not a fraud server from which an eavesdropper has captured transmissions from the previous logon and is echoing packets back to the client computer.

3—The client manipulates random number Rb with another predefined formula and concatenates it with the client's initiating data (i.e., the client initial packet sequence number, the encryption and compression mode for the session, and the operating system platform ID) to form a second logon packet. The operating system platform ID is useful for selecting protocols and data formats when a particular client or server is communicating with systems that may have any one of a variety of operating system software programs running. The client would typically request encryption and compression mode for the session. However, the server may indicate that the particular modes requested are not available.

4—The client then encrypts the second logon packet and sends it to the server.

The fourth step in the process takes place at the server computer as follows.

1—The server decrypts the second logon packet.

2—The server manipulates the random number Rb with the same predefined formula used by the client and verifies if the random numbers are matched. If the random numbers match, then the server knows it is communicating with an authorized client and that the first logon packet was not a replayed packet.

3—The server saves the client initiating data, generates a session key Ks and an initialization vector IV. In the preferred embodiment, Ks and IV are generated using the formula specified in Appendix C of the ANSI X9.17 standard.

4—Ks and IV are sent to the client along with the server initiating data (i.e., the server initial packet sequence number, supported and/or approved encryption and compression modes for the session, and the server operating system platform ID).

The client and server initial packet sequence numbers are used to detect packet deletion and insertion for data exchanged after the logon procedure.

The fifth step in the process takes place at the client computer as follows.

1—The second logon response packet is decrypted by the client.

2—The client encrypts Ks and IV with its own key and saves them in memory for future communication with the server. The logon procedure completes here.

After the logon procedure is successfully completed, all packet headers are encrypted using the session key Ks and the IV. The packet headers are encrypted to prevent intruders from deleting, inserting, modifying, and/or replaying the packets which may have been captured while data was exchanged over communication lines.

For ease of illustration, the following symbols can be used to illustrate the logon process:

Where:

C=a client

S=a server

E=a symmetric cryptosystem such as DES

K=an encryption key generated from the server name

R=a 32 bit random number concatenated with a predefined constant

Ka=a 192 bit key one way hashed from the user ID and password

Ra=a 64 bit random value generated by C

f( )=a hash function such as CRC to calculate the signature

g( )=a hash function such as CRC to calculate the signatures

UID=user IDs

Kb=a 192 bit one way hashed key retrieved from a database

ha( )=a hash function to manipulate the random number Ra

Rb=a 64 bit random value generated by S

hb ( )=a hash function to manipulate the random number Rb

Dc=client initial data

IV=an initial chaining vector for encryption

Ks=a session encryption key

Ds=server initial data

R'a=ha(Ra)

R'b=hb(Rb)

The logon procedure may be listed as:

1. C to S: EK(R)+EKa(Ra,f(Ra,g(R,UID))+ER(UID)
2. S to C: EKb(R'a,Rb)
3. C to S: EKa(R'b, Dc)
4. S to C: EKb(IV,Ks,Ds)

An important advantage of the authentication procedure used by the preferred embodiment is that both the client and the server verify each other as legitimate without sending the password. In addition, the use of a second set of logon packets which contain different encrypted random numbers precludes access by an unauthorized intruder who merely replays intercepted packets.

The heart of this authentication procedure is in the middle part of the logon packet, which contains the random number Ra and the CRC signatures. Since the CRC signature C2 of the random number Ra is encrypted and sent along with the logon packet, the server can authenticate the user right on the first logon packet. The manipulation of the random numbers Ra and Rb in the challenge-response fashion is to help the server defeat the replaying of the logon packet and to allow the client to authenticate the server and to defeat packet replaying as well.

The 32-bit random number in the packet header is used to make the packet header and the user ID look different for every logon packet. The one-way hashed server name is used as a key to quickly detect invalid logon packets before searching the database. This case may occur frequently when the SMODEM protocol is activated to wait for data transferred over a telephone line (i.e., a wrong number is dialed by accident or a call generated by a manual or automated telemarketing company is being received).

In addition, the server name is isolated from the user ID and password when creating a one-way hashed password to allow the portability of the database. For example, when a business grows, another server may be needed at another location and the database can be easily transferred to the new server. Of course, it would be less time-consuming to delete unauthorized users from the database than to add authorized users to the new one. To better protect the valuable information in the database, a password is required before access to the database is granted. More important, the database can be shared among servers. For example, a server Sb can receive the first logon packet and forward the user ID to a database server Sc within a private network for verification. If an access record is found and the user can access the server Sb at this date and time, the database server Sc returns the encrypted one-way hashed password Kb to the server Sb. The server Sb then continues the challenge-response as if the password Kb is returned from a local database. Note that the database server Sc encrypts the one-way hashed password Kb with the session key defined for communication between the server Sb and Sc before sending it across the private network.

In comparison to prior art systems, the design of this invention provides the server a better opportunity to resynchronize itself if the first logon packet is invalid since the receiver of the authenticating packet is in control of what is next, not the sender. On the other hand, in the prior art the sender is in control of what is next. For example, the sender generates a public key, encrypts it with a shared secret key and sends it to the receiver. If the secret key is invalid, the receiver cannot detect it. Thus, a certain number of packets must be received before the receiver can resynchronize or the receiver might have to use a timeout to resynchronize itself.

Finally, the logon protocol of the preferred embodiment is more suitable for a client/server distributed environment, because this logon protocol allows both client and server to authenticate each other without sending the user password across the communication media and prevent intruders from deleting, inserting, modifying, or replaying the logon packets. In addition, if the logon procedure fails at any point, the server releases all resources and destroys the connection without sending the response packet at that point, i.e., if the user enters a wrong server name in the very first logon packet, nothing is sent out from the server to prevent the user, a potential intruder, from knowing anything about the server. Note that this mutual authentication technique requires the client machine to have a local CPU so that the password will not be transmitted over the network before being encrypted.

The client can now perform a mounting procedure to link a network resource on the server to a virtual disk or it can identify a network resource with the following format \\servername\netname:protocol. The format allows the client to communicate with a network domain using any supported protocols. Further, this protocol can be different from the protocol used to perform the logon procedure. That is, the logon communication protocol can be different from the mounting communication protocol. Also, different virtual disks can be mounted with different protocols to different network domains. This method allows communication between a client and network domains, between a network domain and other network domains using multiple communication protocols simultaneously.

Referring to FIGS. 1 and 2, these figures illustrate the interconnection between a client and a server. FIG. 2 is a more detailed view of the system of FIG. 1.

To perform a file transfer operation, an application 102 calls a request router 106. The request router first verifies if the application 102 requests a local or remote resource. This verification is performed using a local mounting table 104 which the request router 106 obtains from the requester 110 when the application 102 is first started.

If the resource is local, the request router 106 calls a local system function call to perform the request and returns the control to the application 102. However, if the resource is remote, the request router 106 first searches its local list to see if the needed communication handle is already stored in the list. This communication handle contains information of the read 204 and write 208 tokens (shown in FIG. 2) and their associated resources. If the communication handle is not found in the local list, the request router 106 sends a message to the requester 110 over the request channel 112 to obtain the handle. Once the handle is obtained, the request router 106 creates a response signal, i.e., a return address, requests the ownership of the write token 208, stores the response signal into the packet header, builds a packet based on the application's 102 request into the write token 208, and signals the session write thread 206 of the communication channel 114 that there is a packet to send.

If the application data is larger than the packet capacity, the request router 106 can send multiple packets in a series at this point. After the packet is sent to the server, the request router releases the write token for use by another thread in the same process or a different process. If the packet was sent to the server successfully, the request router 106 waits for the corresponding response packets, i.e., a packet can cause multiple response packets returned from the server.

When a response packet arrives, the session read thread uses the response signal to tell the corresponding request router that its response packet has come and is available in the read token. At that time, the read token is accessed exclusively by the designated request router. The router then transfers data in the response packet directly to the application's buffers and signals the session read thread 202 of the communication channel 114 that the read token 206 is no longer in use so that the session read thread 202 can re-use the read token 206 for other incoming packets. Finally, after

all response packets of a request packet have arrived, the request router 106 destroys the response signal and returns control to the application 102. The final response packet is determined by a bit in the packet attribute.

The request router 106 sends a message to the command manager of the requester 110 to request the communication handle containing information of the read 204 and write 208 tokens and their associated resources. If the handle already exists, it is passed to the request router 106 immediately after the requester 110 increments the access count of the handle. However, if the handle does not exist at that time, the requester 110 will load the appropriate communication library, allocate the tokens 204, 208 and their associated resources, create a communication channel consisting of a session write thread 206 to perform auto-logon, create a session read thread 204 for the communication channel 114 if auto-logon is successful, and increment the access count of the handle before passing it to the request router 106.

After receiving the handle, the request router 106 saves the handle for use during the entire lifetime of the application. When the application 102 terminates, the request router 106 will signal the requester 110 of the event so that it can decrement the access count of the handle. When the access count is zero for a certain period of time, the session manager of the requester 110 will drop the communication session, release the tokens 204, 208 and their associated resources, and unload the communication library. Thus, this method allows resources to be allocated upon demand and released when no longer in use. Furthermore, the request router 106 can translate and format data in the application timeslices while the requester 110 is communicating with communication devices 120, 122, 124, 126 to better use the CPU time.

The request router 106 can also perform any preparation necessary to transfer the application 102 request to the requester 110 before requesting the ownership of the write token 208 to reduce the time it takes to access the write token 208. In addition, the request router 106 remembers resources for one application 102 at a time. Thus, it reduces the time to search for the needed information. With this method of sending and receiving packets, data can be exchanged asynchronously between a client and a server with minimum resources in a minimum time. In addition, request packets can be accumulated on the server for processing while the previous response packet is processed by the communication devices 120, 122, 124, 126 or traveling over the network.

Message channel 128 and message manager 130 are used to control system messages transmitted in the system. Current mounting table 134 and global mounting table 132 are used to identify usage of system resources. The session control manager is used to control each session between a client and a server.

FIG. 3A and B is a flowchart which illustrates the transfer of information in a session after the logon procedure has completed. When a resource request 302 is made, the system 304 first tests to see if it is for a local resource 304. If so, a local function is called 312 and control is returned 310 to the application. If it is not a local resource, the system creates a response signal 306. If the response signal 306 cannot be created, control is returned to the application. If it is, then the local list is searched 314 for the communication handle. If the communication handle is not found 316, a communication handle is obtained 318 from the requester and then ownership if the write token is requested 320. However, if the communication handle is found 316, then ownership if the write token is immediately requested 320.

If no error occurs when the request for ownership of the write token is made 322, then the response signal is stored in the packet header 326, a request packet is built into the write token 328, the write thread sends the packet, and the write token is released 332. If an error is detected when the packet is sent, the response signal is destroyed 342 and control is returned 344 to the application. If no errors occur during packet transmission 344, then the system waits 336 for the response packet, the data in the response packet is transferred 338 into the application's buffer, the read token is released 340, the response signal is destroyed 342 and control is returned 344 to the application.

FIGS. 4A–C illustrate the memory layout of the packets used in the preferred embodiment. FIG. 4A illustrates a packet as encrypted by security level 1. In security level 1, the packet header is encrypted using single DES encoding. This level of security incurs the least amount of overhead and is preferably used in more secure environments such as LANs.

FIG. 4B illustrates a packet as encrypted by security level 2. In security level 2, the packet header and data are encrypted using single DES encoding. This level of security incurs slightly increased overhead as compared to security level 1, but provides an increased level of security for less secure environments such as wide area networks.

FIG. 4C illustrates a packet as encrypted by security level 3. In security level 3, the packet header and the data are encrypted using triple DES encoding. This level of security incurs the most overhead as compared to security levels 1 and 2, but provides the highest level of security for insecure environments such as public telephone networks.

To protect data exchanged over communication sessions, the preferred embodiment provides two different encryption schemes available to the user at logon. The first scheme is the US Department of Defense Data Encryption Standard (DES) and the second scheme is the triple-DES specified in the ANSI X9.17 and ISO 8732 standards but with three different keys. In addition, the preferred embodiment applies the Cipher Block Chaining mode specified in the FIPS PUB 81 to better protect the data. Once an encryption scheme is selected, data exchanged over all sessions connected to a network domain are encrypted regardless of the communication protocols being used by the sessions. The price to paid for the encryption is minimum anyway since the preferred embodiment encrypts 500,000 bytes per second when running on a Pentium 66 MHz processor. The operating system used can be any suitable personal computer operating system such a Microsoft (TM) Windows 95 (TM), IBM (TM) OS/2 Warp (TM), Unix, etc. If the server is a large system, any one of a number of suitable mainframe operating system software may be used.

In addition to the above encryption schemes, the preferred embodiment employs a dynamic packet header technique to provide extra securities based on the security level selected by the user at logon. If a security level 2 is selected, the packet header and data are encrypted with DES and the packet header is changed to 24 bytes to carry the CRC signatures of the packet header and data for authentication. However, if a security level 3 is selected, the packet header and data are encrypted with triple-DES using three different keys. Finally, if security level 1 is selected, the packet header remains at 16 bytes and no signature is verified for a better performance but the packet header is encrypted with DES to provide security against other threads. Thus, thanks to the dynamic packet header technique, a user can setup different types of firewalls wherever he needs them. For instance, the user can connect to his office from his home using security level 2 and setup his office machine to connect to another server within his organization using a lower security level to gain a better performance.

In order to provide better security, the preferred embodiment allows the user to select if the data should stay in its encrypted form so that only authorized personnel can view the data. This is important for sensitive business data, personnel data, etc. Of course, the key to decrypt the data must be agreed to ahead of time or exchanged over some secured channels to protect the secrecy of the key.

Of course, those skilled in the art will recognize that the user could also have the capability of instructing the system that no encryption will be used. In this case, no encryption would represent a fourth security level (security level 0). Security level 1–3 having been discussed in regard to FIG. 4.

FIGS. 5A–B illustrate the packet queue structure used in the preferred embodiment. FIG. 5A illustrates the TCP/IP and NetBIOS communications structure and FIG. 5B illustrates the SMODEM and SRS232 communications structure. The compressed buffer is a work buffer used to compress data prior to transmission through SMODEM or SRS232 communication lines. A packet header is placed at the beginning of the read token and at the beginning of the write token. In the preferred embodiment, the read and write tokens are stored in shared memory.

FIG. 6 illustrates a configuration in which multiple requesters 110 communicate with a single server 602.

FIG. 7 illustrates a configuration in which a single requester 110 communicates with multiple servers 602.

FIG. 8 illustrates a configuration in which a system 802 and multiple servers 804 communicate with one another.

FIG. 9 illustrates a configuration in which multiple systems 802 and multiple servers 804 communicate with one another via modems 124 over phone lines 906 and also over LANs 902 and wide area networks 904. This figure illustrates the ability of the system to interface with multiple communications protocols.

FIG. 10 illustrates a configuration in which multiple requester systems 1002, multiple server systems 1004, and multiple server/requester systems 1006 communicate with one another. The configuration in this figure is similar to that shown in FIG. 9.

FIGS. 11 and 12 illustrate a configuration in a server 1004 which includes communication sessions 1120 to communicate with requesters, encrypter/decrypter 1128, read threads 1114, write threads 1116, packet queues 1110, 1112, a resource control manager 1102 to control user ID, access permission and alias and path storage 1104, 1106, 1108. The cached user ID and access permission 1124 and the cached alias and associated path 1126 caches are used to store data from the access permission storage 1106 and the alias and path storage disks 1108 for improved system performance.

To protect resources on the network domains, an access control list (ACL) is used for each network domain in access permission storage 1106. The ACLs are managed by network administrators to define to which resources a user can access and what kind of accesses the user has to each resource. The system provides a sophisticated ACL so that a user cannot view or access any resources other than those assigned. The following access permissions are used by our ACLs:

READ_FILE
WRITE_FILE
CREATE_FILE
DELETE_FILE
EXECUTE_FILE
CHANGE_ATTRIBUTE
ACCESS_SUBDIR
CREATE_SUBDIR
REMOVE_SUBDIR

For example, if the user is not permitted access to any subdirectories from a network resource, the user will not see any subdirectory at all when viewing the network resource. If for some reasons the user knows a particular subdirectory exists under the network resource, he cannot access it anyway. The management of network resources and user access permissions is provided with a user-friendly Graphical User Interface application. Together with the logon procedure, ACLs provide effective protections to the resources on the network domains.

FIG. 12 is a more detailed view of the server 1004 of FIG. 11. A control manager 1122 within the server 1004 is responsible for communication between the server 1004 and other applications on the server 1004 machine. Thus, the server 1004 can be informed if a database has been changed by a resource control application. The server 1004 can also accept a message from another application 102 to send to all or selected clients over active sessions. If an electronic mail system should be needed, the server 1004 can save the message and wait until a client is logged on to send the message over the session. To support these features, the control manager 1122 posts message or e-mail packets to the incoming packet queues 1206 of the sessions 1120. When the server processing threads 1114, 1116 of the sessions 1120 retrieves the packets from the queue 1206, it will process the packets based on the packet types defined in the packet headers.

FIG. 13A–D illustrates the packet headers used in the logon procedure. A session key KS and an initialization vector IV are defined for a communication session between a client and a server 1004 when security level 1 or higher is desired (in security level 0, no encryption is used).

FIG. 13E illustrates a normal packet such as those used during data transfer. When an e-mail or message packet is sent, the preferred embodiment uses security level 2 by default to protect the messages. In security level 2, both packet header and data are encrypted using single DES encryption.

The requester also has the capability to signal request routers 106 of all applications 102 when a communication session is terminated abnormally whether the request routers 106 are sending request packets or waiting on response packets. In order to perform this feature, the response signals (i.e., the return addresses stored in the request packets) are saved in response-signal queues by the session write thread 1116. Each communication session has a response-signal queue 1206 to reduce the search time. When the response packets are successfully delivered, their corresponding response signals are removed from the queue by the session read threads 1114 of the corresponding communication channels. If an application 102 terminates before its response packets arrive, the response packets are discarded and the response signals are also removed from the queue after all chaining response packets have arrived.

In addition, the read thread of the client session also recognizes different types of packets to determine whether it should route the received packets to the application's request router or to a message manager within the requester. The message manager of the requester is responsible for message and e-mail packets sent from the connected servers. This feature is important because it allows the server to initiate the sending of packets while a session is active. As an example, a hot-link can be defined so that a server can inform the connected clients if a database should be changed or a server administrator can send a message to all or selected clients telling them if a server should be out of service shortly, etc. In a more advanced application, an electronic-mail server application can be written so that the message packets are saved on the server until a client is logged on. At that time, the server will send the saved messages to the connected client.

In the prior art, the requester is the one that translates and formats requests from the applications; thus, it cannot perform preparation ahead of time. In addition, information accumulating in one place could increase the search time. The prior art requires its intrinsics modules in both the application and the requester which may require more resources to be allocated and more machine instructions to be executed. Furthermore, the prior art does not have the capability to accumulate multiple request packets from a requester so that the server can process the next packet request while the previous response packet is traveling back to the requester on the network or being processed by communication devices in their own memory buffers.

In contrast to the prior art, the preferred embodiment contains the formatting and translating code in just one place, the request router 106. Our requester only encrypts packet headers and packet data if necessary and then calls the transport functions to send the packets to the server. In addition, requester 110 is also responsible for saving logon and mounting information, managing the communication sessions, and delivering response packets received from multiple network domains to multiple request routers while sending request packets to the multiple network domains. Requester 110 does not need to know the format of the response data, and can deliver the response packets immediately upon receiving them. The request routers 106 can then format or translate the response data in the applications timeslices while the requester 110 is waiting for other incoming response packets or reading data from the communication devices 120, 122, 124, 126. Thus, the preferred embodiment achieves better performance than the prior art.

The prior art also requires the intrinsic modules to translate and format the application data from a program stack segment to a parameter block before sending it to its requester where the data is once again formatted or copied into a data communication buffer. In contrast, the request routers 106 in the preferred embodiment format the application data only once and store the formatted data into the write token which will be used by the requester and the communication subsystem to send the request packets to the server. When the response packets arrive, the requester 110 uses the response signals to tell the corresponding request routers that their response packets have arrived. At that time, the request routers 106 transfer response data directly from the read tokens into the application buffers. Thus, the preferred embodiment eliminates the overhead of copying data between memory buffers.

Furthermore, the prior art does not have the dynamic packet header feature to support packet authentication on demand. Neither does its server authenticate the requester to prevent replaying of packets by intruders. The prior art also requires two different programs running on the server to wait for incoming data from different communication protocols. The preferred embodiment only requires the server to be started once for multiple communication protocols.

In general, a session on the server 1004 will support multiple applications on the requester; thus, a server 1004 must somehow remember the resources allocated for the client applications so that these resources can be released whether the client applications terminate abnormally or the communication sessions are destroyed abnormally. Our server supports this feature in each session thread. Since the allocated resources are isolatedly remembered for different requesters, the search time is minimum every time they are added or removed from the memorized list. In addition, security audit can be turned on and off by the network resource manager running on the server over the control channel of the server. The network resource manager can toggle the security audit for users or groups whose IDs are supplied in the auditing request packet, or resources whose names are stored in the auditing request packet. The audit can also be logged based on successful, failed, or both transactions.

In the prior art, the application is the one which determines if a session should be started on the host computer. The application then makes a function call to connect to the host computer and another function call to start a host server process. In the preferred embodiment, the session manager of the requester determines if a connection should be established to couple the client computer to the server computer. Once the connection is established, the server automatically creates a server processing thread to process the client request packets received over the connection. After the connection is established, the session manager also performs the auto-logon itself, not the application. The session can then be shared by all the applications on the client machine.

Thus, the session creation and logon are transparent to the applications. If the logon is successful, the server creates a server receiving thread to receive and accumulate request packets in a packet queue so that they will be processed by the server processing thread. When a session disconnect request packet is received, the server receiving and processing threads terminate themselves. However, if the communication session is destroyed abnormally, the server receiving thread simulates a disconnect request packet and appends it to the packet queue to signal the server processing thread to terminate. The server receiving thread then terminates itself.

Note that in the very first logon manually performed by the user, the operation is slightly different than the auto-logon mentioned in the above paragraph. The requester first receives a logon request from the logon application, it establishes the session itself and then performs the logon. This is so done by the command manager of the requester, not by the session manager.

Since request packets are accumulated in the packet queue in the preferred embodiment, the request packets may not be processed immediately upon arrival. In contrast, the prior art must process the request packets immediately to return the status or data to the requester. This may indicate that other applications on the client computer must wait until the return packet has arrived and processed before they can send their requests to the same host computer.

The prior art requires an application to send a function call to the host computer to established a communication session. Our system establishes a communication session by the requester when it receives a logon request from the logon program or a request router asking for the communcation handle. In addition, our server has the capability to reformat and retranslate the request packets in its own request router before forwarding them to the requester located on the server when the network resources do not reside on the server. That is, multiple servers can be connected together as shown in FIGS. 7–10 to expand the amount of network resources available to requesters. Note that this feature requires the intermediate servers' administrator(s) to manually logon the designated servers since the logon passwords are not stored on the intermediate servers. Users on requesters can perform this logon remotely if their access permissions in the ACLs of the intermediate servers indicate that they can execute programs on the intermediate servers. However, caution must be taken and security level 3 is advised when using this feature since logon user IDs and passwords must be sent along with the executing request packets.

As shown earlier, the very first logon packet is encrypted with three different keys for different parts of the packet. The header of the logon packet is encrypted with a key generated from the server name. This is design to detect outside intruders early in the verification process. For intruders working inside an organization, the server name may be known. Then it comes the middle part of the logon packet which contains the 64-bit random number and the CRC values. These are the heart of the verification since it is encrypted with the key generated from the user ID and the secret password. This scheme allows the server to detect the intruding logon right on the very first packet. The challenge-response process that following the logon packet is to defeat re-played packets.

The encryption system used in the preferred embodiment has several other advantages, as follows. The long term key is derived from a user ID and a secret password. It has 192 bits and is used in a triple-DES encryption enhanced with CBC. The short term key is generated with the X9.17 key generation formula and changed every time a session is established between two nodes on the network. Thus, the encryption occurs at the application layer which exposes the source and destination addresses of the packets when used with TCP/IP and NetBIOS protocols but the intruders must deal with different keys whose lengths are either 64 or 192 bits for different pair of nodes on the network. In addition, the short term key is encrypted and only sent once when the communication session between two nodes is established, not in every packet; thus, it reduces the traffic between two nodes.

Furthermore, the prior art only protects data between site-firewalls, not between nodes. In many cases, data must be protected between nodes within an organization. For instance, high-rank management officers within a private network may want to exchange restricted confidential information without leaks to their employees.

Encryption at the application layer also reduces the cost of replacing the existing network layer and can be done on demand when protection to data is needed. Different security firewalls can easily be established between any pair of nodes with a single click of the fingertip.

Finally, the communication subsystem of the preferred embodiment is a foundation for multiple applications when their use are in demand. With just one communication session between a client and a server, packet sending can be initiated by either party to conduct file transfers, broadcast messages, or e-mail messages. In addition to minimum resources and maximum performance, security is also provided to protect the secret of the data.

While the invention has been described with respect to a preferred embodiment thereof, it will be understood by those skilled in the art that various changes in detail may be made therein without departing from the spirit, scope, and teaching of the invention. For example, the size of encryption keys can be changed, algorithms used to generate the encryption keys can be changed, the device can be implemented in hardware or software, etc. Accordingly, the invention herein disclosed is to be limited only as specified in the following claims.

I claim:

1. A bi-directional security system for a network, comprising:

at least one client, the client further comprising:

client communication means to communicate with at least one server;

packet reception means to receive transmitted packet data from the server;

means to generate and transmit a first packet to the server, at least a portion of the first packet having a first packet header containing client identifying information;

means to encrypt at least a portion of the client identifying information in the first packet header prior to transmission;

means to decrypt at least a portion of the client authenticating information in a second packet header and to determine if the second packet is from the server, the client further having means to terminate the communication if the second packet is from an invalid server;

means to generate and transmit a third packet to the server, at least a portion the third packet having a third packet header containing session information; and

means to encrypt at least a portion of the session information in the third packet header prior to transmission; and

the server further comprising:

server communication means to communicate with the client;

packet reception means to receive transmitted packet data from the client;

means to decrypt at least a portion of the client identifying information in the first packet header and to determine if the first packet is from a valid client, the server further having means to terminate the communication if the first packet is from an invalid client;

means to generate and transmit a second packet to the client in response to the first packet, at least a portion the second packet having the second packet header containing client authenticating information;

means to encrypt at least a portion of the client authenticating information in the second packet header prior to transmission; and

means to decrypt at least a portion of the session information in the third packet header;

whereby, the client and the server each verify the validity of the other by transmitting encrypted identifying information to one another.

2. A security system, as in claim 1, further comprising:

means in the server to generate and transmit a fourth packet to the client in response to the third packet, the fourth packet having a packet header containing session information; and

means to encrypt at least a portion of the session information in the fourth packet header prior to transmission.

3. A security system, as in claim 2, wherein:

the client has a userid;

the client has a password;

the first packet is encrypted by:

concatenating a random number to a predetermined bit constant to form a value R;

a CRC signature C1 is generated from the value R and the userid;

the value R is used as a DES key to encrypt the userid;

the server name is used to generate a key K to encrypt the value R;

the key Ka is generated by a one way hash function from the userid and password; and

a random number Ra and its CRC signature C2 is generated, Ra and C2 are encrypted using key Ka.

4. A security system, as in claim 3, wherein:

the server further comprises an encrypted client password file;

the second packet is encrypted by:

a key K2 is generated from the server name and a one way hash function to decrypt the packet header of the first packet;

the userid is decrypted using the decrypted value R from the packet header;

the decrypted userid is used to access an authorization table to determine if the first packet is valid;

the userid is used to extract a one way hashed password Kb from the encrypted client password file, the password Kb is then used to decrypt values Ra, C1 and C2;

the value Ra is manipulated via a predetermined formula to produce a random number R'a;

a random number Rb is generated by the server; and

R'a and Rb are encrypted with password Kb, inserted into the packet header of the second packet and transmitted to the client.

5. A bidirectional security system for a network, comprising:

at least one client, the client further comprising:

means to encrypt a first logon packet;

means to transmit the first logon packet to the server;

means to decrypt the second logon packet;

means to encrypt a third logon packet with session information;

a server, further comprising:

means to decrypt the first logon packet;

means to encrypt a second logon packet with client authenticating information;

means to transmit the second logon packet to the client;

means to decrypt the third logon packet; and

a communication channel capable transmitting packets between the client machine and the server;

whereby the client and server can establish secure communications by bi-directionally transmitting encrypted data.

6. A security system, as in claim 5, further comprising:

means to encrypt packet data in least two security levels, the first security level having a first packet encryption scheme and the second security level having a second packet encryption scheme;

whereby the security system can selectably encrypt packet data with at least two packet encryption schemes.

7. A security system, as in claim 6, further comprising:

means to encrypt packet data at least three security levels, the third security level having a third packet encryption scheme;

whereby the security system can selectably encrypt packet data with at least three packet encryption schemes.

8. A security system, as in claim 7, wherein the first packet encryption scheme is a single DES encryption.

9. A security system, as in claim 8, wherein the second packet encryption scheme is a triple DES encryption.

10. A security system, as in claim 9, wherein:

the first packet encryption scheme encrypts the packet header information; and

the second packet encryption scheme encrypts the packet header information;

the third packet encryption scheme is a triple DES encryption, and further encrypts the packet header and the packet data.

11. A security system, as in claim 10, wherein:

the server further comprises means to encrypt a fourth logon packet with session information; and

the client further comprises means to decrypt the fourth logon packet.

12. A security system, as in claim 9, wherein:

the client further comprises means to encrypt data packets; and

the server further comprises means to encrypt data packets;

data packets are selectably encrypted using at least one of the security levels; and

means to dynamically adjust the size of the packet header based on the selected encryption scheme.

13. A security system, as in claim 5, wherein:

each client includes at least one application program; and

the server further comprises at least one packet queue for each client;

whereby application performance is improved by reducing packet search time.

14. A method of securely transmitting packet data between a client and a server with encrypted packets, including the steps of:

using at least one communication channel to transmit packets between at least one client machine and at least one server;

encrypting in the client a first logon packet;

transmitting the first logon packet to the server;

decrypting the first logon packet in the server;

encrypting a second logon packet in the server with client authenticating information;

transmitting the second logon packet to the client;

decrypting the second logon packet in the client;

encrypting in the client a third logon packet with session information;

decrypting the third logon packet in the server;

whereby the client and server can establish secure communications by bi-directionally transmitting encrypted data.

15. A method, as in claim 14, including the further steps of:

encrypting a fourth logon packet in the server with session information;

transmitting the fourth logon packet to the client; and

decrypting the fourth logon packet in the client;

using the session information to control encryption of packets while communicating between the client and the server.

16. A method, as in claim 15, including the further step of using at least two selectable encryption schemes, including a first encryption scheme for a first security level and a second encryption scheme for a second security level.

17. A method, as in claim 16, including the further steps of:

using at least two communication channels to communicate between multiple client and server, at least a first communication channel having a first level of security and at least a second communication channel having a second level of security; and

selecting the first encryption scheme for the first communication channel and the second encryption scheme for the second communication channel.

18. A method, as in claim 17, including the further step of using single DES encryption for the first level of security and triple DES encryption for the second level of security.

19. A method, as in claim 18, including the further steps of:

using packets which contain a header portion and a data portion; and

using a third encryption scheme in which triple DES encryption is used for the packet header and the packet data.

20. A method, as in claim 19, including the further steps of:

selecting the encryption scheme based on the nature of the data in the packet; and

dynamically adjusting the size of the packet header based on the selected encryption scheme.

* * * * *

US005878231A

United States Patent [19]

Baehr et al.

[11] **Patent Number: 5,878,231**

[45] **Date of Patent: Mar. 2, 1999**

[54] **SYSTEM FOR PACKET FILTERING OF DATA PACKETS AT A COMPUTER NETWORK INTERFACE**

[75] Inventors: **Geoffrey G. Baehr**, Palo Alto; **William Danielson**, Mountain View; **Thomas L. Lyon**, Palo Alto, all of Calif.; **Geoffrey Mulligan**, Colorado Springs, Colo.; **Martin Patterson**, Grenoble, France; **Glenn C. Scott**, Mountain View; **Carolyn Turbyfill**, Los Gatos, both of Calif.

[73] Assignee: **Sun Microsystems, Inc.**, Palo Alto, Calif.

[21] Appl. No.: **795,374**

[22] Filed: **Feb. 4, 1997**

Related U.S. Application Data

[62] Division of Ser. No. 444,351, May 18, 1995, Pat. No. 5,802,320.

[51] **Int. Cl.**^{6} **G06F 13/38**; G06F 15/17

[52] **U.S. Cl.** **395/200.75**; 395/200.73

[58] **Field of Search** 370/401, 403; 395/200.7, 200.8, 200.71, 200.73, 200.75

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,577,313	3/1986	Sy	371/403
5,550,984	8/1996	Gelb	395/200.75
5,559,883	9/1996	Williams	380/4
5,590,285	12/1996	Krause et al.	395/200.48
5,606,668	2/1997	Shwed	395/187.01
5,623,601	4/1997	Vu	395/187.01

OTHER PUBLICATIONS

"Firewalls and Internet Security," by Cheswick & Bellovin, Addison Wesley, 1994.

"Firewall Routers and Packet Filtering," by Gary Kessler, Feb. 1995.

Ip–masq.c from Linux kernel (v.2.0.27), 1994.

Ip–fw.c from Linux kernel (v 2.0.27), 1994.

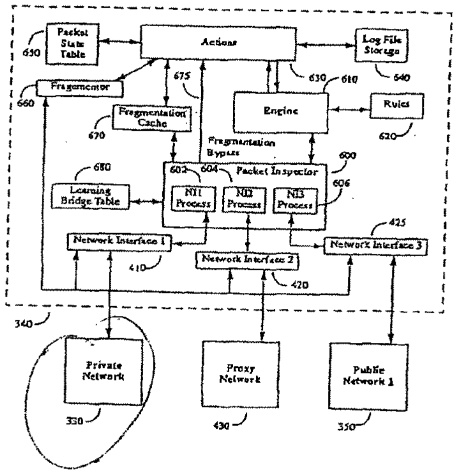
Primary Examiner—David L. Robertson

Attorney, Agent, or Firm—Finnegan, Henderson, Farabow, Garrett & Dunner, L.L.P.

[57] **ABSTRACT**

A system for screening data packets transmitted between a network to be protected, such as a private network, and another network, such as a public network. The system includes a dedicated computer with multiple (specifically, three) types of network ports: one connected to each of the private and public networks, and one connected to a proxy network that contains a predetermined number of the hosts and services, some of which may mirror a subset of those found on the private network. The proxy network is isolated from the private network, so it cannot be used as a jumping off point for intruders. Packets received at the screen (either into or out of a host in the private network) are filtered based upon their contents, state information and other criteria, including their source and destination, and actions are taken by the screen depending upon the determination of the filtering phase. The packets may be allowed through, with or without alteration of their data, IP (internet protocol) address, etc., or they may be dropped, with or without an error message generated to the sender of the packet. Packets may be sent with or without alteration to a host on the proxy network that performs some or all of the functions of the intended destination host as specified by a given packet. The passing through of packets without the addition of any network address pertaining to the screening system allows the screening system to function without being identifiable by such an address, and therefore it is more difficult to target as an IP entity, e.g. by intruders.

12 Claims, 7 Drawing Sheets



VNET00222004

US006332158B1

(12) **United States Patent**
Risley et al.

(10) **Patent No.: US 6,332,158 B1**
(45) **Date of Patent: Dec. 18, 2001**

(54) **DOMAIN NAME SYSTEM LOOKUP ALLOWING INTELLIGENT CORRECTION OF SEARCHES AND PRESENTATION OF AUXILIARY INFORMATION**

(76) Inventors: **Chris Risley**, 372 Stevick Dr., Atherton, CA (US) 94027; **Richard Lamb**, 11 Roxbury Ave., Natick, MA (US) 01760; **Eduard Guzovsky**, 11 Page Rd., Weston, MA (US) 02493

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/207,701**

(22) Filed: **Dec. 9, 1998**

Related U.S. Application Data

(63) Continuation-in-part of application No. 09/204,855, filed on Dec. 3, 1998, now abandoned.

(51) **Int. Cl.**^{7} **G06F 13/00**

(52) **U.S. Cl.** **709/219**; 709/225; 709/313; 709/329

(58) **Field of Search** 709/217, 219, 709/223, 224, 225, 226, 227, 229, 313, 328, 329

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,855,020	*	12/1998	Kirsch	707/10
5,907,680	*	5/1999	Nielsen	707/533
6,041,041	*	3/2000	Ramanathan et al.	370/241
6,041,324	*	3/2000	Earl et al.	707/9
6,092,178	*	7/2000	Jindal et al.	712/27

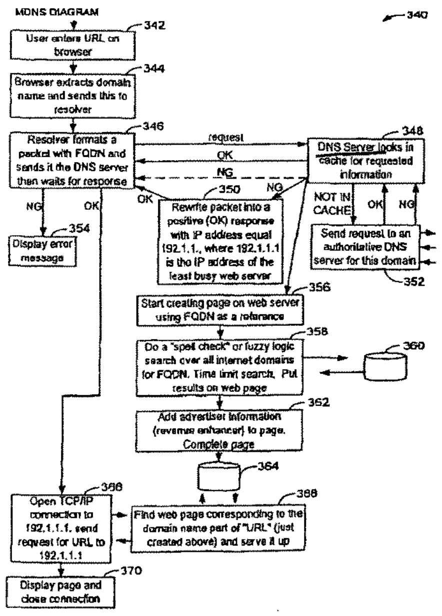
* cited by examiner

Primary Examiner—Viet D. Vu
(74) *Attorney, Agent, or Firm*—Townsend and Townsend and Crew LLP; Charles L. Kulas; Fidel D. Nwamu

(57) **ABSTRACT**

A domain name server assists user's in selecting desired domains in the Internet. A domain name query is sent from a resolver process, or equivalent process, when the user (or a process on the user's computer) wishes to obtain information. If the domain name exists, the domain name server provides the corresponding machine address back to the user's computer. However, when the domain name query uses a non-existent domain name then a machine address for a computer that executes a domain recommendation engine is returned instead of a machine address associated with the invalid domain. The domain recommendation engine assists the user (or process on the user's computer) in locating a desired domain name. The domain name recommendation engine can take into account numerous factors that assist in determining the intended domain, including common misspellings, phonetic errors, sub-domain errors, past statistics on website accessing by the present user and prior users. Auxiliary information is provided to the user along with information to assist in locating the intended domain. The auxiliary information can include sponsorship information, referrals, advertisements, educational or other information. The auxiliary information can be in the form of image, audio, database of other types of information.

22 Claims, 7 Drawing Sheets


MDNS DIAGRAM
340
342
User enters URL on browser
344
Browser extracts domain name and sends this to resolver
346
Resolver formats a packet with FQDN and sends it the DNS server then waits for response
request
OK
NG
348
DNS Server looks in cache for requested information
350
NG
OK
Rewrite packet into a positive (OK) response with IP address equal 192.1.1., where 192.1.1.1 is the IP address of the least busy web server
NOT IN CACHE
OK
NG
Send request to an authoritative DNS server for this domain
352
NG
OK
354
Display error message
356
Start creating page on web server using FQDN as a reference
358
Do a "spell check" or fuzzy logic search over all internet domains for FQDN. Time limit search. Put results on web page
360
362
Add advertiser information (revenue enhancer) to page. Complete page
364
366
Open TCP/IP connection to 192.1.1.1. send request for URL to 192.1.1.1
368
Find web page corresponding to the domain name part of "URL" (just created above) and serve it up
370
Display page and close connection

US005898830A

United States Patent [19]

Wesinger, Jr. et al.

[11] **Patent Number: 5,898,830**

[45] **Date of Patent: Apr. 27, 1999**

[54] **FIREWALL PROVIDING ENHANCED NETWORK SECURITY AND USER TRANSPARENCY**

[75] Inventors: **Ralph E. Wesinger, Jr.**, San Jose; **Christopher D. Coley**, Morgan Hill, both of Calif.

[73] Assignee: **Network Engineering Software**, San Jose, Calif.

[21] Appl. No.: **08/733,361**

[22] Filed: **Oct. 17, 1996**

[51] **Int. Cl.^{6}** **G06F 1/00**

[52] **U.S. Cl.** **395/187.01**; 395/200.55; 395/200.57

[58] **Field of Search** 395/186, 187.01, 395/188.01, 200.3, 200.55, 200.68, 200.57; 380/3, 4, 21, 23, 25; 340/825.3

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,713,753	12/1987	Boebert et al.	364/200
4,799,153	1/1989	Hann et al.	380/25
4,799,156	1/1989	Shavit et al.	364/401
5,191,611	3/1993	Lang	380/25
5,241,594	8/1993	Kung	380/4
5,416,842	5/1995	Aziz	380/30

(List continued on next page.)

OTHER PUBLICATIONS

Kiuchi et al., "C–HTTP The Development of a Secure, Closed HTTP Based Network on the Internet", PRoceedings of SNDSS, IEEE, pp. 64–75, Jun. 1996.

Neuman, "Proxy Based Authorization and Accounting for Distributed Systems", IEEE, pp. 283–291, 1993.

Network Firewalls; *IEEE Communications Magazine*; (Ballovin et al.) pp. 50–57; Sep., 1994.

The MITRE Security Perimeter; *IEEE Communications Magazine*; (Goldberg); pp. 212–218; 1994.

IpAccess—An Internet Service Access System for Firewall Installations; *IEEE Communications Magazine*; (Stempel); pp. 31–41; 1995.

Remote Control of Diverse Network Elements Using SNMP; *IEEE Communications Magazine*; (Aicklen et al.); pp. 673–667; 1995.

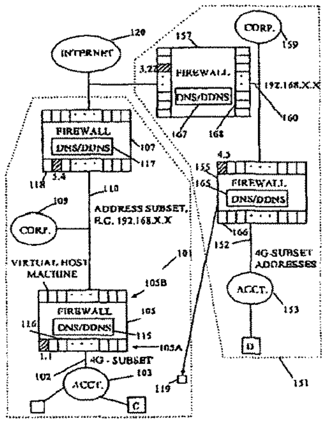
Firewall's Information is Money!, *Scientific Management Corporation.*

Primary Examiner—Joseph Palys
Attorney, Agent, or Firm—McDonnell Boehnen Hulbert & Berghoff

[57] **ABSTRACT**

The present invention, generally speaking, provides a firewall that achieves maximum network security and maximum user convenience. The firewall employs "envoys" that exhibit the security robustness of prior-art proxies and the transparency and ease-of-use of prior-art packet filters, combining the best of both worlds. No traffic can pass through the firewall unless the firewall has established an envoy for that traffic. Both connection-oriented (e.g., TCP) and connectionless (e.g., UDP-based) services may be handled using envoys. Establishment of an envoy may be subjected to a myriad of tests to "qualify" the user, the requested communication, or both. Therefore, a high level of security may be achieved. The usual added burden of prior-art proxy systems is avoided in such a way as to achieve fall transparency-the user can use standard applications and need not even know of the existence of the firewall. To achieve full transparency, the firewall is configured as two or more sets of virtual hosts. The firewall is, therefore, "multi-homed," each home being independently configurable. One set of hosts responds to addresses on a first network interface of the firewall. Another set of hosts responds to addresses on a second network interface of the firewall. In one aspect, programmable transparency is achieved by establishing DNS mappings between remote hosts to be accessed through one of the network interfaces and respective virtual hosts on that interface. In another aspect, automatic transparency may be achieved using code for dynamically mapping remote hosts to virtual hosts in accordance with a technique referred to herein as dynamic DNS, or DDNS.

21 Claims, 9 Drawing Sheets



US006079020A

United States Patent [19]

Liu

[11] **Patent Number: 6,079,020**

[45] **Date of Patent: Jun. 20, 2000**

[54] **METHOD AND APPARATUS FOR MANAGING A VIRTUAL PRIVATE NETWORK**

[75] Inventor: **Quentin C. Liu**, Cupertino, Calif.

[73] Assignee: **VPNet Technologies, Inc.**, Milpitas, Calif.

[21] Appl. No.: **09/013,743**

[22] Filed: **Jan. 27, 1998**

[51] **Int. Cl.**^{7} **G06F 11/30**

[52] **U.S. Cl.** **713/201**; 709/223

[58] **Field of Search** 713/200, 201; 709/220–226

[56] **References Cited**

U.S. PATENT DOCUMENTS

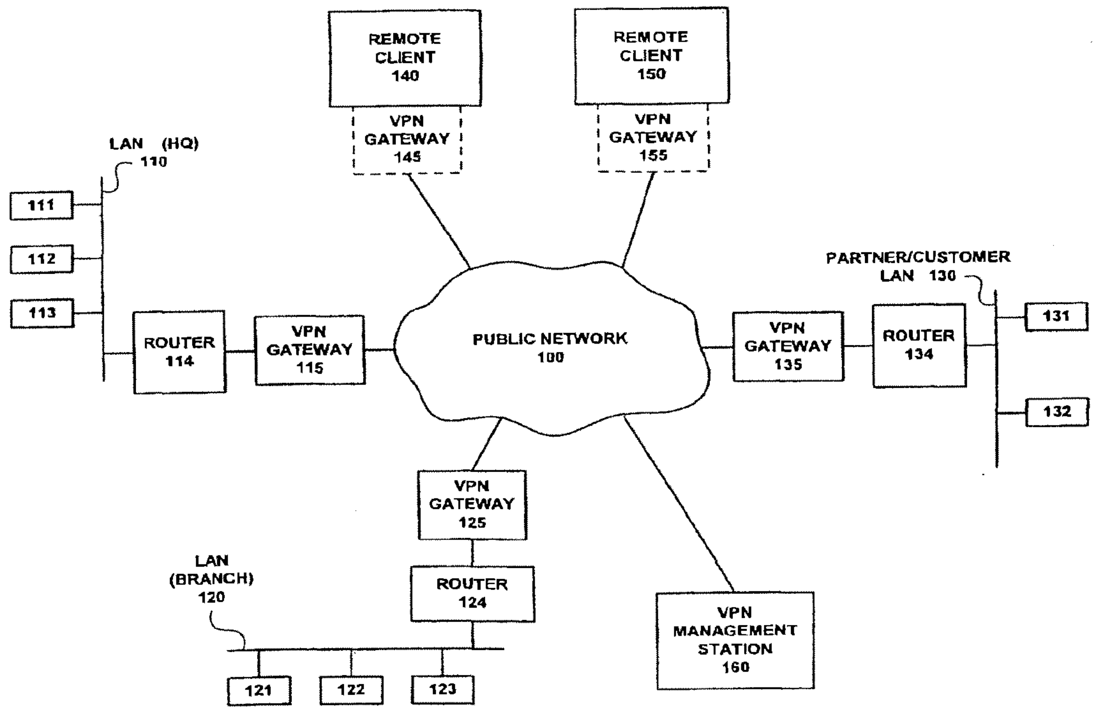
5,339,356	8/1994	Ishii	379/234
5,432,783	7/1995	Ahmed et al.	370/60.1
5,490,212	2/1996	Lautenschlager	379/225
5,504,921	4/1996	Dev et al.	395/800
5,550,816	8/1996	Hardwick et al.	370/60
5,623,601	4/1997	Vu	395/187.01
5,659,542	8/1997	Bell et al.	370/496
5,742,762	4/1998	Scholl et al.	395/200.3
5,768,271	6/1998	Seid et al.	370/389
5,799,016	8/1998	Onweller	370/401
5,898,830	4/1999	Wessinger, Jr. et al.	395/187.01

Primary Examiner—Ayaz R. Sheikh
Assistant Examiner—Jigar Pancholi
Attorney, Agent, or Firm—Park & Vaughan LLP

[57] **ABSTRACT**

The present invention provides a method and an apparatus for managing a virtual private network operating over a public data network. This public data network has been augmented to include a plurality of virtual private network gateways so that communications across the virtual private network are channeled through the virtual private network gateways. One embodiment of the present invention includes a system that operates by receiving a command specifying an operation on the virtual private network. The system determines which virtual private network gateways are affected by the command. The system then automatically translates the command into configuration parameters for virtual private network gateways affected by the command. These configuration parameters specifying how the virtual private network gateways handle communications between specific groups of addresses on the public data network. The system then transmits the configuration parameters to the virtual private network gateways affected by the command, so that the virtual private network gateways are configured to implement the command.

22 Claims, 11 Drawing Sheets



US006330562B1

(12) **United States Patent**
Boden et al.

(10) **Patent No.: US 6,330,562 B1**
(45) **Date of Patent: Dec. 11, 2001**

(54) SYSTEM AND METHOD FOR MANAGING SECURITY OBJECTS

(75) Inventors: **Edward B. Boden**; **Franklin A. Gruber**, both of Vestal; **Mark J. Melville**, Endwell; **Frank V. Paxhia**, Binghamton; **Michael D. Williams**, Owego, all of NY (US)

(73) Assignee: **International Business Machines Corporation**, Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/239,693**

(22) Filed: **Jan. 29, 1999**

(51) **Int. Cl.**^{7} **G06F 17/30**

(52) **U.S. Cl.** 707/10; 707/9; 709/220; 713/200; 713/201; 713/202

(58) **Field of Search** 707/9–10; 713/200–202; 709/220

(56) **References Cited**

U.S. PATENT DOCUMENTS

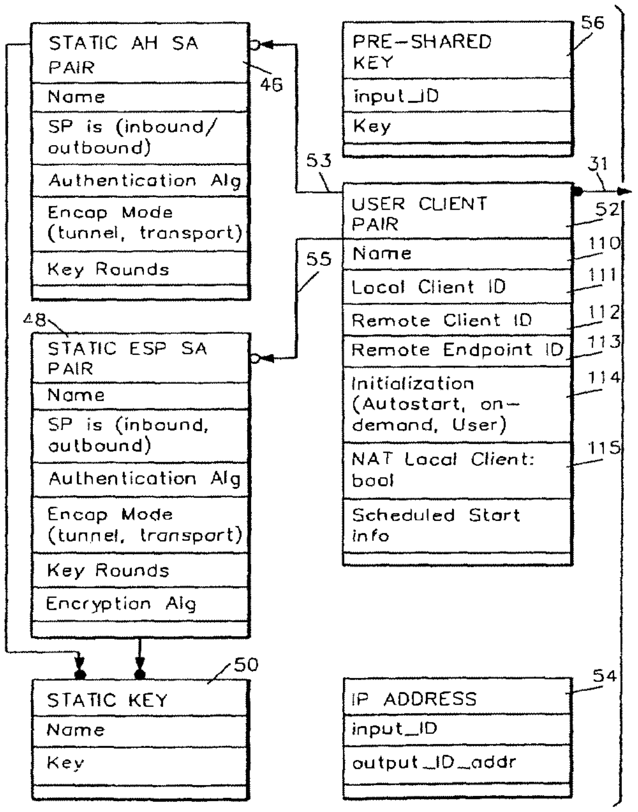
5,051,982	9/1991	Brown et al.	370/58.2
5,621,727	4/1997	Vaudreuil	370/60
5,764,909	6/1998	Nishimura	395/200.53
5,768,271	6/1998	Seid et al.	370/389
5,835,726	11/1998	Shwed et al.	395/200.59
5,842,043	11/1998	Nishimura	395/856

Primary Examiner—Hosain T. Alam
Assistant Examiner—Camy Truong
(74) *Attorney, Agent, or Firm*—Shelley M Beckstrand

(57) **ABSTRACT**

A data model for abstracting customer-defined VPN security policy information. By employing this model, a VPN node (computer system existing in a Virtual Private Network) can gather policy configuration information for itself through a GUY, or some distributed policy source, store this information in a system-defined database, and use this information to dynamically negotiate, create, delete, and maintain secure connections at the IP level with other VPN nodes.

15 Claims, 6 Drawing Sheets



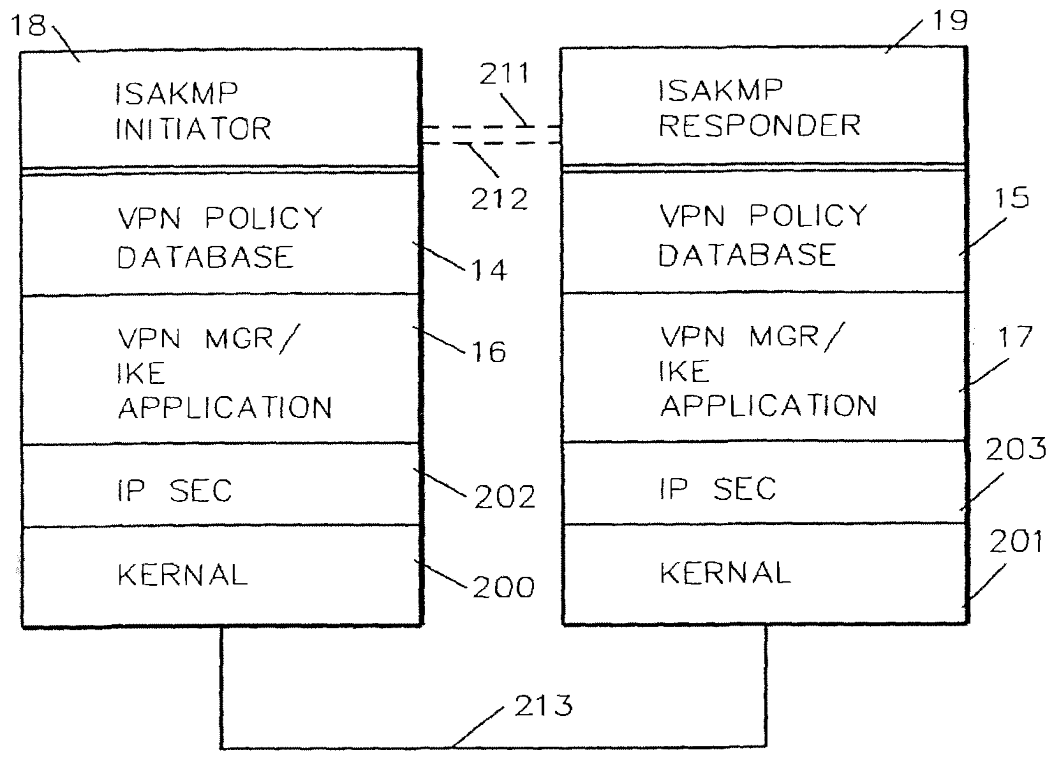

18
ISAKMP INITIATOR
VPN POLICY DATABASE
VPN MGR/ IKE APPLICATION
IP SEC
KERNAL
14
16
202
200
211
212
19
ISAKMP RESPONDER
VPN POLICY DATABASE
VPN MGR/ IKE APPLICATION
IP SEC
KERNAL
15
17
203
201
213

FIG. 1

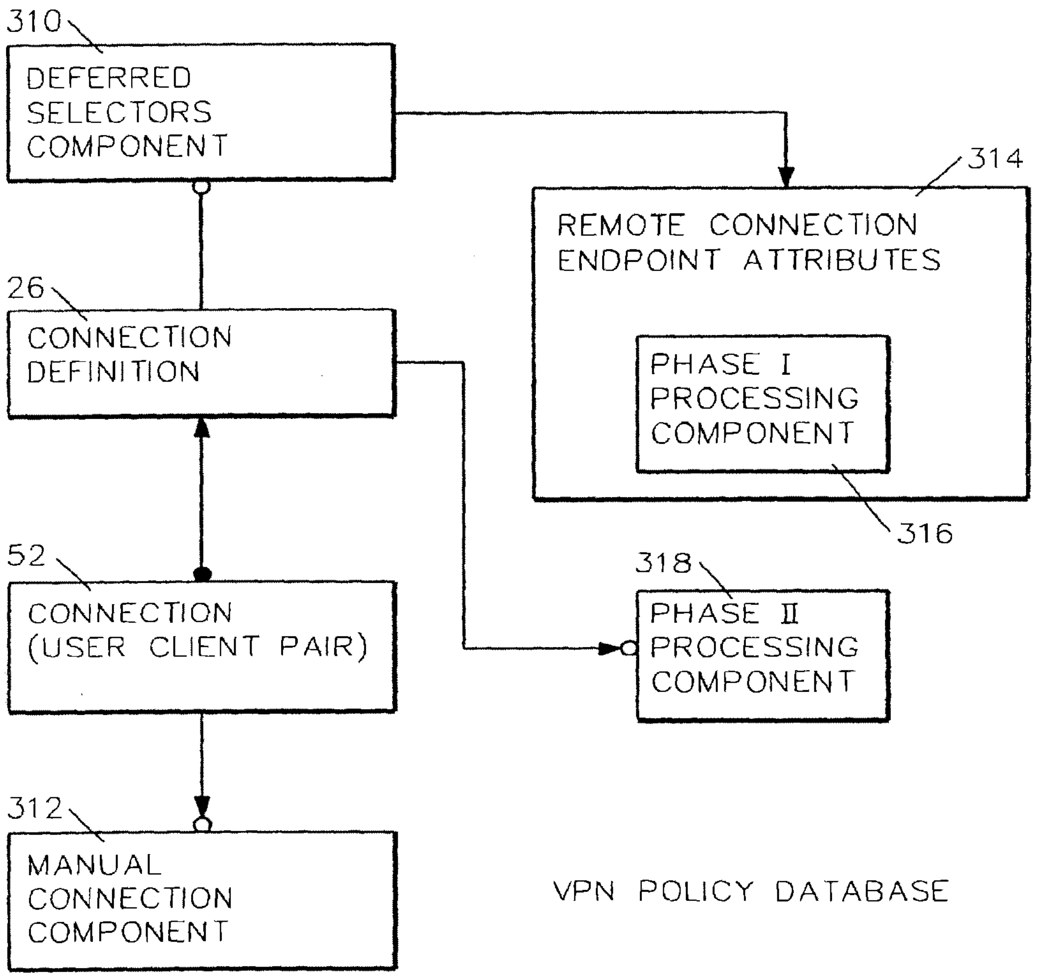

310
DEFERRED SELECTORS COMPONENT
314
REMOTE CONNECTION ENDPOINT ATTRIBUTES
PHASE I PROCESSING COMPONENT
316
26
CONNECTION DEFINITION
52
CONNECTION (USER CLIENT PAIR)
318
PHASE II PROCESSING COMPONENT
312
MANUAL CONNECTION COMPONENT
VPN POLICY DATABASE

FIG. 2

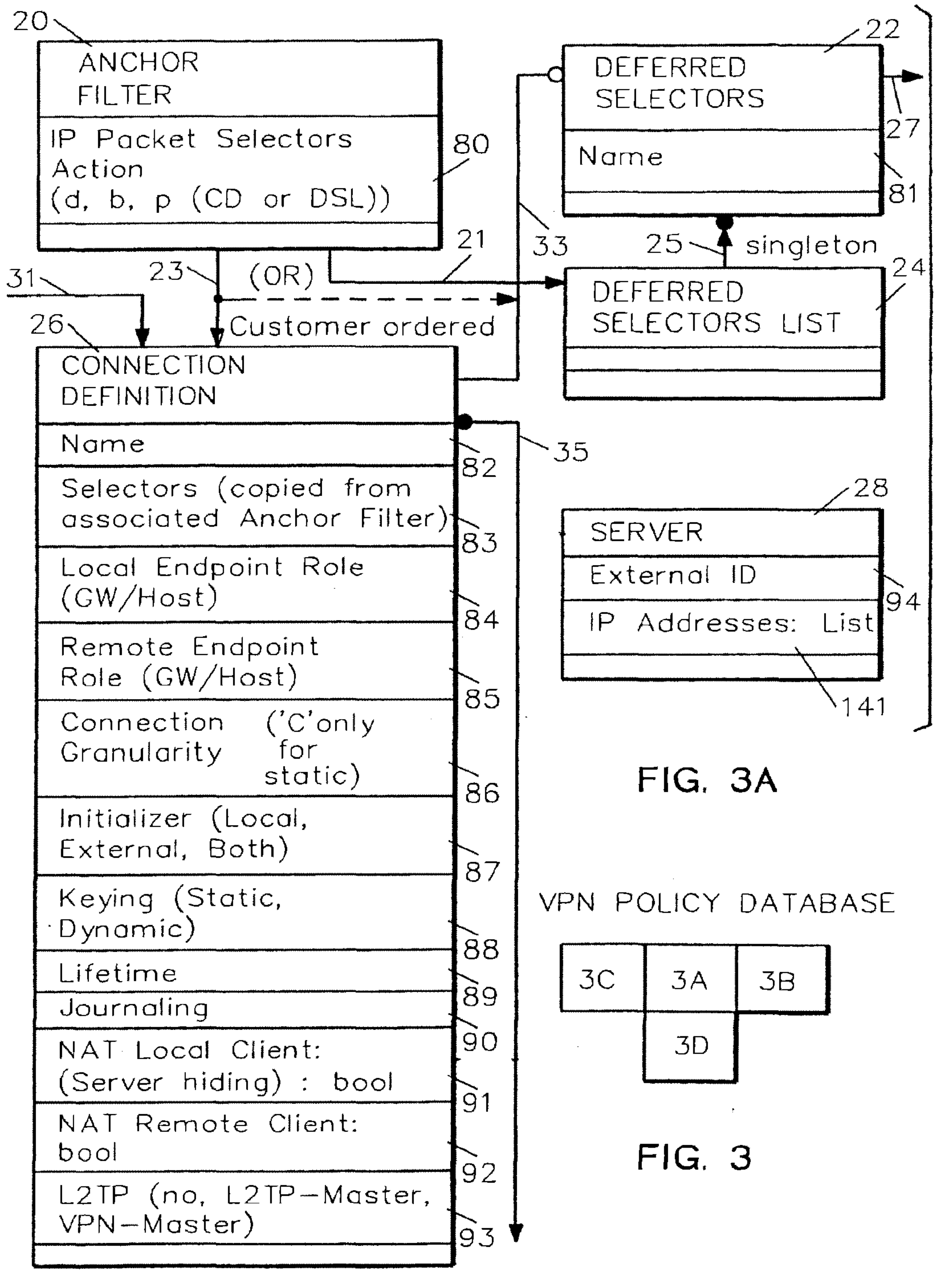

20
ANCHOR FILTER
IP Packet Selectors
Action
(d, b, p (CD or DSL))
80
21
23
(OR)
31
Customer ordered
26
CONNECTION DEFINITION
Name
82
Selectors (copied from associated Anchor Filter)
83
Local Endpoint Role (GW/Host)
84
Remote Endpoint Role (GW/Host)
85
Connection Granularity ('C'only for static)
86
Initializer (Local, External, Both)
87
Keying (Static, Dynamic)
88
Lifetime
89
Journaling
90
NAT Local Client: (Server hiding) : bool
91
NAT Remote Client: bool
92
L2TP (no, L2TP–Master, VPN–Master)
93
22
DEFERRED SELECTORS
Name
27
81
33
25
singleton
24
DEFERRED SELECTORS LIST
35
28
SERVER
External ID
94
IP Addresses: List
141

FIG. 3A

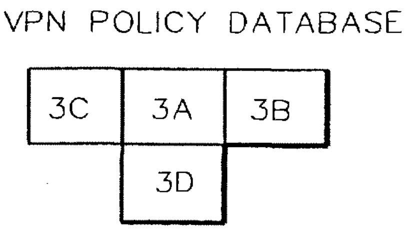

VPN POLICY DATABASE
3C
3A
3B
3D

FIG. 3

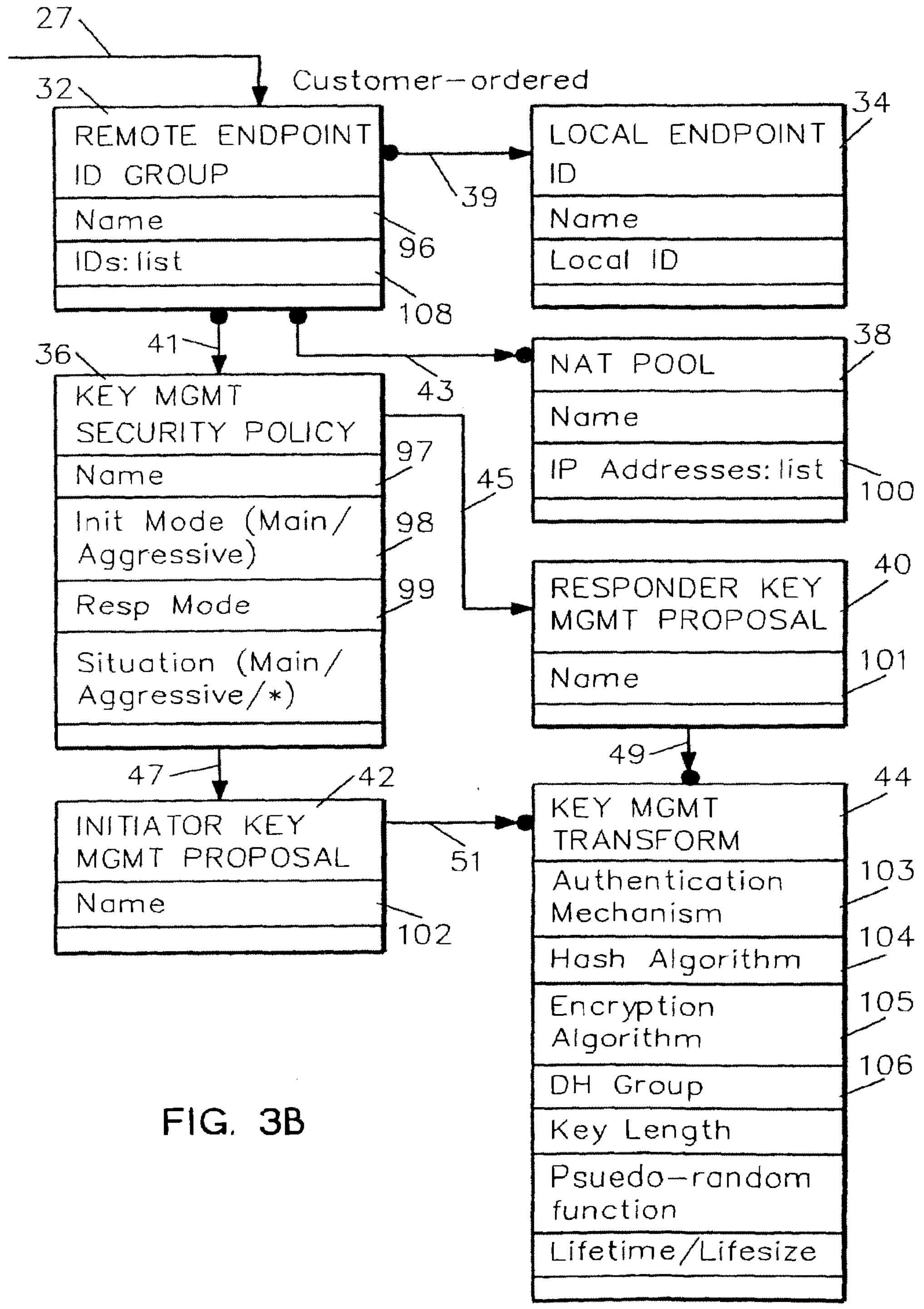

27
Customer-ordered
32
REMOTE ENDPOINT ID GROUP
Name
IDs: list
96
108
39
34
LOCAL ENDPOINT ID
Name
Local ID
41
43
36
38
KEY MGMT SECURITY POLICY
Name
Init Mode (Main/ Aggressive)
Resp Mode
Situation (Main/ Aggressive/*)
97
98
99
45
NAT POOL
Name
IP Addresses: list
100
40
RESPONDER KEY MGMT PROPOSAL
Name
101
49
47
42
INITIATOR KEY MGMT PROPOSAL
Name
102
51
44
KEY MGMT TRANSFORM
Authentication Mechanism
Hash Algorithm
Encryption Algorithm
DH Group
Key Length
Psuedo-random function
Lifetime/Lifesize
103
104
105
106

FIG. 3B

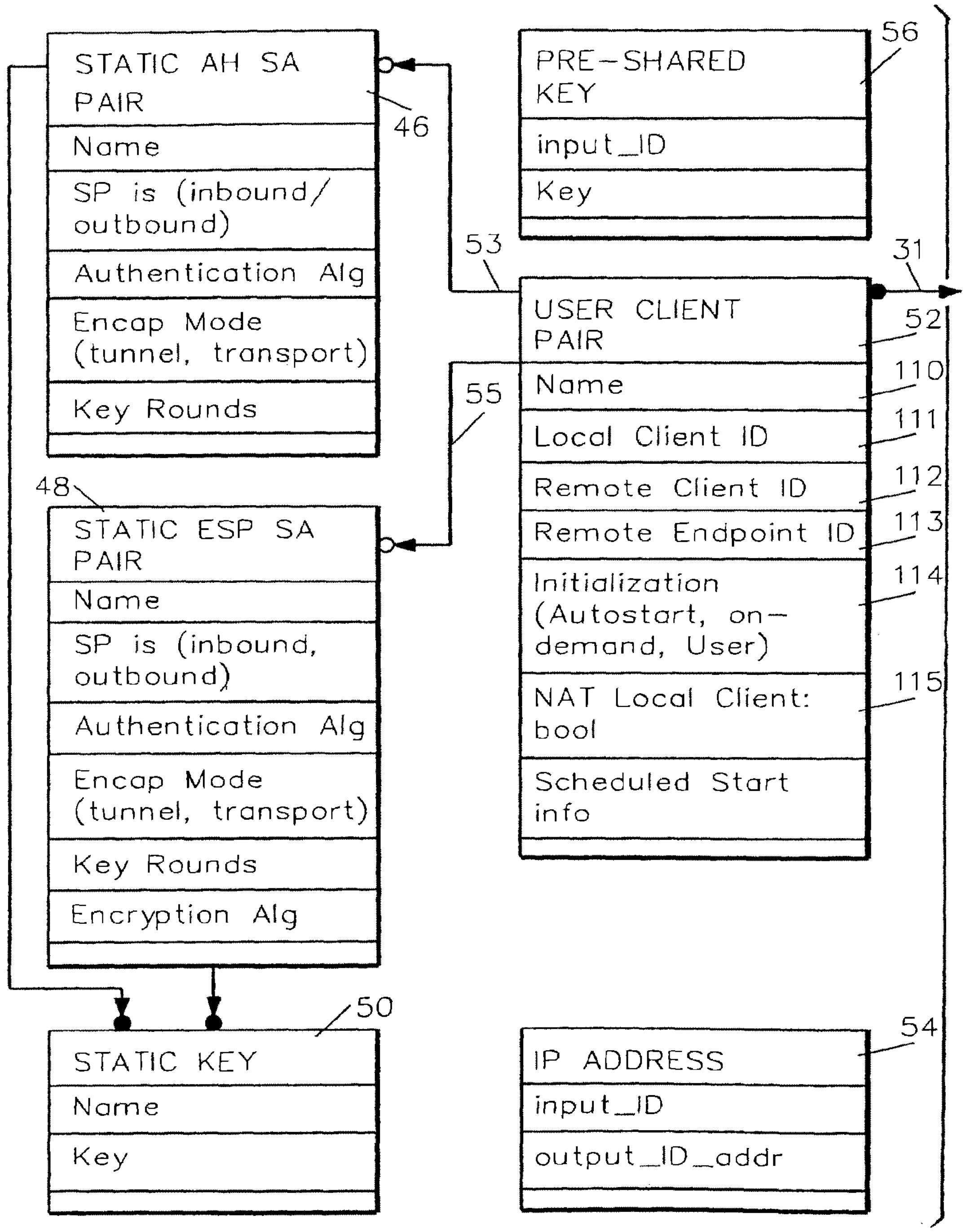

STATIC AH SA PAIR
Name
SP is (inbound/ outbound)
Authentication Alg
Encap Mode (tunnel, transport)
Key Rounds
46
48
STATIC ESP SA PAIR
Name
SP is (inbound, outbound)
Authentication Alg
Encap Mode (tunnel, transport)
Key Rounds
Encryption Alg
50
STATIC KEY
Name
Key
56
PRE-SHARED KEY
input_ID
Key
53
31
USER CLIENT PAIR
52
Name
110
Local Client ID
111
Remote Client ID
112
Remote Endpoint ID
113
Initialization (Autostart, on-demand, User)
114
NAT Local Client: bool
115
Scheduled Start info
55
54
IP ADDRESS
input_ID
output_ID_addr

FIG. 3C

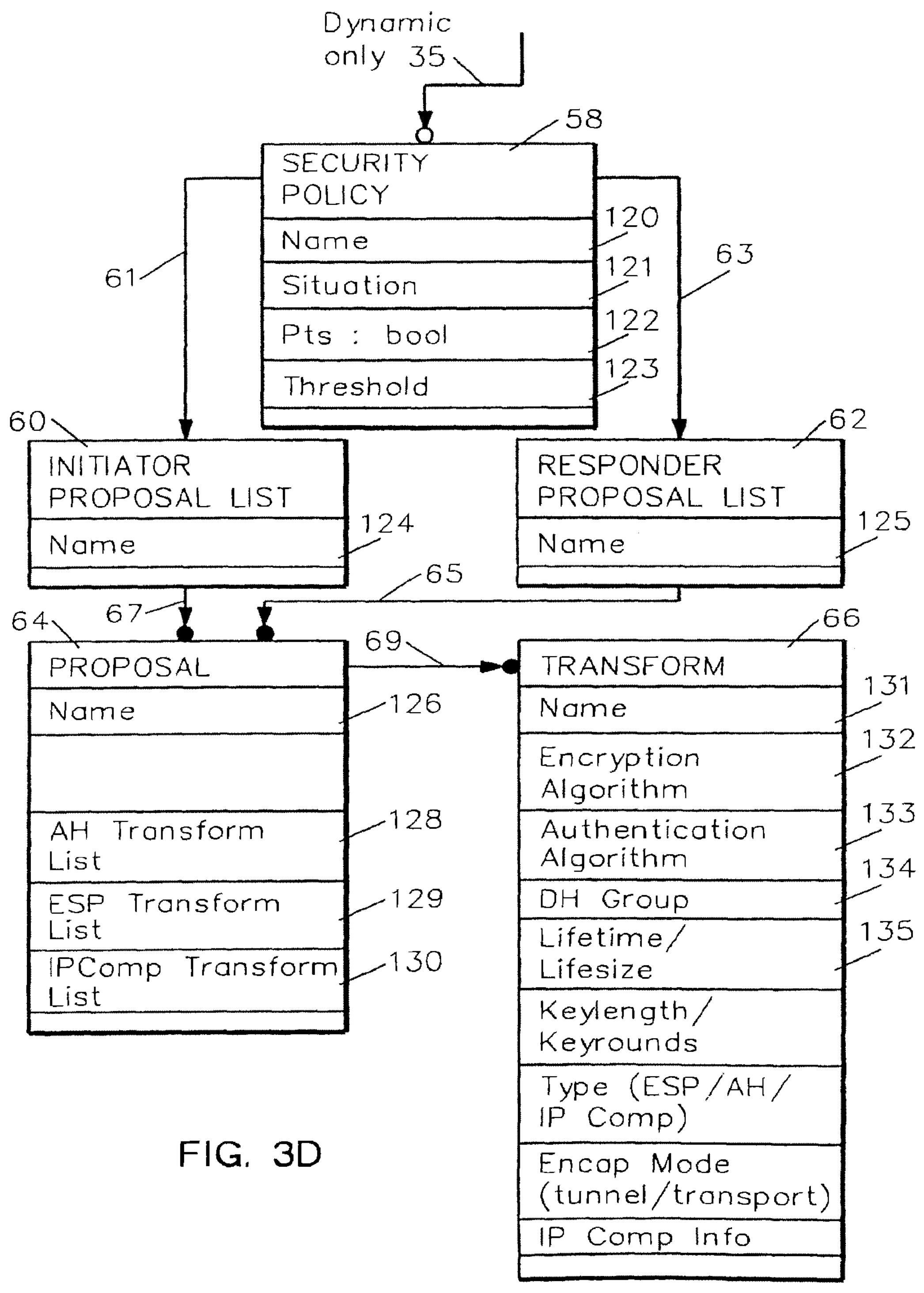

Dynamic only 35
58
SECURITY POLICY
Name
Situation
Pts : bool
Threshold
120
121
122
123
61
63
60
INITIATOR PROPOSAL LIST
Name
124
62
RESPONDER PROPOSAL LIST
Name
125
65
64
67
PROPOSAL
Name
AH Transform List
ESP Transform List
IPComp Transform List
126
128
129
130
69
66
TRANSFORM
Name
Encryption Algorithm
Authentication Algorithm
DH Group
Lifetime/ Lifesize
Keylength/ Keyrounds
Type (ESP/AH/ IP Comp)
Encap Mode (tunnel/transport)
IP Comp Info
131
132
133
134
135

FIG. 3D

4. Checking those attributes of the client ID pair for which the granularity 86 indicates 'single' or 'client' against the local endpoint role 84 and remote endpoint role 85, as shown in Table 2; and
5. Using the client ID pair and the connection definition selectors 83 to generate the connection client IDs according to the connection granularity; 'client' or 'single' indicate the value comes from the client ID pair, 'filter' indicates the value comes from the connection definition selectors 83.

The resulting connection client IDs are then used to create the filter rules (SPD entry) for this connection in the kernal.

Advantages over the Prior Art

It is a further advantage of the invention that there is provided a system and method for creating, maintaining, deleting and retrieving VPN policy objects.

It is a further advantage of the invention that there is provided a system and method for enabling acceptance of previously unknown IDci/IDcr values from a remote system.

It is a further advantage of the invention that there is provided a system and method enabling dynamic generation, load, and management of multiple IPSec filter rules.

It is a further advantage of the invention that there is provided a system and method enabling ISAKMP phase II driven phase I connections.

It is a further advantage of the invention that there is provided a system and method enabling handling of remote initiating hosts with dynamically assigned IP addresses with differing security policy requirements.

It is a further advantage of the invention that there is provided flexibility in policy definition in the areas of dynamically-assigned IP addresses, remotely-defined ISAKMP client IDs (IDci/IDcr), and separation of ISAKMP Phase I (key management) policy information from ISAKMP Phase II (data management) policy information.

It is a further advantage of the invention that there is provided a data model for representing and abstracting IPSec/ISAKMP-based VPN configuration information for an IPSec-capable computer system in a virtual private network that (1) allows for each customer-generated customer-ordered security policy database (SPD) entry, multiple VPN connections to be dynamically established (these connections may or may not have been previously defined); (2) allows for a data-security-policy-driven approach to rekeying (via IKE) where (a) the key management connection (i.e. the secure connection used to exchange keying material for the data connections) is created and maintained by security policy and on an on-demand basis by data connection activity, and (b) the key connection security policy is determined solely by the identity of the remote connection endpoint; (3) allows for dynamically establishing VPN connections with different security policies and other attributes, based solely on an unfixed IP address (e.g. a user ID)—these connections may or may not have been previously defined. This aspect is used for supporting systems with dynamically-assigned IP addresses that wish to establish a VPN connection with the local system.

Alternative Embodiments

It will be appreciated that, although specific embodiments of the invention have been described herein for purposes of illustration, various modifications may be made without departing from the spirit and scope of the invention. In particular, it is within the scope of the invention to provide a program storage or memory device such as a solid or fluid transmission medium, magnetic or optical wire, tape or disc, or the like, for storing signals readable by a machine for controlling the operation of a computer according to the method of the invention and/or to structure its components in accordance with the system of the invention.

Accordingly, the scope of protection of this invention is limited only by the following claims and their equivalents.

We claim:

1. A policy database system for managing security objects, comprising:
 a deferred selectors component;
 a connection definition;
 a user client pair;
 a manual connection component;
 a remote connection endpoint attributes component including a phase I processing component; and
 a phase II processing component;
 said connection definition having a zero or one reference relationship with said deferred selectors component, a zero or more reference relationship with said user client pair, and a zero or one reference relationship with said phase II processing component; said user client pair further having a zero or one reference relationship with said manual connection component; and said deferred selectors component having a one and only one reference relationship with said remote connection attributes component.

2. The policy database system of claim 1 further for enabling acceptance at a responder node of a previously unknown client ID pair from an initiator node, said connection definition comprising indicia for determining if said unknown client pair is acceptable to said responder node and said phase II processing component comprising a policy for negotiating said unknown client ID pair.

3. The policy database system of claim 1 further for enabling dynamic generation, loading and management of multiple connection filters, said connection definition being selectable selectively by said user client pair or a client ID pair received from a remote initiator node for identifying pertinent granularity attributes defining the subset of datagrams that can be associated with any one connection instantiated from said connection definition.

4. The policy database system of claim 1 further for enabling ISAKMP phase II driven phase I connections, said remote connection endpoints attributes further comprising a remote endpoint identifier and a reference pointer for associating said remote endpoint identifier with a phase I negotiation policy in said phase I processing component.

5. The policy database system of claim 1 further for enabling secure connection by a responder node to a remote initiating host with dynamically assigned IP address, further comprising:
 an anchor filter for defining datagrams that may be associated with remote hosts using dynamically assigned IP addresses;
 said deferred selectors component further providing a one to many mapping from said anchor filter to said connection definitions.

6. A method for managing a policy database, said database including a deferred selectors component, a connection definition, a user client pair, a manual connection component, a remote connection endpoint attributes component including a phase I processing component; and a phase II processing component, comprising the steps of:

maintaining a zero or one reference relationship of said connection definition with said deferred selectors component;

maintaining a zero or more reference relationship of said connection definition with said user client pair;

maintaining a zero or one reference relationship of said connection definition with said phase II processing component;

maintaining a zero or one reference relationship of said user client pair with said manual connection component; and

maintaining a one and only one reference relationship of said deferred selectors component with said remote connection attributes component.

7. The method of claim 6, further for enabling acceptance at a responder node of a previously unknown client ID pair from an initiator node, comprising the further steps of:

determining from connection definition indicia if said unknown client pair is acceptable to said responder node, and if so

obtaining from said phase II processing component a policy for negotiating said unknown client ID pair.

8. The method of claim 6, further for enabling dynamic generation, load and management of multiple connection filters, comprising the further steps of:

obtaining from a said connection definition, selectively selected by said user client pair or a client ID pair received from a remote initiator node, granularity attributes defining the subset of datagrams that can be associated with any one connection instantiated from said connection definition.

9. The method of claim 6, further for enabling ISAKMP phase II driven phase I connections, comprising the further steps of:

associating a remote endpoint identifier in said remote connection endpoints attributes with a phase I negotiation policy in said phase I processing component.

10. The method of claim 6, further for enabling secure connection by a responder node to a remote initiating host with dynamically assigned IP address, further comprising the steps of:

providing an anchor filter for defining datagrams that may be associated with remote hosts using dynamically assigned IP addresses; and

said deferred selectors component further providing a one to many mapping from said anchor filter to said connection definitions.

11. A program storage device readable by a machine, tangibly embodying a program of instructions executable by a machine to perform method steps for managing a policy database, said database including a deferred selectors component, a connection definition, a user client pair, a manual connection component, a remote connection endpoint attributes component including a phase I processing component; and a phase II processing component, said method steps comprising:

maintaining a zero or one reference relationship of said connection definition with said deferred selectors component;

maintaining a zero or more reference relationship of said connection definition with said user client pair;

maintaining a zero or one reference relationship of said connection definition with said phase II processing component;

maintaining a zero or one reference relationship of said user client pair with said manual connection component; and

maintaining a one and only one reference relationship of said deferred selectors component with said remote connection attributes component.

12. An article of manufacture comprising:

a computer useable medium having computer readable program code means embodied therein for managing a policy database, said database including a deferred selectors component, a connection definition, a user client pair, a manual connection component, a remote connection endpoint attributes component including a phase I processing component; and a phase II processing component, the computer readable program means in said article of manufacture comprising:

computer readable program code means for causing a computer to effect maintaining a zero or one reference relationship of said connection definition with said deferred selectors component;

computer readable program code means for causing a computer to effect maintaining a zero or more reference relationship of said connection definition with said user client pair;

computer readable program code means for causing a computer to effect maintaining a zero or one reference relationship of said connection definition with said phase II processing component;

computer readable program code means for causing a computer to effect maintaining a zero or one reference relationship of said user client pair with said manual connection component; and

computer readable program code means for causing a computer to effect maintaining a one and only one reference relationship of said deferred selectors component with said remote connection attributes component.

13. A policy database system for managing security objects and enabling ISAKMP phase II driven phase I connections, comprising:

a deferred selectors component;

a connection definition;

a user client pair;

a manual connection component;

a remote connection endpoint attributes component including a phase I processing component; and

a phase II processing component;

said connection definition having a zero or one reference relationship with said deferred selectors component, a zero or more reference relationship with said user client pair, and a zero or one reference relationship with said phase II processing component; said user client pair further having a zero or one reference relationship with said manual connection component; and said deferred selectors component having a one and only one reference relationship with said remote connection attributes component; and

said remote connection endpoints attributes further comprising a remote endpoint identifier and a reference pointer for associating said remote endpoint identifier with a phase I negotiation policy in said phase I processing component.

14. A method for managing a policy database and enabling ISAKMP phase II driven phase I connections, said database including a deferred selectors component, a connection definition, a user client pair, a manual connection component, a remote connection endpoint attributes component including a phase I processing component; and a phase II processing component, comprising the steps of:

maintaining a zero or one reference relationship of said connection definition with said deferred selectors component;

maintaining a zero or more reference relationship of said connection definition with said user client pair;

maintaining a zero or one reference relationship of said connection definition with said phase II processing component;

maintaining a zero or one reference relationship of said user client pair with said manual connection component;

maintaining a one and only one reference relationship of said deferred selectors component with said remote connection attributes component; and

associating a remote endpoint identifier in said remote connection endpoints attributes with a phase I negotiation policy in said phase I processing component.

15. A program storage device readable by a machine, tangibly embodying a program of instructions executable by a machine to perform method steps for managing a policy database and enabling ISAKMP phase II driven phase I connections, said database including a deferred selectors component, a connection definition, a user client pair, a manual connection component, a remote connection endpoint attributes component including a phase I processing component; and a phase II processing component, said method steps comprising:

maintaining a zero or one reference relationship of said connection definition with said deferred selectors component;

maintaining a zero or more reference relationship of said connection definition with said user client pair;

maintaining a zero or one reference relationship of said connection definition with said phase II processing component;

maintaining a zero or one reference relationship of said user client pair with said manual connection component;

maintaining a one and only one reference relationship of said deferred selectors component with said remote connection attributes component; and

associating a remote endpoint identifier in said remote connection endpoints attributes with a phase I negotiation policy in said phase I processing component.

* * * * *

US006081900A

United States Patent [19]

Subramaniam et al.

[11] **Patent Number: 6,081,900**

[45] **Date of Patent: Jun. 27, 2000**

[54] **SECURE INTRANET ACCESS**

[75] Inventors: **Anand Subramaniam**, San Jose, Calif.; **Hashem M. Ebrahimi**, Salt Lake City, Utah

[73] Assignee: **Novell, Inc.**, Provo, Utah

[21] Appl. No.: **09/268,795**

[22] Filed: **Mar. 16, 1999**

[51] **Int. Cl.^{7}** **H04L 9/32; G06F 13/38**

[52] **U.S. Cl.** 713/201; 713/153; 707/10; 707/513; 709/230; 709/245

[58] **Field of Search** 713/151, 153, 713/155, 160, 162, 201; 709/217, 218, 214, 230, 238, 245; 707/10, 501, 513

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,522,041	5/1996	Murakami et al.	709/203
5,550,984	8/1996	Gelb	395/200.17
5,553,239	9/1996	Heath et al.	395/187.01
5,673,322	9/1997	Pepe et al.	380/49
5,706,434	1/1998	Kremen et al.	709/218
5,727,145	3/1998	Nessett et al.	713/200
5,745,360	4/1998	Leone et al.	707/513
5,752,022	5/1998	Chiu et al.	707/10
5,757,924	5/1998	Friedman et al.	713/162
5,761,683	6/1998	Logan et al.	707/513
5,768,271	6/1998	Seid et al.	370/389
5,805,803	9/1998	Birrell et al.	713/201
5,825,890	10/1998	Elgamal et al.	380/49
5,890,171	3/1999	Blumer et al.	707/501
5,913,025	6/1999	Higley et al.	713/201
5,991,810	11/1999	Shapiro et al.	709/229

OTHER PUBLICATIONS

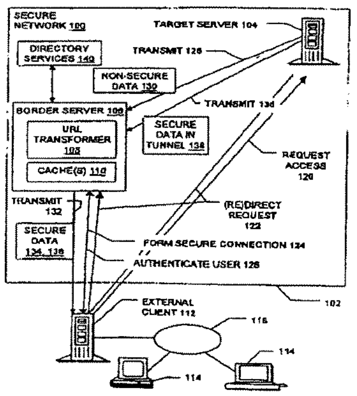
Andrew S. Tanenbaum, *Computer Networks*, 3d. Ed. (1996), pp. 28–39, 396–417, 601–621, 681–695.
"Securing Communications on the Intranet and Over the Internet", Netscape Communications Corporation (Jul. 1996), pp. 1–12.
"WebSTAR/SSL Security Toolkit: Troubleshooting", no later than Aug. 4, 1998, pp. 1–2.
Paul Ferguson, "What is a VPN?", Apr. 1998, pp. 1–22.
Virtual Private Networking: An Overview, no later than Jun. 25, 1998, pp. 1–20.
"How the proxy works", Jul. 22, 1998, pp. 1–2.
Microsoft Proxy Server Marketing Bulletin, 1998, pp. 1–8.
Microsoft Proxy Server What's New, 1998, pp. 1–4.
"CSM Proxy Server White Paper", 1995–1998, pp. 1–13.
"CSM Proxy Server, the Ultimate Gateway to the Internet!", no later than Aug. 4, 1998, pp. 1–5.
Qun Zhong et al., "Security Control for COTS Components", Jun. 1998 IEEE Computer, pp. 67–73.
"r^3 CypherClient Information", 1998, pp. 1–4.
NetSafe V3.0 The Firewall Solution from Siemens Nixdorf, May 31, 1997, pp. 1–3.
Submission: HTTPS v1.0 Package for OmmiWeb 3.x (Rhapsody), Jan. 24, 1998, pp. 1–3.
OWF basics, Mar. 31, 1997, pp. 1–2.
"info needed to write https proxy", Apr. 11, 1996.
Webroute 1.3.0, May 28, 1997, pp. 1–2.
Werner Feibel, *Novell's Complete Encyclopedia of Networking*, 1995, pp. 625–630.
Bruce Schneier, *Applied Cryptography*, 1994, pp. 436–437.
"A New Management and Security Architecture for Extranets", 1996–1999, pp. 1–14.
Netscape Proxy Server Administrator's Guide, 1997, Contents, Introduction, and Chapters 1–17.

Primary Examiner—Gilberto Barrón, Jr.
Attorney, Agent, or Firm—Computer Law++

[57] **ABSTRACT**

Methods, signals, devices, and systems are provided for secure access to a network from an external client. Requests for access to confidential data may be redirected from a target server to a border server, after which a secure sockets layer connection between the border server and the external client carries user authentication information. After the user is authenticated to the network, requests may be redirected back to the original target server. Web pages sent from the target server to the external client are scanned for non-secure URLs such as those containing "http://" and modified to make them secure. The target server and the border server utilize various combinations of secure and non-secure caches. Although tunneling may be used, the extensive configuration management burdens imposed by virtual private networks are not required.

30 Claims, 4 Drawing Sheets



IW 7112490

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

March 11, 2008

THIS IS TO CERTIFY THAT ANNEXED IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF THE FILE WRAPPER AND CONTENTS OF:

APPLICATION NUMBER: *09/504,783*
FILING DATE: *February 15, 2000*
PATENT NUMBER: *6,502,135*
ISSUE DATE: *December 31, 2002*

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

M. Tarver

M. TARVER
Certifying Officer

UNITED STATES PATENT AND TRADEMARK OFFICE DEPARTMENT OF COMMERCE

PART (7) OF (7) PART(S)

PATENT COOPERATION TREATY

PCT

INTERNATIONAL SEARCH REPORT

(PCT Article 18 and Rules 43 and 44)

Applicant's or agent's file reference 00479.00028	**FOR FURTHER ACTION** see Notification of Transmittal of International Search Report (Form PCT/ISA/220) as well as, where applicable, item 5 below.	
International application No. PCT/US 01/13260	International filing date *(day/month/year)* 25/04/2001	(Earliest) Priority Date *(day/month/year)* 30/10/1998
Applicant SCIENCE APPLICATIONS INTERNATIONAL CORPORATION		

This International Search Report has been prepared by this International Searching Authority and is transmitted to the applicant according to Article 18. A copy is being transmitted to the International Bureau.

This International Search Report consists of a total of 6 sheets.

[✓ X] It is also accompanied by a copy of each prior art document cited in this report.

1. **Basis of the report**

   a. With regard to the **language,** the international search was carried out on the basis of the international application in the language in which it was filed, unless otherwise indicated under this item.

   ☐ the international search was carried out on the basis of a translation of the international application furnished to this Authority (Rule 23.1(b)).

   b. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, the international search was carried out on the basis of the sequence listing :

   ☐ contained in the international application in written form.

   ☐ filed together with the international application in computer readable form.

   ☐ furnished subsequently to this Authority in written form.

   ☐ furnished subsequently to this Authority in computer readble form.

   ☐ the statement that the subsequently furnished written sequence listing does not go beyond the disclosure in the international application as filed has been furnished.

   ☐ the statement that the information recorded in computer readable form is identical to the written sequence listing has been furnished

2. ☐ **Certain claims were found unsearchable** (See Box I).

3. [✓ X] **Unity of invention is lacking** (see Box II).

4. With regard to the **title,**

   ☐ the text is approved as submitted by the applicant.

   [✓ X] the text has been established by this Authority to read as follows:

   SECURE DOMAIN NAME SERVICE

5. With regard to the **abstract,**

   [✓ X] the text is approved as submitted by the applicant.

   ☐ the text has been established, according to Rule 38.2(b), by this Authority as it appears in Box III. The applicant may, within one month from the date of mailing of this international search report, submit comments to this Authority.

6. The figure of the **drawings** to be published with the abstract is Figure No. 26

   ☐ as suggested by the applicant.

   [✓ X] because the applicant failed to suggest a figure.

   ☐ because this figure better characterizes the invention.

   ☐ None of the figures.

Form PCT/ISA/210 (first sheet) (July 1998)

IN...RNATIONAL SEARCH REPORT

International Application No
PCT/US 01/13260

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L29/12 H04L29/06 G06F17/60

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04L G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, PAJ, IBM-TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	DONALD E. EASTLAKE 3RD: "<draft-ietf-dnssec-secext2-05.txt> Domain Name System Security Extensions" INTERNET DRAFT, 'Online! April 1998 (1998-04), XP002199931 Retrieved from the Internet: <URL:ftp://ftp.inet.no/pub/ietf/internet-drafts/draft-ietf-dnssec-secext2-05.txt> 'retrieved on 2002-05-23! 1. Overview of the contents 2.3 Data origin authemtication and integrity 2.4 DNS transaction and request authentication --- -/--	1,4-6

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search 12 August 2002	Date of mailing of the international search report 23. 08. 2002
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Bertolissi, E

3

Form PCT/ISA/210 (second sheet) (July 1992)

IN'~RNATIONAL SEARCH REPORT

International Application No
PCT/US 01/13260

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
Y	CHAPMAN D.B.; ZWICKY E.D.: " Building Internet Firewalls" O'REILLY, November 1995 (1995-11), XP002199932 pag 278-296 pag 351-375 ---	1,4-6
P,X	DE 199 24 575 A (SUN MICROSYSTEMS INC) 2 December 1999 (1999-12-02) abstract column 2, line 48 -column 3, line 6 ---	1,4,5,7
A	P. SRISURESH, G. TSIRTSIS, P. AKKIRAJU, A. HEFFERNAN: "<draft-ietf-nat-dns-alg-00.txt> DNS extensions to Network Address Translators (DNS_ALG)" INTERNET DRAFT, 'Online! July 1998 (1998-07), XP002199933 Retrieved from the Internet: <URL:ftp://ftp.inet.no/pub/ietf/internet-drafts/draft-ietf-nat-dns-alg-00.txt> 'retrieved on 2002-05-23! 1. Introduction 2. Requirement for DNS extensions fig 3 5.3 Incoming name lookup queries 8. Security considerations ---	1-12
A	EP 0 838 930 A (DIGITAL EQUIPMENT CORP) 29 April 1998 (1998-04-29) abstract column 9, line 11 -column 10, line 34 ---	1-12
X	JAMES E. BELLAIRE: "Subject: New Statement of Rules - Naming Internet Domains " INTERNET NEWSGROUP, 'Online! 30 July 1995 (1995-07-30), XP002209580 comp.dcom.telecom Retrieved from the Internet: <URL:http://groups.google.com/> 'retrieved on 2002-08-12! page 1, paragraph 8 page 2, paragraph 4 ---	13,15
A	CLARK D: "US CALLS FOR PRIVATE DOMAIN-NAME SYSTEM" COMPUTER, IEEE COMPUTER SOCIETY, LONG BEACH., CA, US, US, vol. 31, no. 8, 1 August 1998 (1998-08-01), pages 22-25, XP000780513 ISSN: 0018-9162 page 22 --- -/--	13-16

3

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

International Application No
PCT/US 01/13260

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	BEQUAI A: "Balancing Legal Concerns Over Crime and Security in Cyberspace" COMPUTERS & SECURITY. INTERNATIONAL JOURNAL DEVOTED TO THE STUDY OF TECHNICAL AND FINANCIAL ASPECTS OF COMPUTER SECURITY, ELSEVIER SCIENCE PUBLISHERS. AMSTERDAM, NL, vol. 17, no. 4, 1998, pages 293-298, XP004129224 ISSN: 0167-4048 pag 296-297 Lanham Act ---	13-16
A	RICH WINKEL : "CAQ: NETWORKING WITH SPOOKS: THE NET & THE CONTROL OF INFORMATION " INTERNMET NEWSGROUP, 'Online! 21 June 1997 (1997-06-21), XP002209581 misc.activism.progressive Retrieved from the Internet: <URL:http://groups.google.com/> 'retrieved on 2002-08-12! the whole document -----	13-16

3

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 01/13260

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This International Search Report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the International Application that do not comply with the prescribed requirements to such an extent that no meaningful International Search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

see additional sheet

1. [X] As all required additional search fees were timely paid by the applicant, this International Search Report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this International Search Report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this International Search Report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest.

[X] No protest accompanied the payment of additional search fees.

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 210

This International Searching Authority found multiple (groups of) inventions in this international application, as follows:

1. Claims: 1-12

   A portal for authenticating a query for a secure computer network address

2. Claims: 13-16

   A method and a computer readable storage medium for registering a secure domain name

IN˜ ˜RNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 01/13260

Patent document cited in search report		Publication date	Patent family member(s)		Publication date
DE 19924575	A	02-12-1999	DE	19924575 A1	02-12-1999
			FR	2782873 A1	03-03-2000
			GB	2340702 A ,B	23-02-2000
			JP	2000049867 A	18-02-2000
EP 0838930	A	29-04-1998	US	6101543 A	08-08-2000
			EP	0838930 A2	29-04-1998
			JP	10178450 A	30-06-1998

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/08219

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) :G06F 11/00
US CL : 713/201
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 713/201,200,202; 340/825.31,825.34; 380/255;

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
APS US PATENT FILE; WEST; JPAB; EPAB; DWPI; TDBD;

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,805,801 A (HOLLOWAY ET AL) 08 SEPTEMBER 1998, Entire document.	1-25
Y	US 5,796,942 A (ESBENSEN) 18 AUGUST 1998, Entire document.	1-25
Y,P	US 5,905,859 A (HOLLOWAY ET AL) 18 MAY 1999, Entire document.	1-25
Y	US 5,892,903 A (KLAUS) 06 APRIL 1999, Entire document.	1-25
A	US 5,537,099 A (LIANG) 16 JULY 1996, Entire document.	1-25
A	US 5,278,901 A (SHIEH ET AL) 11 JANUARY 1994, Entire document.	1-25

[✓ X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date		
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search 20 JULY 2000	Date of mailing of the international search report 22 AUG 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer [illegible] NADEEM IQBAL Telephone No. (703) 308-5228

Form PCT/ISA/210 (second sheet) (July 1998)★

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/08219

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A,P	US 5,991,881 A (CONKLIN ET AL) 23 NOVEMBER 1999, Entire document.	1-25

Form PCT/ISA/210 (continuation of second sheet) (July 1998)★

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE 00/02565

A. CLASSIFICATION OF SUBJECT MATTER

IPC7: H04L 12/46, H04L 12/56, H04L 9/00
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC7: H04L, G09F, H04Q

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE,DK,FI,NO classes as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5898830 A (R.E.WESINGER, JR. ET AL), 27 April 1999 (27.04.99), column 3, line 47 - column 4, line 52, figure 1, claims 1-10, abstract, cited in Application --	1-20
A	C. HUITEMA: An Experiment in DNS BasedIP Routing. K B Labs Kashpureff Boling Laboratories, Inc., Network Working Group, rfc 1383, INRIA dec. 1992. http:www.kblabs.com/lab/lib/rfcs/1300/rfc1383.txt.html --	1-20
A	WO 9859470 A2 (TELEFONAKTIEBOLAGET LM ERICSSON (PUBL)), 30 December 1998 (30.12.98), page 1, line 13 - page 3, line 16, figures 1-2, claims 1-12 --	1,20

[✓] Further documents are listed in the continuation of Box C. [✓] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier application or patent but published on or after the international filing date	"Y"	document of particular relevance: the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 17 April 2001	Date of mailing of the international search report 18 -04- 2001
Name and mailing address of the ISA/ Swedish Patent Office Box 5055, S-102 42 STOCKHOLM Facsimile No. +46 8 666 02 86	Authorized officer Roger Bou Faisal/LR Telephone No. +46 8 782 25 00

Form PCT/ISA/210 (second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT
Information on patent family members

25/02/01

International application No.
PCT/SE 00/02565

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5898830 A	27/04/99	US 6052788 A	18/04/00
WO 9859470 A2	30/12/98	AU 8052398 A	04/01/99
		SE 9702385 A	24/12/98
WO 9726731 A1	24/07/97	AU 2242697 A	11/08/97

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
H04L 12/56, 29/02

A3

(11) International Publication Number: **WO 98/59470**

(43) International Publication Date: 30 December 1998 (30.12.98)

(21) International Application Number: PCT/SE98/01217

(22) International Filing Date: 23 June 1998 (23.06.98)

(30) Priority Data:
9702385-7 23 June 1997 (23.06.97) SE

(71) Applicants *(for all designated States except US)*: TELEFONAKTIEBOLAGET LM ERICSSON (publ) [SE/SE]; S-126 25 Stockholm (SE). TELIA AB [SE/SE]; S-123 86 Farsta (SE).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: KANTER, Theo [NL/SE]; Rönninge skolväg 35E, S-144 62 Rönninge (SE). FOGELHOLM, Rabbe [SE/SE]; Turevägen 54 B, S-191 47 Sollentuna (SE).

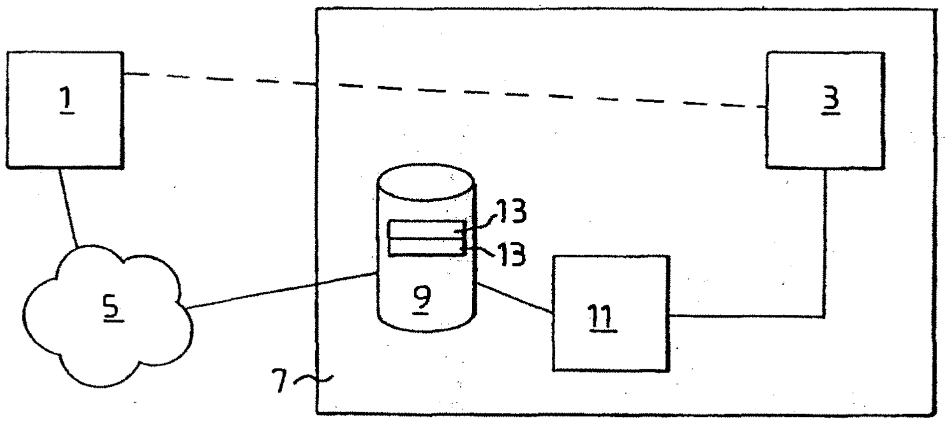
(74) Agents: HERBJØRNSEN, Rut et al., Albihns Patentbyrå Stockholm AB, P.O. Box 3137, S-103 62 Stockholm (SE).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(88) Date of publication of the international search report:
18 March 1999 (18.03.99)

(54) Title: METHOD AND APPARATUS TO ENABLE A FIRST SUBSCRIBER IN A LARGER NETWORK TO RETRIEVE THE ADDRESS OF A SECOND SUBSCRIBER IN A VIRTUAL PRIVATE NETWORK



(57) Abstract

The present invention relates to an apparatus and a method for use in a virtual private network, VPN, (7, 7'), or a network domain forming part of a larger network, such as the Internet, to enable a first subscriber (1; 1') in the larger network to retrieve the address of a second subscriber (3; 3') in the VPN. The address may be returned to the first subscriber (1; 1') or a connection means (11) may set up the connection between the subscribers (1, 3; 1', 3') automatically.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE 98/01217

A. CLASSIFICATION OF SUBJECT MATTER

IPC6: H04L 12/56, H04L 29/02
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC6: H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE,DK,FI,NO classes as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WPIL, EDOC, JAPIO

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	ITU-T Recommendation H. 323, 1996, "Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service" Paragraph 6.4, 3.41, 3.43	4-6
Y		1-3,7-12
	--	
Y	IETF RFC 883, Volume, November 1983, P. Mockapetris, "DOMAIN NAMES - IMPLEMEntATION and SPECIFICATION" page 23	1-3,7-12
	--	
A	IETF RFC 1383, Volume, December 1992, C. Huitema, "An Experiment in DNS Based IP Routing", paragraph 2	1-12
	--	

[✓] Further documents are listed in the continuation of Box C. [✓] See patent family annex.

* Special categories of cited documents:	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A" document defining the general state of the art which is not considered to be of particular relevance	"X" document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E" erlier document but published on or after the international filing date	"Y" document of particular relevance: the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&" document member of the same patent family
"O" document referring to an oral disclosure, use, exhibition or other means	
"P" document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search 12 January 1999	Date of mailing of the international search report 22 -01- 1999
Name and mailing address of the ISA/ Swedish Patent Office Box 5055, S-102 42 STOCKHOLM Facsimile No. +46 8 666 02 86	Authorized officer Christina Halldin Telephone No. +46 8 782 25 00

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE 98/01217

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	IETF RFC 2052, Volume, October 1996, A. Gulbrandsen et al, "A DNS RR for specifying the location of services (DNS SRV)", see the whole document --	1-12
A	EP 0752674 A1 (SUN MICROSYSEMS, INC.), 8 January 1997 (08.01.97), abstract -- --------	1-12

INTERNATIONAL SEARCH REPORT
Information on patent family members

01/12/98

International application No.
PCT/SE 98/01217

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0752674 A1	08/01/97	JP 9171465 A	30/06/97
		US 5745683 A	28/04/98

Form PCT/ISA/210 (patent family annex) (July 1992)

JMP/INT'L

PATENT COOPERATION TREATY

From the INTERNATIONAL SEARCHING AUTHORITY

To:
BANNER & WITCOFF, LTD.
Attn. Curtin, Joseph P.
1001 G Street, N.W.
Eleventh Floor
Washington, DC 20001-4597 K9
UNITED STATES OF AMERICA

RECEIVED
JUN 24 2002
BANNER & WITCOFF, LTD.

PCT

INVITATION TO PAY ADDITIONAL FEES

(PCT Article 17(3)(a) and Rule 40)

00479.00028

DOCKETED
JUN 25 2002
SB
Fee Due 8/2/02

	Date of mailing *(day/month/year)* 18/06/2002
Applicant's or agent's file reference 00479.00028	**PAYMENT DUE** within 45 ~~months~~/days from the above date of mailing
International application No. PCT/US 01/13260	International filing date *(day/month/year)* 25/04/2001
Applicant SCIENCE APPLICATIONS INTERNATIONAL CORPORATION	

1. This International Searching Authority

(i) considers that there are 2 *(number of)* inventions claimed in the international application covered by the claims indicated ~~below~~/on the extra sheet:

and it considers that the international application does not comply with the requirements of unity of invention (Rules 13.1, 13.2 and 13.3) for the reasons indicated ~~below~~/on the extra sheet:

(ii) [X] has carried out a partial international search (see Annex) [ ] will establish the international search report on those parts of the international application which relate to the invention first mentioned in claims Nos.:
1-12

(iii) will establish the international search report on the other parts of the international application only if, and to the extent to which, additional fees are paid

2. The applicant is hereby **invited,** within the time limit indicated above, to pay the amount indicated below:

EUR 945,00	x	1	=	EUR 945,00
Fee per additional invention		number of additional inventions		total amount of additional fees
Or,	x		=	

The applicant is informed that, according to Rule 40.2(c), **the payment of any additional fee may be made under protest,** i.e., a reasoned statement to the effect that the international application complies with the requirement of unity of invention or that the amount of the required additional fee is excessive.

3. [ ] Claim(s) Nos.: have been found to be unsearchable under Article 17(2)(b) because of defects under Article 17(2)(a) and therefore have not been included with any invention.

Name and mailing address of the International Searching Authority	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL-2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Claude Berthon

Form PCT/ISA/206 (July 1992)

RECOMMANDÉ

Annex to Form PCT/ISA/206
COMMUNICATION RELATING TO THE RESULTS OF THE PARTIAL INTERNATIONAL SEARCH

International Application No
PCT/US 01/13260

1.The present communication is an Annex to the invitation to pay additional fees (Form PCT/ISA/206). It shows the results of the international search established on the parts of the international application which relate to the invention first mentioned in claims Nos.:
1-12

2.This communication is not the international search report which will be established according to Article 18 and Rule 43.

3.If the applicant does not pay any additional search fees, the information appearing in this communication will be considered as the result of the international search and will be included as such in the international search report.

4.If the applicant pays additional fees, the international search report will contain both the information appearing in this communication and the results of the international search on other parts of the international application for which such fees will have been paid.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
Y	DONALD E. EASTLAKE 3RD : "<draft-ietf-dnssec-secext2-05.txt> Domain Name System Security Extensions" INTERNET DRAFT , [Online] April 1998 (1998-04), XP002199931 Retrieved from the Internet: <URL:ftp://ftp.inet.no/pub/ietf/internet-drafts/draft-ietf-dnssec-secext2-05.txt> [retrieved on 2002-05-23] 1. Overview of the contents 2.3 Data origin authemtication and integrity 2.4 DNS transaction and request authentication ---	1,4-6
Y	CHAPMAN D.B.; ZWICKY E.D.: " Building Internet Firewalls " O'REILLY, November 1995 (1995-11), XP002199932 pag 278-296 pag 351-375 ---	1,4-6
P,X	DE 199 24 575 A (SUN MICROSYSTEMS INC) 2 December 1999 (1999-12-02) abstract column 2, line 48 -column 3, line 6 ---	1,4,5,7

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

2

Form PCT/ISA/206 (Annex, first sheet) (July 1992)

Annex to Form PCT/ISA/206
COMMUNICAT[illegible] R[illegible]TING TO THE RESULTS
OF THE PART[illegible] I[illegible]RNATIONAL SEARCH

[illegible]al Application No
PCT/US 01/13260

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	P. SRISURESH, G. TSIRTSIS, P. AKKIRAJU, A. HEFFERNAN: "<draft-ietf-nat-dns-alg-00.txt> DNS extensions to Network Address Translators (DNS_ALG)" INTERNET DRAFT , [Online] July 1998 (1998-07), XP002199933 Retrieved from the Internet: <URL:ftp://ftp.inet.no/pub/ietf/internet-drafts/draft-ietf-nat-dns-alg-00.txt> [retrieved on 2002-05-23] 1. Introduction 2. Requirement for DNS extensions fig 3 5.3 Incoming name lookup queries 8. Security considerations ---	1-12
A	EP 0 838 930 A (DIGITAL EQUIPMENT CORP) 29 April 1998 (1998-04-29) abstract column 9, line 11 -column 10, line 34 -----	1-12

2

Form PCT/ISA/206 (Annex, continuation sheet)(July 1992)

INVITATION TO PAY ADDITIONAL FEES

International application No.
PCT/US 01/13260

This International Searching Authority found multiple (groups of) inventions in this international application, as follows:

1. Claims: 1-12

   A portal for authenticating a query for a secure computer network address

2. Claims: 13-16

   A method and a computer readable storage medium for registering a secure domain name

For the following reasoning document XP002199931 is taken into account. From this prior art document what is known is a domain name system which implements security extensions.

With the reference to the prior art document, the first group yields the potential special technical feature of using a portal for authenticating a query for a secure computer network address, hence solving the object problem of hiding information from untrusted clients.

With the reference to the prior art document, the second group yields the potential special technical feature of veryfing ownership information for an equivalent non-secure domain name corresponding to the secure domain name, hence solving the object problem of conflicting ownership of secure and non-secure domain names.

Consequently, neither the objective problems underlying the subjects of the two claimed inventions, nor the solutions as defined by the special technical features described allow for the link of a common inventive concept to be established between said inventions. In conclusion therefore the two groups of claims are not linked by a single general inventive concept. The application hence does not meet the requirements of unity of invention as defined in Rule 13 (1) and (2) of the PCT.

Patent Family Annex

I. lat[illegible] patent family members

[illegible] al Application No
PCT/US 01/13260

Patent document cited in search report		Publication date	Patent family member(s)			Publication date
DE 19924575	A	02-12-1999	DE	19924575	A1	02-12-1999
			FR	2782873	A1	03-03-2000
			GB	2340702	A ,B	23-02-2000
			JP	2000049867	A	18-02-2000
EP 0838930	A	29-04-1998	US	6101543	A	08-08-2000
			EP	0838930	A2	29-04-1998
			JP	10178450	A	30-06-1998

Form PCT/ISA/206 (patent family annex) (July 1992)

¨̈IͰ ͻRTANT INFORMATIC¨ (

Bank and giro accounts of the
European Patent Organisation
for payments in

EUR

N° 3 338 800/11 (BLZ 700 800 00)
Dresdner Bank
Promenadeplatz 7
D-80273 München

N° 182000-805 (BLZ 700 100 80)
Postbank AG
Bayerstr. 49
D-80138 München

Bank and giro accounts of the
European Patent Organisation
for payments in

DEM

N° 3 338 800 00 (BLZ 700 800 00)
Dresdner Bank
Promenadeplatz 7
D-80273 München

N° 300-800 (BLZ 700 100 80)
Postbank AG
Bayerstr. 49
D-80138 München

JMK/BCW/INT

000479·

DOCKE

JUL 0 8

Fee

8

PATENT COOPERATION TRE...TY

From the INTERNATIONAL SEARCHING AUTHORITY

To:

BANNER & WITCOFF, LTD.
Attn. Curtin, Joseph W.
1001 G Street, N.W.
Eleventh Floor
Washington, DC 20001-4597 K9
UNITED STATES OF AMERICA

RECEIVED
JUL 0 8 2002
BANNER & WITCOFF, LTD.

PCT

INVITATION TO PAY ADDITIONAL FEES

(PCT Article 17(3)(a) and Rule 40.1)

	Date of mailing *(day/month/year)* 28/06/2002
Applicant's or agent's file reference 00479.00027	**PAYMENT DUE** within 45 ~~months~~/days from the above date of mailing
International application No. PCT/US 01/13261	International filing date *(day/month/year)* 25/04/2001
Applicant SCIENCE APPLICATIONS INTERNATIONAL CORPORATION	

1. This International Searching Authority

(i) considers that there are 3 *(number of)* inventions claimed in the international application covered by the claims indicated ~~below~~/on the extra sheet:

and it considers that the international application does not comply with the requirements of unity of invention (Rules 13.1, 13.2 and 13.3) for the reasons indicated ~~below~~/on the extra sheet:

(ii) [X] has carried out a partial international search (see Annex) [ ] will establish the international search report on those parts of the international application which relate to the invention first mentioned in claims Nos.:
1-4,8-14,16-19,23-29,53-59,63-70,74-81,85-92,96-102,106-112

(iii) will establish the international search report on the other parts of the international application only if, and to the extent to which, additional fees are paid

2. The applicant is hereby **invited**, within the time limit indicated above, to pay the amount indicated below:

EUR 945,00 x 2 = EUR 1.890,00
Fee per additional invention / number of additional inventions / total amount of additional fees

Or, ______ x ______ = ______

The applicant is informed that, according to Rule 40.2(c), **the payment of any additional fee may be made under protest**, i.e., a reasoned statement to the effect that the international application complies with the requirement of unity of invention or that the amount of the required additional fee is excessive.

3. [ ] Claim(s) Nos. ______ have been found to be unsearchable under Article 17(2)(b) because of defects under Article 17(2)(a) and therefore have not been included with any invention.

Name and mailing address of the International Searching Authority European Patent Office, P.B. 5818 Patentlaan 2 NL-2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Claude Berthon

Form PCT/ISA/206 (July 1992)

Annex to Form PCT/ISA/206
COMMUNICATION RELATING TO THE RESULTS OF THE PARTIAL INTERNATIONAL SEARCH

International Application No
PCT/US 01/13261

1.The present communication is an Annex to the invitation to pay additional fees (Form PCT/ISA/206). It shows the results of the international search established on the parts of the international application which relate to the invention first mentioned in claims Nos.:
1-4,8-14,16-19,23-29,53-59,63-70,74-81,85-92,96-102,106-112,116

2.This communication is not the international search report which will be established according to Article 18 and Rule 4

3.If the applicant does not pay any additional search fees, the information appearing in this communication will be considered as the result of the international search and will be included as such in the international search report.

4.If the applicant pays additional fees, the international search report will contain both the information appearing in this communication and the results of the international search on other parts of the international application for which such fees will have been paid.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 838 930 A (DIGITAL EQUIPMENT CORP) 29 April 1998 (1998-04-29)	1,8,16, 23,53, 54, 56-58, 63-65, 67-69, 74-76, 78-80, 85-87, 89-91, 96-98, 100,101, 106-108, 110,111, 116
Y	abstract column 3, line 30 -column 4, line 14 column 16, line 12 -column 17, line 14 column 22, line 56 -column 23, line 20 figures 3,4,9,11-13,21,22 ---	2,4, 9-14,17, 18, 24-29, 55,59, 66,70, 77,81, 88,92, 99,102, 109,112

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

3

Form PCT/ISA/206 (Annex, first sheet) (July 1992)

Annex to Form PCT/ISA/206
CO JNICATION RELATING TO THE RESULTS
OF THE PARTIAL INTERNATIONAL SEARCH

International Application No
PCT/US 01/13261

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	GB 2 317 792 A (SECURE COMPUTING CORP) 1 April 1998 (1998-04-01) abstract page 2, line 3 - line 25 page 3, line 1 - line 10 page 8, line 18 - line 24 page 10, line 20 -page 12, line 2 page 12, line 26 - line 31 page 13, line 16 - line 29 figures 1-4	1,4,8, 14,16, 18,23, 29, 53-58, 63-69, 74-80, 85-91, 96-101, 106-111, 116
Y	EP 0 814 589 A (AT & T CORP) 29 December 1997 (1997-12-29) page 1, line 45 -page 2, line 2 page 5, line 8 - line 14	2,17
X Y	US 5 588 060 A (AZIZ ASHAR) 24 December 1996 (1996-12-24) abstract column 4, line 38 - line 46 column 6, line 50 - line 67 figure 2	1,3,16, 19 59,70, 81,92, 102,112
Y	US 5 689 566 A (NGUYEN MINHTAM C) 18 November 1997 (1997-11-18) abstract column 9, line 10 -column 10, line 4	4,9,10, 14,18, 24,25, 29,55, 66,77, 88,99, 109
Y	WO 98 27783 A (NORTHERN TELECOM LTD ;HOLMES KIM (US); HUI MARGARET (US); TELLO AN) 25 June 1998 (1998-06-25) abstract figure 3	11,26

-/--

3

Form PCT/ISA/206 (Annex, continuation sheet)(July 1992)

Annex to Form PCT/ISA/206
CO JNICATION RELATING TO THE RESULTS
OF THE PARTIAL INTERNATIONAL SEARCH

International Application No
PCT/US 01/13261

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
Y	LAURIE WELLS (LANCASTERB1B@EMAIL.MSN.COM): "Subject: Security Icon " USENET NEWSGROUP, 'Online! 19 October 1998 (1998-10-19), XP002200606 microsoft.public.inetexplorer.ie4.security Retrieved from the Internet: <URL:http://groups.google.com/> 'retrieved on 2002-05-30! the whole document	12,13, 27,28
A	STALLINGS W: "CRYPTOGRAPHY AND NETWORK SECURITY, PRINCIPLES AND PRACTICE, 2ND EDITION" CRYPTOGRAPHY AND NETWORK SECURITY, XX, XX, 8 June 1998 (1998-06-08), pages 399-440, XP002167283 13.4 Encapsulating security payload 13.5 Combining security associations	53-59, 63-70, 74-81, 85-92, 96-102, 106-112, 116
L	WILLIAM STALLINGS (WS@SHORE.NET): "Subject: new cryptography and network security book " USENET NEWSGROUP, 'Online! 8 June 1998 (1998-06-08), XP002200607 comp.security.misc Retrieved from the Internet: <URL:http://groups.google.com/> 'retrieved on 2002-05-30! Proof of publication date of XP002167283	

3

Form PCT/ISA/206 (Annex, continuation sheet)(July 1992)

INVITATION TO PAY ADDITIONAL FEES

International application No.
PCT/US 01/13261

This International Searching Authority found multiple (groups of) inventions in this international application, as follows:

1. Claims: 1-4 8-14 16-19 23-29 53-59 63-70 74-81 85-92 96-102 106-112 116

   A method and computer readable medium for loading a secure communication software module

2. Claims: 5-7 (as dependent from 1) 20-22 (as dependent from 16) 60-62 (as dependent from 53) 71-73 (as dependent from 64) 82-84 (as dependent from 75) 93-95 (as dependent from 86) 103-105 (as dependent from 97) 113-115 (as dependent from 107)

   A method and computer readable medium based on a computer address hopping regime

3. Claims: 15 (as dependent from 1), 30 (as dependent from 16), 31-52

   A method and computer readable medium for sending a query for a secure network addess to a secure domain name server

For the following reasoning document EP838930 is taken into account. From this prior art document what is known (see fig. 3) is a system and a method for establishing a secure virtual private network link over a computer network (as defined in claim 1).

With the reference to the prior art document, the first group yields the potential special technical feature of locally storing a secure communication software module when the it is not already locally available (as defined in claim 2), hence solving the object problem manually configuring every computer for performing secure communications.

With the reference to the prior art document, the second group yields the potential special technical feature of establishing a connection based on a computer address hopping regime (as defined in claim 5), hence solving the object problem of providing security against eavesdropping on communications over the Internet.

With the reference to the prior art document, the third group yields the potential special technical feature of sending a query for a secure network address to a secure domain name server (as defined in claim 31), hence solving the object problem hiding the true IP address of web sites.

Consequently, neither the objective problems underlying the subjects of the three claimed inventions, nor the solutions as defined by the special technical features described allow for the link of a common inventive concept to be established between said inventions. In

INVITATION TO PAY ADDITIONAL FEES

International application No.

PCT/US 01/13261

conclusion therefore the three groups of claims are not linked by a single general inventive concept. The application hence does not meet the requirements of unity of invention as defined in Rule 13 (1) and (2) of the PCT.

Patent Family Annex

Information on patent family members

International Application No
PCT/US 01/13261

Patent document cited in search report		Publication date	Patent family member(s)		Publication date
EP 0838930	A	29-04-1998	US	6101543 A	08-08-2000
			EP	0838930 A2	29-04-1998
			JP	10178450 A	30-06-1998
GB 2317792	A	01-04-1998	US	5983350 A	09-11-1999
			US	5950195 A	07-09-1999
			DE	19741239 A1	07-05-1998
			DE	19741246 A1	19-03-1998
			GB	2317539 A ,B	25-03-1998
EP 0814589	A	29-12-1997	US	6058250 A	02-05-2000
			CA	2204058 A1	19-12-1997
			EP	0814589 A2	29-12-1997
US 5588060	A	24-12-1996	EP	0693836 A1	24-01-1996
			JP	8008895 A	12-01-1996
			US	5668877 A	16-09-1997
			US	5633933 A	27-05-1997
			US	6091820 A	18-07-2000
			US	6026167 A	15-02-2000
US 5689566	A	18-11-1997	US	5638448 A	10-06-1997
WO 9827783	A	25-06-1998	US	6032118 A	29-02-2000
			AU	727878 B2	04-01-2001
			AU	5131198 A	15-07-1998
			DE	19782193 D2	25-11-1999
			EP	1008275 A1	14-06-2000
			GB	2336511 A ,B	20-10-1999
			WO	9827783 A1	25-06-1998
			SE	9902261 A	16-06-1999

/NTL

PATENT COOPERATION TREATY

From the INTERNATIONAL SEARCHING AUTHORITY

PCT

NOTIFICATION OF TRANSMITTAL OF THE INTERNATIONAL SEARCH REPORT OR THE DECLARATION

(PCT Rule 44.1)

To:
BANNER & WITCOFF, LTD.
Attn. Wright, Bradley C.
1001 G Street, N.W.
Eleventh Floor
Washington, DC 20001-4597
UNITED STATES OF AMERICA

00479.00029

RECEIVED
AUG 27 2002
BANNER WITCOFF

Case Closed

Date of mailing (day/month/year)	20/08/2002
Applicant's or agent's file reference 00479.00029	FOR FURTHER ACTION See paragraphs 1 and 4 below
International application No. PCT/US 01/04340	International filing date (day/month/year) 12/02/2001
Applicant SCIENCE APPLICATIONS INTERNATIONAL CORPORATION	

1. [✓] The applicant is hereby notified that the International Search Report has been established and is transmitted herewith.

Filing of amendments and statement under Article 19:
The applicant is entitled, if he so wishes, to amend the claims of the International Application (see Rule 46):

When? The time limit for filing such amendments is normally 2 months from the date of transmittal of the International Search Report; however, for more details, see the notes on the accompanying sheet.

Where? Directly to the International Bureau of WIPO
34, chemin des Colombettes
1211 Geneva 20, Switzerland
Fascimile No.: (41-22) 740.14.35

For more detailed instructions, see the notes on the accompanying sheet.

2. [ ] The applicant is hereby notified that no International Search Report will be established and that the declaration under Article 17(2)(a) to that effect is transmitted herewith.

3. [ ] **With regard to the protest** against payment of (an) additional fee(s) under Rule 40.2, the applicant is notified that:

[ ] the protest together with the decision thereon has been transmitted to the International Bureau together with the applicant's request to forward the texts of both the protest and the decision thereon to the designated Offices.

[ ] no decision has been made yet on the protest; the applicant will be notified as soon as a decision is made.

4. **Further action(s):** The applicant is reminded of the following:

Shortly after **18 months** from the priority date, the international application will be published by the International Bureau. If the applicant wishes to avoid or postpone publication, a notice of withdrawal of the international application, or of the priority claim, must reach the International Bureau as provided in Rules 90*bis*.1 and 90*bis*.3, respectively, before the completion of the technical preparations for international publication.

Within **19 months** from the priority date, a demand for international preliminary examination must be filed if the applicant wishes to postpone the entry into the national phase until 30 months from the priority date (in some Offices even later).

Within **20 months** from the priority date, the applicant must perform the prescribed acts for entry into the national phase before all designated Offices which have not been elected in the demand or in a later election within 19 months from the priority date or could not be elected because they are not bound by Chapter II.

Name and mailing address of the International Searching Authority	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL-2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Claude Berthon

Form PCT/ISA/220 (July 1998)

NOTES TO FORM PCT/ISA/220

These Notes are intended to give the basic instructions concerning the filing of amendments under article 19. The Notes are based on the requirements of the Patent Cooperation Treaty, the Regulations and the Administrative Instructions under that Treaty. In case of discrepancy between these Notes and those requirements, the latter are applicable. For more detailed information, see also the PCT Applicant's Guide, a publication of WIPO.

In these Notes, "Article", "Rule", and "Section" refer to the provisions of the PCT, the PCT Regulations and the PCT Administrative Instructions, respectively.

INSTRUCTIONS CONCERNING AMENDMENTS UNDER ARTICLE 19

The applicant has, after having received the international search report, one opportunity to amend the claims of the international application. It should however be emphasized that, since all parts of the international application (claims, description and drawings) may be amended during the international preliminary examination procedure, there is usually no need to file amendments of the claims under Article 19 except where, e.g. the applicant wants the latter to be published for the purposes of provisional protection or has another reason for amending the claims before international publication. Furthermore, it should be emphasized that provisional protection is available in some States only.

What parts of the international application may be amended?

Under Article 19, only the claims may be amended.

During the international phase, the claims may also be amended (or further amended) under Article 34 before the International Preliminary Examining Authority. The description and drawings may only be amended under Article 34 before the International Examining Authority.

Upon entry into the national phase, all parts of the international application may be amended under Article 28 or, where applicable, Article 41.

When? Within 2 months from the date of transmittal of the international search report or 16 months from the priority date, whichever time limit expires later. It should be noted, however, that the amendments will be considered as having been received on time if they are received by the International Bureau after the expiration of the applicable time limit but before the completion of the technical preparations for international publication (Rule 46.1).

Where not to file the amendments?

The amendments may only be filed with the International Bureau and not with the receiving Office or the International Searching Authority (Rule 46.2).

Where a demand for international preliminary examination has been/is filed, see below.

How? Either by cancelling one or more entire claims, by adding one or more new claims or by amending the text of one or more of the claims as filed.

A replacement sheet must be submitted for each sheet of the claims which, on account of an amendment or amendments, differs from the sheet originally filed.

All the claims appearing on a replacement sheet must be numbered in Arabic numerals. Where a claim is cancelled, no renumbering of the other claims is required. In all cases where claims are renumbered, they must be renumbered consecutively (Administrative Instructions, Section 205(b)).

The amendments must be made in the language in which the international application is to be published.

What documents must/may accompany the amendments?

Letter (Section 205(b)):

The amendments must be submitted with a letter.

The letter will not be published with the international application and the amended claims. It should not be confused with the "Statement under Article 19(1)" (see below, under "Statement under Article 19(1)").

The letter must be in English or French, at the choice of the applicant. However, if the language of the international application is English, the letter must be in English; if the language of the international application is French, the letter must be in French.

NOTES TO FORM PCT/ISA/220 (continued)

The letter must indicate the differences between the claims as filed and the claims as amended. It must, in particular, indicate, in connection with each claim appearing in the international application (it being understood that identical indications concerning several claims may be grouped),whether

(i) the claim is unchanged;

(ii) the claim is cancelled;

(iii) the claim is new;

(iv) the claim replaces one or more claims as filed;

(v) the claim is the result of the division of a claim as filed.

The following examples illustrate the manner in which amendments must be explained in the accompanying letter:

1. [Where originally there were 48 claims and after amendment of some claims there are 51]:
"Claims 1 to 29, 31, 32, 34, 35, 37 to 48 replaced by amended claims bearing the same numbers; claims 30, 33 and 36 unchanged; new claims 49 to 51 added."
2. [Where originally there were 15 claims and after amendment of all claims there are 11]:
"Claims 1 to 15 replaced by amended claims 1 to 11."
3. [Where originally there were 14 claims and the amendments consist in cancelling some claims and in adding new claims]:
"Claims 1 to 6 and 14 unchanged; claims 7 to 13 cancelled; new claims 15, 16 and 17 added." or
"Claims 7 to 13 cancelled; new claims 15, 16 and 17 added; all other claims unchanged."
4. [Where various kinds of amendments are made]:
"Claims 1-10 unchanged; claims 11 to 13, 18 and 19 cancelled; claims 14, 15 and 16 replaced by amended claim 14; claim 17 subdivided into amended claims 15, 16 and 17; new claims 20 and 21 added."

"Statement under article 19(1)" (Rule 46.4)

The amendments may be accompanied by a statement explaining the amendments and indicating any impact that such amendments might have on the description and the drawings (which cannot be amended under Article 19(1)).

The statement will be published with the international application and the amended claims.

It must be in the language in which the international application is to be published.

It must be brief, not exceeding 500 words if in English or if translated into English.

It should not be confused with and does not replace the letter indicating the differences between the claims as filed and as amended. It must be filed on a separate sheet and must be identified as such by a heading, preferably by using the words "Statement under Article 19(1)."

It may not contain any disparaging comments on the international search report or the relevance of citations contained in that report. Reference to citations, relevant to a given claim, contained in the international search report may be made only in connection with an amendment of that claim.

Consequence if a demand for international preliminary examination has already been filed

If, at the time of filing any amendments and any accompanying statement, under Article 19, a demand for international preliminary examination has already been submitted, the applicant must preferably, at the time of filing the amendments (and any statement) with the International Bureau, also file with the International Preliminary Examining Authority a copy of such amendments (and of any statement) and, where required, a translation of such amendments for the procedure before that Authority (see Rules 55.3(a) and 62.2, first sentence). For further information, see the Notes to the demand form (PCT/IPEA/401).

Consequence with regard to translation of the international application for entry into the national phase

The applicant's attention is drawn to the fact that, upon entry into the national phase, a translation of the claims as amended under Article 19 may have to be furnished to the designated/elected Offices, instead of, or in addition to, the translation of the claims as filed.

For further details on the requirements of each designated/elected Office, see Volume II of the PCT Applicant's Guide.

[illegible]

RECEIVED

AUG 2 9 2002

BANNER WITCOFF

PATENT COOPERATION TREATY

From the INTERNATIONAL SEARCHING AUTHORITY

To:

BANNER & WITCOFF, LTD.
Attn. Curtin, Joseph P.
1001 G Street, N.W.
Eleventh Floor
Washington, DC 20001-4597 K9
UNITED STATES OF AMERICA

[illegible]

00479.00028

DOCKETED

AUG 2 9 2002

PCT

NOTIFICATION OF TRANSMITTAL OF THE INTERNATIONAL SEARCH REPORT OR THE DECLARATION

(PCT Rule 44.1)

00479.00028

DOCKETED

AUG 2 9 2002

Art. 19 Amend 10/23/02

	Date of mailing *(day/month/year)* 23/08/2002
Applicant's or agent's file reference 00479.00028	**FOR FURTHER ACTION** See paragraphs 1 and 4 below
International application No. PCT/US 01/ 13260	International filing date *(day/month/year)* 25/04/2001
Applicant SCIENCE APPLICATIONS INTERNATIONAL CORPORATION	

IDS [illegible] 11/23/02

30 day 9/22/02

1. [X] The applicant is hereby notified that the International Search Report has been established and is transmitted herewith.

Filing of amendments and statement under Article 19:
The applicant is entitled, if he so wishes, to amend the claims of the International Application (see Rule 46):

When? The time limit for filing such amendments is normally 2 months from the date of transmittal of the International Search Report; however, for more details, see the notes on the accompanying sheet.

Where? Directly to the International Bureau of WIPO
34, chemin des Colombettes
1211 Geneva 20, Switzerland
Fascimile No.: (41-22) 740.14.35

For more detailed instructions, see the notes on the accompanying sheet.

2. [ ] The applicant is hereby notified that no International Search Report will be established and that the declaration under Article 17(2)(a) to that effect is transmitted herewith.

3. [ ] **With regard to the protest** against payment of (an) additional fee(s) under Rule 40.2, the applicant is notified that:

[ ] the protest together with the decision thereon has been transmitted to the International Bureau together with the applicant's request to forward the texts of both the protest and the decision thereon to the designated Offices.

[ ] no decision has been made yet on the protest; the applicant will be notified as soon as a decision is made.

4. **Further action(s):** The applicant is reminded of the following:

Shortly after **18 months** from the priority date, the international application will be published by the International Bureau. If the applicant wishes to avoid or postpone publication, a notice of withdrawal of the international application, or of the priority claim, must reach the International Bureau as provided in Rules 90*bis*.1 and 90*bis*.3, respectively, before the completion of the technical preparations for international publication.

Within **19 months** from the priority date, a demand for international preliminary examination must be filed if the applicant wishes to postpone the entry into the national phase until 30 months from the priority date (in some Offices even later).

Within **20 months** from the priority date, the applicant must perform the prescribed acts for entry into the national phase before all designated Offices which have not been elected in the demand or in a later election within 19 months from the priority date or could not be elected because they are not bound by Chapter II.

Name and mailing address of the International Searching Authority	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2, NL-2280 HV Rijswijk, Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Claude Berthon

Form PCT/ISA/220 (July 1998)

NOTES TO FORM PCT/ISA/220

These Notes are intended to give the basic instructions concerning the filing of amendments under article 19. The Notes are based on the requirements of the Patent Cooperation Treaty, the Regulations and the Administrative Instructions under that Treaty. In case of discrepancy between these Notes and those requirements, the latter are applicable. For more detailed information, see also the PCT Applicant's Guide, a publication of WIPO.

In these Notes, "Article", "Rule", and "Section" refer to the provisions of the PCT, the PCT Regulations and the PCT Administrative Instructions, respectively.

INSTRUCTIONS CONCERNING AMENDMENTS UNDER ARTICLE 19

The applicant has, after having received the international search report, one opportunity to amend the claims of the international application. It should however be emphasized that, since all parts of the international application (claims, description and drawings) may be amended during the international preliminary examination procedure, there is usually no need to file amendments of the claims under Article 19 except where, e.g. the applicant wants the latter to be published for the purposes of provisional protection or has another reason for amending the claims before international publication. Furthermore, it should be emphasized that provisional protection is available in some States only.

What parts of the international application may be amended?

Under Article 19, only the claims may be amended.

During the international phase, the claims may also be amended (or further amended) under Article 34 before the International Preliminary Examining Authority. The description and drawings may only be amended under Article 34 before the International Examining Authority.

Upon entry into the national phase, all parts of the international application may be amended under Article 28 or, where applicable, Article 41.

When? Within 2 months from the date of transmittal of the international search report or 16 months from the priority date, whichever time limit expires later. It should be noted, however, that the amendments will be considered as having been received on time if they are received by the International Bureau after the expiration of the applicable time limit but before the completion of the technical preparations for international publication (Rule 46.1).

Where not to file the amendments?

The amendments may only be filed with the International Bureau and not with the receiving Office or the International Searching Authority (Rule 46.2).

Where a demand for international preliminary examination has been/is filed, see below.

How? Either by cancelling one or more entire claims, by adding one or more new claims or by amending the text of one or more of the claims as filed.

A replacement sheet must be submitted for each sheet of the claims which, on account of an amendment or amendments, differs from the sheet originally filed.

All the claims appearing on a replacement sheet must be numbered in Arabic numerals. Where a claim is cancelled, no renumbering of the other claims is required. In all cases where claims are renumbered, they must be renumbered consecutively (Administrative Instructions, Section 205(b)).

The amendments must be made in the language in which the international application is to be published.

What documents must/may accompany the amendments?

Letter (Section 205(b)):

The amendments must be submitted with a letter.

The letter will not be published with the international application and the amended claims. It should not be confused with the "Statement under Article 19(1)" (see below, under "Statement under Article 19(1)").

The letter must be in English or French, at the choice of the applicant. However, if the language of the international application is English, the letter must be in English; if the language of the international application is French, the letter must be in French.

NOTES TO FORM PCT/ISA/220 (continued)

The letter must indicate the differences between the claims as filed and the claims as amended. It must, in particular, indicate, in connection with each claim appearing in the international application (it being understood that identical indications concerning several claims may be grouped),whether

(i) the claim is unchanged;

(ii) the claim is cancelled;

(iii) the claim is new;

(iv) the claim replaces one or more claims as filed;

(v) the claim is the result of the division of a claim as filed.

The following examples illustrate the manner in which amendments must be explained in the accompanying letter:

1. [Where originally there were 48 claims and after amendment of some claims there are 51]: "Claims 1 to 29, 31, 32, 34, 35, 37 to 48 replaced by amended claims bearing the same numbers; claims 30, 33 and 36 unchanged; new claims 49 to 51 added."
2. [Where originally there were 15 claims and after amendment of all claims there are 11]: "Claims 1 to 15 replaced by amended claims 1 to 11."
3. [Where originally there were 14 claims and the amendments consist in cancelling some claims and in adding new claims]: "Claims 1 to 6 and 14 unchanged; claims 7 to 13 cancelled; new claims 15, 16 and 17 added." or "Claims 7 to 13 cancelled; new claims 15, 16 and 17 added; all other claims unchanged."
4. [Where various kinds of amendments are made]: "Claims 1-10 unchanged; claims 11 to 13, 18 and 19 cancelled; claims 14, 15 and 16 replaced by amended claim 14; claim 17 subdivided into amended claims 15, 16 and 17; new claims 20 and 21 added."

"Statement under article 19(1)" (Rule 46.4)

The amendments may be accompanied by a statement explaining the amendments and indicating any impact that such amendments might have on the description and the drawings (which cannot be amended under Article 19(1)).

The statement will be published with the international application and the amended claims.

It must be in the language in which the international application is to be published.

It must be brief, not exceeding 500 words if in English or if translated into English.

It should not be confused with and does not replace the letter indicating the differences between the claims as filed and as amended. It must be filed on a separate sheet and must be identified as such by a heading, preferably by using the words "Statement under Article 19(1)."

It may not contain any disparaging comments on the international search report or the relevance of citations contained in that report. Reference to citations, relevant to a given claim, contained in the international search report may be made only in connection with an amendment of that claim.

Consequence if a demand for international preliminary examination has already been filed

If, at the time of filing any amendments and any accompanying statement, under Article 19, a demand for international preliminary examination has already been submitted, the applicant must preferably, at the time of filing the amendments (and any statement) with the International Bureau, also file with the International Preliminary Examining Authority a copy of such amendments (and of any statement) and, where required, a translation of such amendments for the procedure before that Authority (see Rules 55.3(a) and 62.2, first sentence). For further information, see the Notes to the demand form (PCT/IPEA/401).

Consequence with regard to translation of the international application for entry into the national phase

The applicant's attention is drawn to the fact that, upon entry into the national phase, a translation of the claims as amended under Article 19 may have to be furnished to the designated/elected Offices, instead of, or in addition to, the translation of the claims as filed.

For further details on the requirements of each designated/elected Office, see Volume II of the PCT Applicant's Guide.

PATENT COOPERATION TREAT

PCT

INTERNATIONAL SEARCH REPORT

(PCT Article 18 and Rules 43 and 44)

Applicant's or agent's file reference 00479.00029	**FOR FURTHER ACTION** see Notification of Transmittal of International Search Report (Form PCT/ISA/220) as well as, where applicable, item 5 below.	
International application No. PCT/US 01/ 04340	International filing date *(day/month/year)* 12/02/2001	(Earliest) Priority Date *(day/month/year)* 15/02/2000
Applicant SCIENCE APPLICATIONS INTERNATIONAL CORPORATION		

This International Search Report has been prepared by this International Searching Authority and is transmitted to the applicant according to Article 18. A copy is being transmitted to the International Bureau.

This International Search Report consists of a total of 5 sheets.

[✓] It is also accompanied by a copy of each prior art document cited in this report.

1. **Basis of the report**

   a. With regard to the **language**, the international search was carried out on the basis of the international application in the language in which it was filed, unless otherwise indicated under this item.

   [ ] the international search was carried out on the basis of a translation of the international application furnished to this Authority (Rule 23.1(b)).

   b. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, the international search was carried out on the basis of the sequence listing :

   [ ] contained in the international application in written form.
   [ ] filed together with the international application in computer readable form.
   [ ] furnished subsequently to this Authority in written form.
   [ ] furnished subsequently to this Authority in computer readble form.
   [ ] the statement that the subsequently furnished written sequence listing does not go beyond the disclosure in the international application as filed has been furnished.
   [ ] the statement that the information recorded in computer readable form is identical to the written sequence listing has been furnished

2. [ ] **Certain claims were found unsearchable** (See Box I).
3. [✓] **Unity of invention is lacking** (see Box II).

4. With regard to the **title,**

   [ ] the text is approved as submitted by the applicant.
   [✓] the text has been established by this Authority to read as follows:

   AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS WITH ASSURED SYSTEM AVAILABILI TY

5. With regard to the **abstract,**

   [✓] the text is approved as submitted by the applicant.
   [ ] the text has been established, according to Rule 38.2(b), by this Authority as it appears in Box III. The applicant may, within one month from the date of mailing of this international search report, submit comments to this Authority.

6. The figure of the **drawings** to be published with the abstract is Figure No. 3a

   [ ] as suggested by the applicant.
   [✓] because the applicant failed to suggest a figure.
   [ ] because this figure better characterizes the invention.
   [ ] None of the figures.

Form PCT/ISA/210 (first sheet) (July 1998)

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 01/04340

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L12/56 H04L29/06 H04L12/46

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, PAJ, INSPEC, COMPENDEX

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	EP 0 858 189 A (HITACHI LTD) 12 August 1998 (1998-08-12) column 6, line 35 -column 10, line 13 ---	1-27

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search 6 August 2002	Date of mailing of the international search report 20.08.2002
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Ströbeck, A.

2

Form PCT/ISA/210 (second sheet) (July 1992)

International Application No
PCT/US 01/04340

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	MURTHY ET AL: "Congestion-oriented shortest multipath routing" PROCEEDINGS OF IEEE INFOCOM 1996. CONFERENCE ON COMPUTER COMMUNICATIONS. FIFTEENTH ANNUAL JOINT CONFERENCE OF THE IEEE COMPUTER AND COMMUNICATIONS SOCIETIES. NETWORKING THE NEXT GENERATION. SAN FRANCISCO, MAR. 24 - 28, 1996, PROCEEDINGS OF INFOCOM, L, vol. 2 CONF. 15, 24 March 1996 (1996-03-24), pages 1028-1036, XP010158171 ISBN: 0-8186-7293-5 abstract page 1028, left-hand column, line 38 -right-hand column, line 29	1-27
E	WO 01 50688 A (TELEFONAKTIEBOLAGET LM ERICSSON (PUBL)) 12 July 2001 (2001-07-12) page 11, line 18 -page 13, line 21	28,29,34
A	WO 98 59470 A (TELEFONAKTIEBOLAGET LM ERICSSON (PUBL)) 30 December 1998 (1998-12-30) page 4, line 5 -page 5, line 2	28-39
X	WO 99 48303 A (CISCO TECHNOLOGY, INC.) 23 September 1999 (1999-09-23) page 1, line 8 -page 2, line 5 page 5, line 33 -page 6, line 15 page 7, line 21 - line 33	40,50
A		41-49, 51-59
A	JONES JIM ET AL: "Distributed Denial of Service Attacks: Defenses" INTERNET ARTICLE, 'Online! 2000, XP002208785 Retrieved from the Internet: <URL:www.bai.org/pdf/DDOS-defense.pdf > 'retrieved on 2002-08-05! paragraph '0005!	60-66
X	WO 99 38081 A (ASCEND COMMUNICATIONS INC) 29 July 1999 (1999-07-29) page 9, line 13 -page 10, line 17 page 11, line 10 -page 12, line 2	67
A		68-71

2

International application No.
PCT/US 01/04340

INTERNATIONAL SEARCH REPORT

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This International Search Report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. [ ] Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. [ ] Claims Nos.:
because they relate to parts of the International Application that do not comply with the prescribed requirements to such an extent that no meaningful International Search can be carried out, specifically:

3. [ ] Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

see additional sheet

1. [X] As all required additional search fees were timely paid by the applicant, this International Search Report covers all searchable claims.

2. [ ] As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.

3. [ ] As only some of the required additional search fees were timely paid by the applicant, this International Search Report covers only those claims for which fees were paid, specifically claims Nos.:

4. [ ] No required additional search fees were timely paid by the applicant. Consequently, this International Search Report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

[ ] The additional search fees were accompanied by the applicant's protest.

[X] No protest accompanied the payment of additional search fees.

Form PCT/ISA/210 (continuation of first sheet (1)) (July 1998)

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 210

This International Searching Authority found multiple (groups of) inventions in this international application, as follows:

1. Claims: 1-27

   A system and a method to balance the load between communication paths with varying transmission quality.

2. Claims: 28-39

   A system and a method to prevent someone from learning requested IP addresses by intercepting DNS requests.

3. Claims: 40-59

   A method to prevent a denial-of-service attack from an unauthenticated user flooding dummy data packets on to a low bandwidth link.

4. Claims: 60-66

   A method to prevent an authenticated user residing within a secure system from flooding it with dummy data packets.

5. Claims: 67-71

   A method to allocate memory in a central computer communicating with a potentially large number of client computers.

INT⁻⁻RNATIONAL SEARCH REPORT

ıııormation on patent family members

.ıternational Application No
PCT/US 01/04340

Patent document cited in search report		Publication date	Patent family member(s)			Publication date
EP 0858189	A	12-08-1998	JP	10224400	A	21-08-1998
			EP	0858189	A2	12-08-1998
			US	6112248	A	29-08-2000
WO 0150688	A	12-07-2001	SE	517217	C2	07-05-2002
			AU	2564501	A	16-07-2001
			WO	0150688	A1	12-07-2001
			SE	9904841	A	30-06-2001
			US	2001006523	A1	05-07-2001
WO 9859470	A	30-12-1998	AU	8052398	A	04-01-1999
			SE	9702385	A	24-12-1998
			WO	9859470	A2	30-12-1998
WO 9948303	A	23-09-1999	AU	3098299	A	11-10-1999
			WO	9948303	A2	23-09-1999
WO 9938081	A	29-07-1999	US	6055575	A	25-04-2000
			AU	2562599	A	09-08-1999
			CA	2318267	A1	29-07-1999
			EP	1064602	A1	03-01-2001
			WO	9938081	A1	29-07-1999

INTE(.ONAL SEARCH REPORT

ıational Application No
PCT/US 99/25323

COPY

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L29/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	FASBENDER A ET AL: "VARIABLE AND SCALABLE SECURITY: PROTECTION OF LOCATION INFORMATION IN MOBILE IP" IEEE VEHICULAR TECHNOLOGY CONFERENCE,US,NEW YORK, IEEE, vol. CONF. 46, 1996, pages 963-967, XP000593113 ISBN: 0-7803-3158-3 the whole document -----	1-63

☐ Further documents are listed in the continuation of box C.

☐ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search 20 July 2000	Date of mailing of the international search report 27/07/2000
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Canosa Aresté, C

3

Form PCT/ISA/210 (second sheet) (July 1992)

INTE   IONAL SEARCH REPORT

tional Application No
PCT/US 99/25325

COPY

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L29/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	FASBENDER A ET AL: "VARIABLE AND SCALABLE SECURITY: PROTECTION OF LOCATION INFORMATION IN MOBILE IP" IEEE VEHICULAR TECHNOLOGY CONFERENCE,US,NEW YORK, IEEE, vol. CONF. 46, 1996, pages 963-967, XP000593113 ISBN: 0-7803-3158-3 the whole document -----	1-67

☐ Further documents are listed in the continuation of box C.

☐ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search 20 July 2000	Date of mailing of the international search report 27/07/2000
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Canosa Aresté, C

1

Form PCT/ISA/210 (second sheet) (July 1992)

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 814 589 A3

(12) # EUROPEAN PATENT APPLICATION

(88) Date of publication A3:
30.08.2000 Bulletin 2000/35

(43) Date of publication A2:
29.12.1997 Bulletin 1997/52

(21) Application number: 97109792.8

(22) Date of filing: 16.06.1997

(51) Int. Cl.7: H04L 29/06, G06F 17/60, G07F 19/00

(84) Designated Contracting States:
AT BE CH DE DK ES FR GB GR IE IT LI LU MC NL PT SE

(30) Priority: 19.06.1996 US 667524

(71) Applicant: AT&T Corp.
New York, NY 10013-2412 (US)

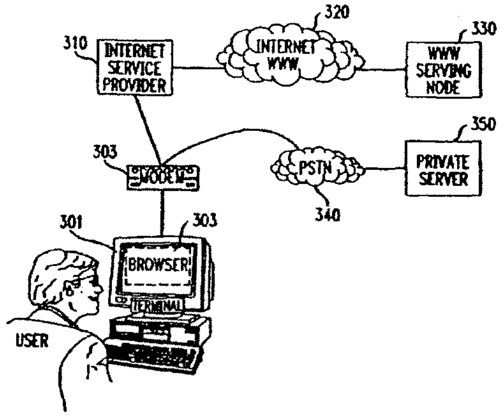
(72) Inventors:
- Harwood, Jonathan P.
Morganville, N.J. 07751 (US)
- Kimmeth, Thomas
Gladstone, N.J. 07977 (US)
- Nusbaum, Kurt
Downers Grove, Illinois 60515 (US)

(74) Representative: Kuhnen & Wacker
Patentanwaltsgesellschaft mbH,
Alois-Steinecker-Strasse 22
85354 Freising (DE)

(54) **System and method for automated network reconfiguration**

(57) A method is disclosed for providing an enhanced level of security for sensitive or proprietary information associated with information transactions in a public network, such as the Internet (101). In carrying out that method, an on-line information transaction is bifurcated between a generalized information access portion of such a transaction and an exchange of sensitive user information. With such a bifurcation, the generalized information access portion of the transaction, which generally would constitute the more substantial (in terms of network resources) portion of the transaction, would be handled via a non-secure network, usually a public network such as the Internet (320). The portion of the transaction involving sensitive user information, on the other hand, would be handled by a separate secure connection, such as a private network, or intranetwork (340). An important characteristic of this bifurcation arrangement is the provision of a means for automated reconfiguration of a user terminal as between accessing the Ageneralized information via the non-secure network and access to the secure communications network for the exchange of sensitive user information. Such an automated reconfiguration will be carried out without the necessity for any action on the part of the user, and indeed will be largely invisible to the user.

FIG. 3



EP 0 814 589 A3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 97 10 9792

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
A	EP 0 590 861 A (AMERICAN TELEPHONE & TELEGRAPH) 6 April 1994 (1994-04-06) * column 1, line 9 - line 24 * * column 3, line 15 - line 17 * * column 3, line 43 - column 4, line 7 *	1-30	H04L29/06 G06F17/60 G07F19/00
A	BAGSHAW E: "NET PROFITS" APRIL 1995 PC PRO, pages 176,178-182, XP002059701 ISSN: 1355-4603 * left-hand column, line 181, last paragraph *	1-30	
A	BELLARE M ET AL: "IKP - A FAMILY OF SECURE ELECTRONIC PAYMENT PROTOCOLS" PROCEEDINGS OF THE FIRST USENIX WORKSHOP ON ELECTRONIC COMMERCE, JULY 11-12, 1995, pages 89-106, XP000579445 * page 95, right-hand column *	1-30	
A	WO 96 00485 A (TELEFON AB LM ERICSSON) 4 January 1996 (1996-01-04) * page 6, line 1 - line 3; figure 1 * * page 10, line 13 - page 11, line 29 *	1-30	TECHNICAL FIELDS SEARCHED (Int.Cl.6) H04L G06F G07F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	5 July 2000	Vercauteren, S

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 97 10 9792

05-07-2000

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0590861 A	06-04-1994	CA 2100134 A	30-03-1994
		JP 7129671 A	19-05-1995
		MX 9305830 A	30-06-1994
		US 5485510 A	16-01-1996
WO 9600485 A	04-01-1996	US 5668876 A	16-09-1997
		AU 692881 B	18-06-1998
		AU 2688795 A	19-01-1996
		CA 2193819 A	04-01-1996
		EP 0766902 A	09-04-1997
		FI 965161 A	13-02-1997
		JP 10502195 T	24-02-1998

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

FIG. 1 (Prior Art)

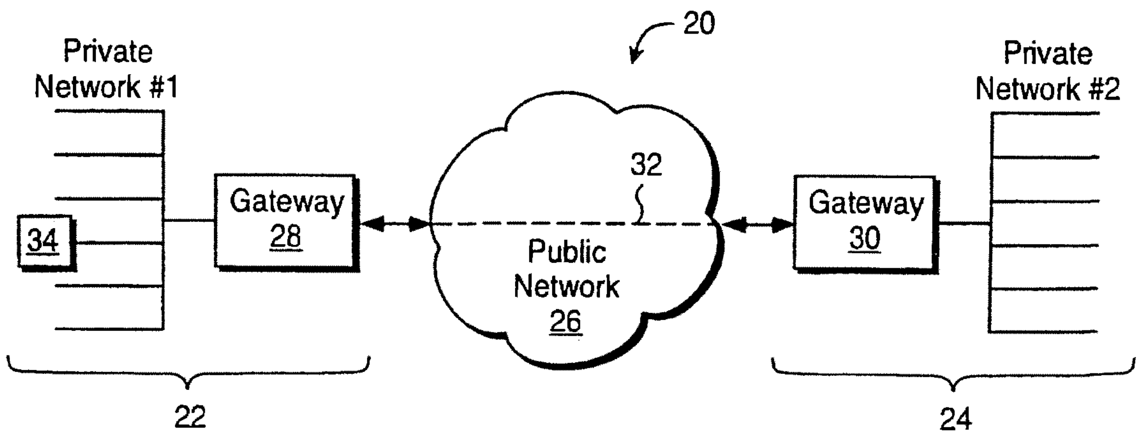


FIG. 2

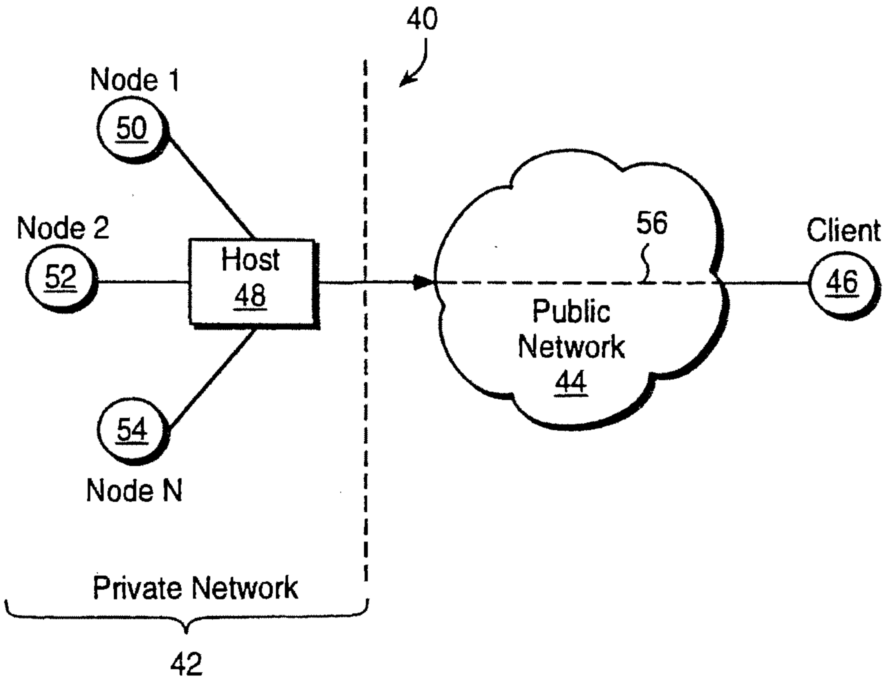
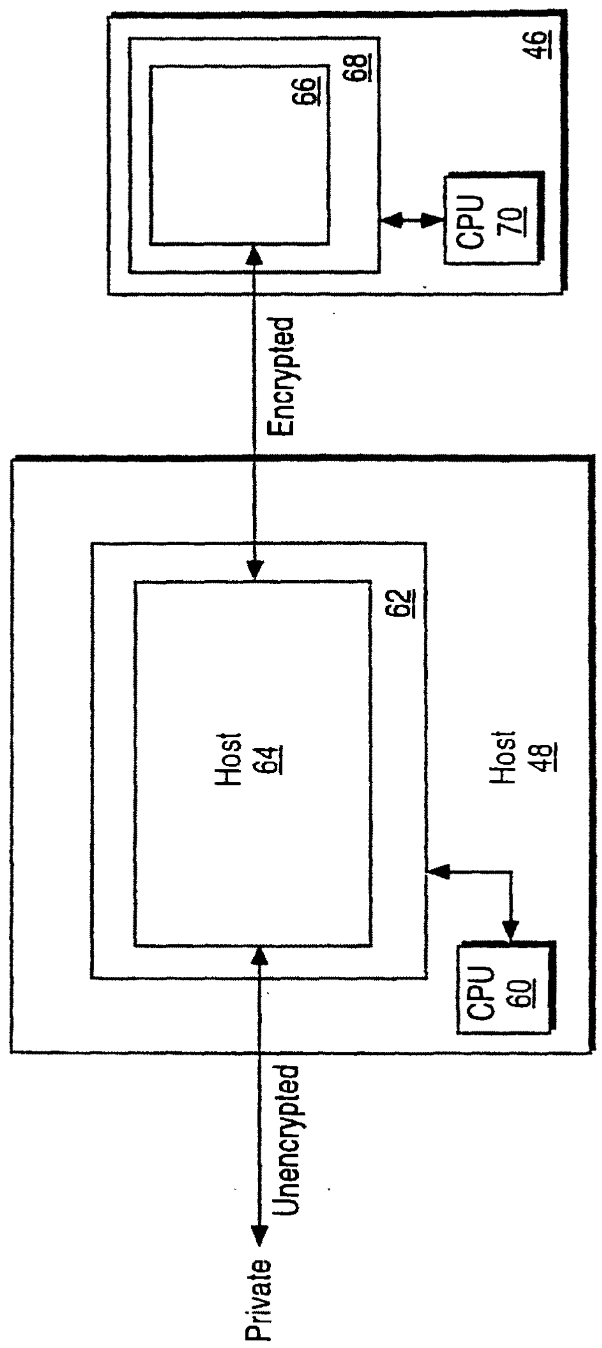
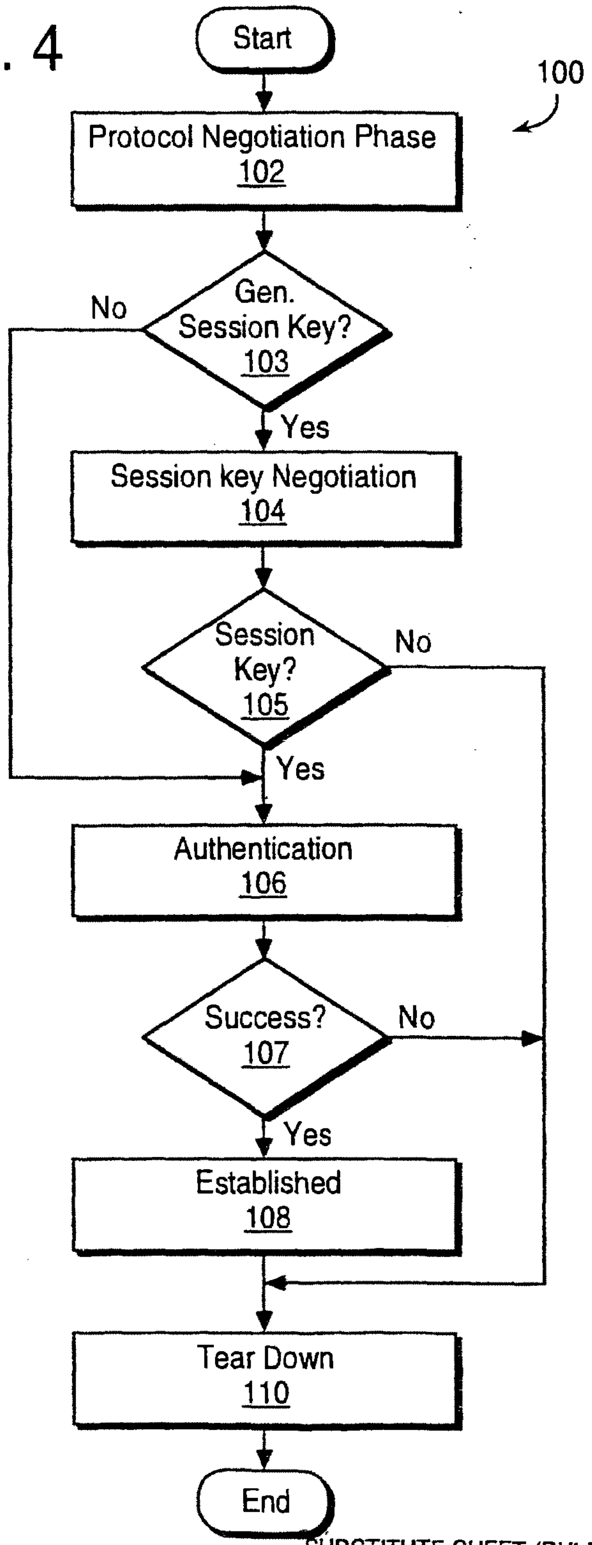


FIG. 3



SUBSTITUTE SHEET (RULE 26)

FIG. 4


Start
100
Protocol Negotiation Phase
102
Gen. Session Key?
103
No
Yes
Session key Negotiation
104
Session Key?
105
No
Yes
Authentication
106
Success?
107
No
Yes
Established
108
Tear Down
110
End

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/01583

A. CLASSIFICATION OF SUBJECT MATTER
IPC(6) :G06F 13/00; H04L 9/30
US CL : 709/245 , 229 ; 380/30
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 709/245 , 229, 228, 226; 380/30 , 23

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
APS, Internet
Search terms : private and public network, protocols, authentication, session key, encrypt, decrypt, encapsulation.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,416,842 A (AZIZ) 16 MAY 1995, Abstract, Figs. 2 and 5 and 6, col. 4 line 65 - col. 5 line 48, col. 6 lines 3-35 and lines 40-51, col. 7 lines 16-35, col. 7 line 63 - col. 8 line 2	1-14
Y	US 5,550,984 A (GELB) 27 AUGUST 1996, Abstract, Fig. 1, col. 5 line 45 - col. 6 line 51, col. 7 line 58 - col. 8 line 19	1-14
Y	US 5,548,646 A (AZIZ ET. AL.) 20 AUGUST 1996, Abstract, Figs. 5 and 6, col. 9 lines 1-50, col. 10 line 32 - col. 11 line 67	1-14
Y,P	US 5,835,726 A (SHWED ET. AL.) 10 November 1998, Abstract, Fig. 21, col. 20 line 41 - col. 21 line 7, col. 22 line 19 - col. 23 line 9.	1-14

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 21 MAY 1999	Date of mailing of the international search report 02 JUN 1999
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer AHMAD MATAR [illegible] Telephone No. (703) 305-4731

Form PCT/ISA/210 (second sheet)(July 1992) ★

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 814 589 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
29.12.1997 Bulletin 1997/52

(51) Int. Cl.^{6}: H04L 29/06

(21) Application number: 97109792.8

(22) Date of filing: 16.06.1997

(84) Designated Contracting States:
AT BE CH DE DK ES FR GB GR IE IT LI LU MC NL PT SE

(30) Priority: 19.06.1996 US 667524

(71) Applicant: AT&T Corp.
New York, NY 10013-2412 (US)

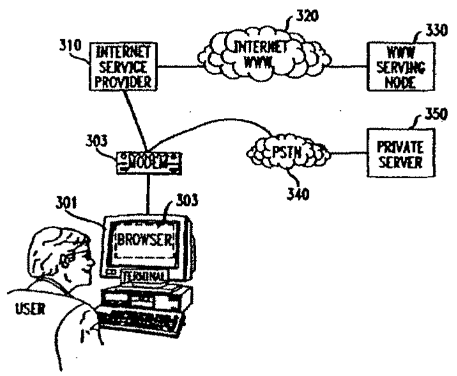
(72) Inventors:
- Harwood, Jonathan P.
  Morganville, N.J. 07751 (US)
- Kimmeth, Thomas
  Gladstone, N.J. 07977 (US)
- Nusbaum, Kurt
  Downers Grove, Illinois 60515 (US)

(74) Representative:
KUHNEN, WACKER & PARTNER
Alois-Steinecker-Strasse 22
85354 Freising (DE)

(54) System and method for automated network reconfiguration

(57) A method is disclosed for providing an enhanced level of security for sensitive or proprietary information associated with information transactions in a public network, such as the Internet. In carrying out that method, an on-line information transaction is bifurcated between a generalized information access portion of such a transaction and an exchange of sensitive user information. With such a bifurcation, the generalized information access portion of the transaction, which generally would constitute the more substantial (in terms of network resources) portion of the transaction, would be handled via a non-secure network, usually a public network such as the Internet. The portion of the transaction involving sensitive user information, on the other hand, would be handled by a separate secure connection, such as a private network, or intranetwork. An important characteristic of this bifurcation arrangement is the provision of a means for automated reconfiguration of a user terminal as between accessing the Ageneralized information via the non-secure network and access to the secure communications network for the exchange of sensitive user information. Such an automated reconfiguration will be carried out without the necessity for any action on the part of the user, and indeed will be largely invisible to the user.

FIG. 3



EP 0 814 589 A2

Description

FIELD OF THE INVENTION

This invention is related to the field of data communications, and more particularly to a method and means for establishing an automatic reconfiguration of a user terminal among alternative tasks.

BACKGROUND OF THE INVENTION

With the increasing popularity of personal computers over the last several years has come a striking growth in transaction-oriented computer-to-computer communications (as opposed to bulk-data transfers among such computers). For convenience herein such transaction-oriented computer-to-computer communications will be described by the shorthand term "information transaction". That growth in the use of computers for such information transactions has unquestionably been fueled by the existence of an international infrastructure for implementing such data communications, known as the Internet. And, driven by the burgeoning demand for such information transaction services, the Internet has itself experienced explosive growth in the amount of traffic handled.

At least partly in response to that demand, a new level of accessibility to various information sources has recently been introduced to the Internet, known as the World Wide Web ("WWW"). The WWW allows a user to access a universe of information which combines text, audio, graphics and animation within a hypermedia document. Links are contained within a WWW document which allow simple and rapid access to related documents. Using a system known as the HyperText Markup Language ("HTML"), pages of information in the WWW contain pointers to other pages, those pointers typically being a key word (commonly known as a hyperlink word). When a user selects one of those key words, a hyperlink is created to another information layer (which may be in the same, or a different information server), where typically additional detail related to that key word will be found.

In order to facilitate implementation of the WWW on the Internet, new software tools have been developed for user terminals, usually known as Web Browsers, which provide a user with a graphical user interface means for accessing information on the Web, and navigating among information layers therein. A commonly used such Web Browser is that provided by Netscape.

The substantial growth in the use of computer networks, and particularly the WWW, for such information transactions, has predictably led to significant commercialization of this communications medium. For example, with the WWW, a user is not only able to access numerous information sources, some public and some commercial, but is also able to access "catalogs" of merchandise, where individual items from such a catalog can be identified and ordered, and is able to carry out a number of banking and other financial transactions. As will be obvious, such commercial transactions will typically involve sensitive and proprietary information, such as credit card numbers and financial information of a user. Thus, with the growth of commercial activity in the Internet, has also come a heightened concern with security.

It is well known that there are persons with a high level of skill in the computer arts, commonly known as "hackers", who have both the ability and the will to intercept communications via the Internet. Such persons are thereby able to gain unauthorized access to various sensitive user information, potentially compromising or misappropriating such information.

The vulnerability of such sensitive user information to misuse when so transmitted via the Internet is a phenomena which has only recently received wide public attention. Unless such security concerns can be quickly addressed and alleviated, the commercial development of this new communications medium may be slowed or even stalled altogether.

SUMMARY OF THE INVENTION

Accordingly, it is an object of the invention to provide an acceptable level of security for sensitive or proprietary information associated with information transactions in a public network, such as the Internet. That object is realized through an arrangement whereby an on-line information transaction is bifurcated between a generalized information access portion of such a transaction and an exchange of sensitive user information. With such a bifurcation, the generalized information access portion of the transaction, which generally would constitute the more substantial (in terms of network resources) portion of the transaction would be handled via a non-secure network, usually a public network such as the Internet. The portion of the transaction involving sensitive user information, on the other hand, would be handled by a separate secure connection, such as a private network, or intranetwork. An important characteristic of this bifurcation arrangement is the provision of a means for automated reconfiguration of a user terminal as between accessing the generalized information via the non-secure network and access to the secure communications network for the exchange of sensitive user information. Such an automated reconfiguration will be carried out without the necessity for any action on the part of the user, and indeed will be largely invisible to the user. In a further embodiment of the invention, a transfer of data is provided from a public to a private network, wherein data selected by a user from a public network site may be arranged and displayed at a user terminal and, subject to further user selection/confirmation activity, thereafter transferred to a private network.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 depicts an illustrative case of information transactions carried out via a public network such as the Internet.
Figure 2 shows the architecture of a browser as would typically be applied for accessing a hypermedia web page.
Figure 3 illustrates the primary elements of the reconfigurable dual-path method of the invention.
Figure 4 depicts in flow chart form the basic jump capability of the methodology of the invention.
Figures 5A & 5B (generally designated collectively herein as "Figure 5") depict in flow chart form the "shopping cart" capability of the methodology of the invention.
Figure 6A & 6B (generally designated collectively herein as "Figure 6") depict in flow chart form the stored configuration capability of the methodology of the invention.
Figure 7A & 7B (generally designated collectively herein as "Figure 7") depict in flow chart form the off-line form capability of the methodology of the invention.

DETAILED DESCRIPTION

For clarity of explanation, the illustrative embodiment of the present invention is presented as comprising individual functional blocks. The functions these blocks represent may be provided through the use of either shared or dedicated hardware, including, but not limited to, hardware capable of executing software.

Figure 1 depicts an illustrative case of information transactions carried out via the Internet. As seen in the figure, an exemplary user obtains access to the Internet by First connecting, via a Terminal **110** having an associated Browser **111**, to an Internet Service Provider **112** selected by the user. That connection between the user and the Internet Service Provider will typically be made via the Public Switched Telephone Network (PSTN) from a modem associated with the user's Terminal to a network node in the Internet maintained by the selected Internet Service Provider.

Once the user has obtained access to the selected Internet Service Provider, an address is provided for connection to another user or other termination site and such a connection is made via the Internet to that destination location. As can be seen from the figure, communication via the Internet may be either user-to-user, as from Terminal **110** to Terminal **130**, or from a user to a node representing an information source accessed via the Internet, such as Public Site **120**.

It will of course be understood that the Internet provides service to a large number of users and includes a large number of such Public Sites, but the illustration provides the essential idea of the communication paths established for such Internet communication. It will also be understood that a number of service classifications are supported by the Internet, with the World Wide Web service, which represents a preferred embodiment for the public network aspect of the method of the invention, being one of the currently most heavily trafficked of such services.

The Web Browser, such as depicted at **111**, can be seen as a software application operating in conjunction with a user terminal (such as Terminal **110**) which provides an interface between such a user terminal and the particular functionality of the WWW information site. The architecture of such a browser is generally described in terms of three main components, as illustrated in Figure 2. At the top level is the Browser **201**, which enables the acquisition of information pages from a WWW server (beginning, in all cases, with the "home page" for that server), for display at a display device associated with the terminal. The Browser also provides the necessary interface for the terminal with the HTML functionality used by the server to provide access to other linked information layers.

The second level of the browser architecture is the TCP/IP Stack **202**, which handles the communications protocols used for connecting the terminal to the WWW server. The bottom level of this architecture is the Dialer **203**, which typically handles the function of providing dialing and setup digits to a modem, as illustrated at **204**, such a modem generally being a part of the terminal. Normally, upon receiving dialing and other setup information from the dialer, the modem would cause a connection to be made via the PSTN to the Internet Service Provider selected for that terminal.

After a connection is established in this manner to the Internet Service Provider, an address would be provided for the WWW information node sought to be contacted, a connection to that node made through the Internet, and the home page for that node caused to be displayed at the terminal's display device. A user would then select a key word in that home page, typically by clicking on the word with a mouse or similar device, and, upon transmission of that selection signal to the WWW server, a hyperlink would be created to the linked information layer and the open page of that layer would be caused to be displayed at the user terminal.

As explained above, serious questions have been raised in respect to the security of communications via the public Internet. (Note, that the discussion herein is focused on the Internet, and particularly the WWW functionality of the Internet, as a preferred embodiment of such public data communication networks generally, but the methodology of the invention will be applicable to any such network.) To address this problem, the methodology of the invention begins with a bifurcation of the information transaction between a user and the selected information transaction provider into a portion related to sensitive or proprietary user information, and other information comprising that transaction. With such a bifurcation, it becomes possible to provide substantial security for that proprietary information by use of an alternative communications path for that separated portion of the transaction via a private network, or intranetwork -- *i.e.*, a connection between a user's terminal and a secure serving node on that private network. It is anticipated that a coordination means will be established in respect to the management of information among the public and private network elements of the bifurcated information transaction.

In its basic form, this methodology may be carried out by the user terminal initiating a call via the Internet to a selected WWW node, and upon establishing connection to that node, proceeding with the desired information transaction up to the point where an exchange of sensitive or proprietary information were required. At that point the user terminal would be instructed by the WWW server to terminate that connection (*i.e.*, hangup) and to place a new call to an identified private network server for the necessary exchange of sensitive information.

However, in order to accomplish such a dual-path transaction, it is necessary that the browser at the user terminal be reconfigured to provide the dialing, authorization (*i.e.*, login and password), and other needed information for accessing the alternative private network, in order to implement the proprietary portion of the transaction. It will also usually be the case that, upon completion of that private-network transaction, the original dialer, stack and browser configurations will need to be restored, in order for the terminal to retain its normal Internet access functionality. Such a reconfiguration and subsequent restoral of the necessary parameters in the browser, stack and dialer is likely to be well beyond the capabilities of the average user.

Accordingly, as a further embodiment of the inventive methodology, an automated browser reconfiguration means is provided which interoperates with the browser. This browser reconfiguration means is described in detail hereafter and will be referred to as the "Bridging Software".

Figure 3 provides an illustration of the primary elements of the reconfigurable dual-path method of the invention. As seen in the figure, a first path comparable to the Internet link shown in Figure 1, between User Terminal **301** and WWW Serving Node **330** (via Browser **302**, Modem **303**, Internet Service Provider **310**, and Internet **320**) is provided. However, an alternative path is now provided from the output of Modem **303** to Private Server **350**. That path is illustrated as being via the PSTN, which is generally regarded as being highly secure, but an alternative dedicated or other more-secure path between the User Terminal **301** and the Private Server **350** could as well be provided. In keeping with the discussion above, Browser **302** shown in Figure 3 would also include the Bridging Software installed as a helper application for implementing the automatic reconfiguration of the Browser.

In the operation of this system, a user would normally make an initial connection to an Internet application, such as the application represented by WWW Serving Node **330**, which, *e.g.*, might be a shopping application, a financial transaction, or the provision of an enrollment form for off-line preparation. After conducting all, or some portion of an information transaction short of an exchange of sensitive or proprietary information, including a capture by the user's terminal of needed information from the public site, a user provides a signal indicative of an end to that portion of that transaction. During the course of the public portion of the information transaction, specially configured files are sent from the WWW serving node to the Bridging Software associated with Browser **302**. Such files contain instructions for the Bridging Software to store information-like products -- e.g., for selected items from a catalog, forms for enrollment, or non-secure portions of a financial transaction, and reconfiguration information for dialing and logging into the private portion of the transaction. The Bridging Software then hangs up the Internet connection, edits the user terminal's browser, stack and dialer files to reconfigure the terminal to connect to the private server. Prior to automatic redialing of the new private site for the user, the Bridging Software may be instructed by the application operating at WWW Server Node **330** to display items chosen for purchase, or to display a form for the end-user to complete off-line before dialing the private application. Upon connecting to the private application and completing the transaction as to the user sensitive information in a private environment, the Bridging Software then restores the end-user software to the dialing and authorization parameters required to dial to the public Internet.

A particularly advantageous application of the automated reconfiguration and information transfer methodology of the Bridging Software is that it adds value to certain WWW servers which do not possess the Common Gateway Interface ("CGI") capability -- *i.e.*, a provision of specialized functions on the server beyond just displaying HTML files, and are accordingly unable to accomplish any transactional processing in respect to items selected by a user. In effect, such a non-CGI server, on its own, can only serve as a "billboard" for the items represented in its database.

However, with the collection and redelivery process of the Bridging Software, a data capture and processing mechanism can be implemented for servers operating in a non-CGI environment -- such servers being incapable of more than the simple delivery of static data packets corresponding to available items. The data set enabled by the Bridging Software is a mechanism for augmenting such limited server capabilities by defining a flexible mechanism for the receipt, display, and delivery of arbitrary data from one site to another.

In such a scenario, the Bridging Software receives a "shopping cart" item list from the host as a data-set defined with a static MIME data packet associated with the Bridging Software. This information comprising the data-set may be updated, displayed to the user in a "read-only" fashion, or presented to the user for order selection.

During the process of interacting with the WWW server, a user may trigger HTML links resulting in additional MIME packets for the Bridging Software being delivered to the client. These packets allow items to be added and/or removed from the specified data set or presented to the user for local confirmation. The user will interact with a pop-up screen provided by the Bridging Software which presents the items available with product information, such as part number, description, unit cost, etc. The user identifies those items which are to be placed into the "shopping cart" and the quantity of items desired. Upon completion of the form, the Bridging Software stores the order in a format suitable for subsequent delivery to the private server site.

An additional feature provided by the methodology of the Bridging Software is an automated mechanism for providing compatibility with user terminals not previously having the Bridging Software included with the terminal's browser. To that end, the Bridging Software located at an accessed public network site initially checks to see if the browser counterpart for that software is loaded at the calling user terminal. If yes, the heretofore described processes of the Bridging Software go forward. If not however, a request is sent through the public host to download the Bridging Software to the calling terminal. After such a download, a helper application loads the Bridging Software to the terminal's browser.

I. Illustrative Embodiments

A variety of browser reconfiguration applications are supported by the automated browser reconfiguration means of the invention. Four essentially diverse capabilities of this invention, which support such applications, are described hereafter as illustrative embodiments of the invention.

A. Basic Jump Capabilities

In this configuration, which is illustrated in flow chart form in Figure 4, an end-user is connected to a chosen WWW serving node (where a desired information product is made available) via a modem and an Internet browser associated with the user's terminal (**Step 401** of Figure 4). After conducting an information transaction with the selected WWW serving node for some interval (determined in relation to the specific application accessed), the user clicks on a hypertext link, or picture, to begin an automated process which will cause that public session to be terminated and a new connection established to an alternate private data network (**Step 402**).

In response to that user action, a data message containing parameter reconfiguration instructions is passed from the WWW server application to the Bridging Software at the user's terminal (**Step 403**). Upon receiving such instructions, the Bridging Software edits the user's on-line communications software parameters, reconfiguring that software to dial the alternate data network (**Step 404**). This reconfiguration is fully automatic and transparent to the user, and includes parameters such as modem dial number, login, password, and TCP/IP addresses. At that point, the Bridging Software causes the modem to disconnect the current data network connection, shutting down the browser, and to then dial the alternate private data network (**Step 405**).

With the establishment of a connection to the private server on the alternate data network, the user interacts with the alternate data network application as appropriate (**Step 406**), and after an interval completes his activity with the alternate data network and provides an indication of such completion (**Step 407**). A data message containing parameter reconfiguration instructions is then passed from the alternate data network application to the Bridging Software (**Step 408**).

At that point, the Bridging Software again edits the user's on-line communications software parameters, reconfiguring them to dial the original public data network, or another preselected network (**Step 409**). As with the first reconfiguration, this configuration is automatic and includes parameters such as modem dial number, login, password, and TCP/IP addresses. The Bridging Software automatically causes the current private data network to be disconnected by the modem (**Step 410**), and if appropriate, causes the original public data network to be redialed (**Step 411**). When such a reconnection to the public data network is established, the end-user would then continue his application in the public data network.

B. "Shopping Cart" Capability

With this configuration, illustrated in flow chart form in Figure 5, a user begins by establishing a connection to a WWW application (assuming for the moment that the application is non-CGI enabled) at a serving node for that application, using the Internet browser and modem associated with the user's terminal (**Step 501** of Figure 5). Upon finding an item in that application to be saved, or remembered for later consideration, or purchase, the user clicks on a hypertext link, or picture, representing that item (**Step 502**). That application then sends a data message to the Bridging Software containing information about the items selected (**Step 503**) and such information is stored by the Bridging Software in the "shopping cart" file in the user's terminal (**Step 504**). Such selection download and storage steps (*i.e.*, steps 502, 503 & 504) are repeated for as many items as the user chooses to select. At any point after the Bridging Software has received the first set of item selection information, the user can instruct the Bridging Software to cause those selected items about which such information has been received to be displayed locally (at the user's terminal), where the user may review or edit (including deletion if desired) the collection of items theretofore selected. The application may also control display characteristics such as color and font for such locally displayed items. Note that in the case of a CGI-enabled application, the application itself will keep track of the items selected by the user and only download the totality of the selected items at the end of the selection process, and accordingly, the described local display option will not be applicable to such a CGI-enabled application.

At the point of completion of his "shopping", the user clicks on a hyper-text link or picture to "check out" (**Step 505**), which will begin a process of causing a jump to an alternate data network for the completion of sensitive portions of the transaction. To that end, a data message containing parameter reconfiguration instructions is passed from the WWW application to the Bridging Software (**Step 506**). It is to be noted that, as a security measure, information such as the new dial number, IP address, home page, configuration data (*e.g.*, login, password, DNS address) may be passed over the public network in encrypted form.

Upon receiving such reconfiguration instructions, the Bridging Software edits the user's on-line communications software parameters, reconfiguring that software to dial the alternate data network (**Step 507**). This reconfiguration is fully automatic and transparent to the user, and includes parameters such as modem dial number, login, password, and TCP/IP addresses. At that point, the Bridging Software causes the modem to disconnect the current data network connection, shutting down the browser, and to then dial the alternate data network (**Step 508**).

The Bridging Software passes the stored "shopping cart" data captured from the WWW application to the alternate network application (**Step 509**), where that data may be displayed for the user, permitting the user to confirm and/or modify the data (**Step 510**). The user interacts with the alternate data network application as appropriate, and after an interval completes his activity with the alternate data network (**Step 511**) and thus, by providing an appropriate completion signal to the application, completing the private portion of the information transaction (**Step 512**). A data message containing parameter reconfiguration instructions is then passed from the alternate data network application to the Bridging Software (**Step 513**).

The Bridging Software, at this point, again edits the user's on-line communications software parameters, reconfiguring them to dial the original (or another pre-defined) data network (**Step 514**). As with the first reconfiguration, this configuration is automatic and includes parameters such as modem dial number, login, password, and TCP/IP addresses. The Bridging Software automatically causes the current private data network to be disconnected by the modem (**Step 515**), and if appropriate, causes the original public data network to be redialed (**Step 516**). When such a reconnection is established to the point in the public data network where the user had left off to handle the secured aspects of his information transaction, the user would then continue his application in the public data network.

C. <u>Stored Configuration Capabilities</u>

For this configuration, depicted in flow chart form in Figure 6, an end-user is connected to a chosen WWW serving node (where a desired information product is made available) via a modem and an Internet browser associated with the user's terminal (**Step 601** of Figure 6). The user selects a hypertext link or picture associated with the WWW application by clicking on such link or picture (**Step 602**). A data message containing parameter reconfiguration instructions and an application icon (related to the selected hypertext link or picture) is passed from the WWW application to the Bridging Software (**Step 603**).

The Bridging Software creates an icon for display at the user's terminal, and saves a Bridging Software configuration file that is associated with that icon (**Step 604**). Such Bridging Software actions are automatic and multiple selections may he captured in this manner. At this point the user may continue the on-line session, or, if all desired selections have been made, a signal is provided from the user that the session should be discontinued (**Step 605**). The Bridging Software then automatically disconnects the current data network connection (**Step 606**).

After disconnecting from the WWW application, and following an interval determined by the user, a new application is selected by the user by clicking on the appropriate new icon displayed at the user's terminal (**Step 607**). The Bridging Software receives the reconfiguration instructions from the file associated with the selected icon (**Step 608**).

The Bridging Software edits the user's on-line communications software parameters, reconfiguring that software to dial the alternate data network (**Step 609**). The Bridging Software then automatically starts the user's Internet browser software and causes the alternate network application to be dialed by the modem associated with that terminal (**Step 610**). Upon establishing a connection to the alternate network, the user interacts with that application and completes the transaction to the user's satisfaction (**Step 611**). After a signal is sent to the alternate network indicating such completion of the user's activity (**Step 612**), a data message containing parameter reconfiguration instructions is passed from the alternate data network application to the Bridging Software (**Step 613**). That Software then causes the user's terminal configuration parameters to be reset (Step 614) and the alternate data network to be automatically disconnected (Step 615).

D. Off-Line Form Capability

In this configuration, depicted in flow chart form in Figure 7, an end-user is connected to a chosen WWW serving node (where a desired information product is made available) via a modem and an Internet browser associated with the user's terminal (Step 701 of Figure 7). The user selects a hypertext link or picture associated with an off-line form application -- an exemplary such form being an HTML-based form -- by clicking on such link or picture (Step 702). A data message containing parameter reconfiguration instructions for the Bridging Software, the selected off-line-form application, and an optional icon (related to the selected hypertext link or picture) is passed from the WWW application to the Bridging Software (Step 703). Note that the selected off-line form may be for either single or multiple use.

In the case of a delayed or multiple use of the selected form, the Bridging Software may create an icon for display at the user's terminal, and will save a Bridging Software configuration file that is associated with that icon (Step 704). The form in question is also saved on the user's terminal. Such Bridging Software actions are automatic. At this point the user may continue the on-line session, or, if all desired selections have been made, a signal is provided from the user that the session should be discontinued (Step 705). The Bridging Software then automatically disconnects the current data network connection (Step 706).

After disconnecting from the WWW application, two cases are to be considered as to the further processing of the selected form: (1) an immediate single use of the form and (2) either a delayed or multiple use of the form. In the first case, the Bridging Software edits the user's on-line communications software parameters, reconfiguring that software to dial the alternate data network. The Bridging Software then automatically starts the user's Internet browser software which is caused to display the off-line form. The user then completes the off-line form and chooses a "Submit Form" button displayed at his terminal.

In the second case, the Bridging Software will have created an icon for display at the user's terminal and saved a Bridging Software configuration file associated with that icon. Following an interval determined by the user, the off-line-form application is started by the user by clicking on the new form icon displayed at the user's terminal (Step 707). The Bridging Software receives the reconfiguration instructions from the file associated with the selected icon (Step 708).

The Bridging Software edits the user's on-line communications software parameters, reconfiguring that software to dial the alternate data network (Step 709). The Bridging Software then automatically starts the user's Internet browser software which is caused to display the off-line form (Step 710). The user then completes the off-line form and chooses a "Submit Form" button displayed at his terminal (Step 711).

In either the first or second case, following activation of the "Submit Form" button, the alternate network application is then caused to be dialed by the Bridging Software. Upon establishing a connection to the alternate network, the form data is passed to the alternate network (Step 712). The user then interacts with that application and completes the application (Step 713). After a signal is sent to the alternate network indicating such completion of the user's activity (Step 714),a data message containing parameter reconfiguration instructions is passed from the alternate data network application to the Bridging Software (Step 715). That Software then causes the user's terminal configuration parameters to be reset (Step 716) and the alternate data network to be automatically disconnected (Step 717).

CONCLUSION

A system and method has been described for the automatic switching of an information transaction between two or more alternate networks. This functionality, which incorporates a reconfiguration means designated herein as the Bridging Software, supports the movement of application specific data from one on-line environment to another. Among potential applications of this process for passing data between different environments are: selected items for purchase ("shopping cart"), captured data from forms, and other server captured data such as web pages visited.

The Bridging Software reconfiguration means is intended to work with various Web Browser software implementations, including the Netscape Personal Edition (NPE) Software for Windows 3.1 and 3.11, and which represents a working embodiment for the invention. The Bridging Software installs itself as a helper application within the browser application and utilizes a special MIME type configuration file to pass reconfiguration and "shopping cart" information from the server to the client software.

When an application requires a user to re-connect to a private application, a reconfiguration file is passed to the Bridging Software helper application via a CGI script or simple hyper-text link. The helper application disconnects the current data connection, reconfigures the dial parameters (dial #, login password, DNS address, and home page) and initiates the dial program so the end-user can access the private application.

When the end-user connects to the private application, the Bridging Software reconfiguration means provides the new "private server" application with data collected from the "public server", and the application resumes in a private, secure environment.

The Bridging Software allows both short term and long term storage of dial configurations. Configurations passed to the Bridging Software can be designated as single use configurations and discarded after the application has terminated, or saved and displayed to the end-user as a dial choice by the Bridging Software.

Although the present embodiment of the invention has been described in detail, it should be understood that various changes, alterations and substitutions can be made therein without departing from the spirit and scope of the invention as defined by the appended claims. In particular, it is noted that, while the invention has been primarily described in terms of a preferred embodiment based on an automatic reconfiguration between a public and a private data network, any the methodology of the invention will be equally applicable to any set of alternate networks.

Claims

1. A method for managing a transaction via a communications path between a terminal device and a serving node in a data network, said method comprising the steps of:

   establishing an initial communications path via a first connection between said terminal device and a serving node in a first data network;
   receiving information from said serving node in said first data network for effecting a reconfiguration of said communications path for said transaction from said first connection in said first data network to a second connection in a second data network; and
   automatically connecting said terminal device to a serving node in said second data network via said second connection.

2. A method for managing a transaction via a communications path between a terminal device and a serving node in a data network, said method comprising the steps of:

   establishing an initial communications path via a first connection between said terminal device and a serving node in a first data network;
   selecting at least one information item from a data base of said information items provided at said serving node in said first data network;
   causing said selected information items to be downloaded to said terminal device via said first connection;
   receiving information from said serving node in said first data network for effecting a reconfiguration of said communications path for said transaction from said first connection in said first data network to a second connection in a second data network; and
   automatically connecting said terminal device to a serving node in said second data network via said second connection.

3. A method for managing a transaction via a communications path between a terminal device and a serving node in a data network, said method comprising the steps of:

   establishing an initial communications path via a first connection between said terminal device and a serving node in a first data network;
   identifying at least one data network application from a data base of said data network applications provided at said serving node in said first data network;
   receiving information from said serving node in said first data network for reconfiguring said terminal device for implementation of a communication path via an alternate connection between said terminal device and at least one of said identified data network applications in a second data network; and
   in response to a selection signal from a user, automatically connecting said terminal device to a selected one of said identified data network applications via said alternate connection.

4. A method for managing a transaction via a communications path between a terminal device and a serving node in a data network, said method comprising the steps of:

   establishing an initial communications path via a first connection between said terminal device and a serving node in a first data network;
   selecting an off-line form application from a data base provided at said serving node in said first data network;
   receiving information from said serving node in said first data network for reconfiguring said terminal device for implementation of a communication path via a second connection between said terminal device and said

selected off-line form application in a second data network; and
in response to, a selection signal from a user, automatically connecting said terminal device to said selected off-line form application.

5. The method for managing a transaction of Claim 1 or 2 including the further step of recognizing a signal to reconfigure said communications path from said first connection to said second connection.

6. The method for managing a transaction of Claim 3 wherein said selected data network application is operated at a serving node in said second data network.

7. The method for managing a transaction of Claim 4 wherein said selected off-line form application is operated at a serving node in said second data network.

8. The method for managing a transaction of one of the Claims 1, 2, 6 or 7 wherein said serving nodes in said first and said second data networks are manifested in a common node.

9. The method for managing a transaction of Claim 1 or 2 wherein said step of receiving information includes the further step of effecting said reconfiguration of said communications path.

10. The method for managing a transaction of Claim 1 or 2 wherein said step of automatically connecting includes the step of automatically disconnecting said first connection prior to implementation of said second connection.

11. The method for managing a transaction of Claim 1 or 2 including the further steps of:

automatically disconnecting said second connection in response to a user signal; and
reconfiguring said terminal device to enable, in response to user instruction, an implementation of a connection via an identified data network.

12. The method for managing a transaction of Claim 11 wherein said step of automatically reconfiguring said terminal device includes the step of effecting said implementation of said connection via said identified data network.

13. The method for managing a transaction of Claim 2 wherein said step of causing said selected information items to be downloaded includes the further step of causing said selected information items to be displayed at said terminal device.

14. The method for managing a transaction of Claim 13 wherein said displayed selected items can be edited by a user at said terminal device.

15. The method for managing a transaction of Claim 13 wherein display characteristics for said displayed selected items can be controlled at said terminal device.

16. The method for managing a transaction of Claim 2 wherein said step of automatically connecting includes the step of uploading said selected information items from said terminal device to said service provider via said second connection.

17. The method for managing a transaction of Claim 3 including the further steps of:

automatically disconnecting said alternate connection in response to a user signal; and
reconfiguring said terminal device to enable implementation of a pre-selected connection between said terminal device and an identified data network.

18. The method for managing a transaction of Claim 17 wherein said step of automatically reconfiguring said terminal device includes the further step of effecting said implementation of said pre-selected connection.

19. The method for managing a transaction of Claim 4 including the further step of downloading from said serving node in said first data network to said terminal device of an off-line form related to said off-line form application.

20. The method for managing a transaction of Claim 4 including the further step of uploading said downloaded off-line

form from said terminal device to said selected off-line form application, after processing by a user.

21. The method for managing a transaction of Claim 4 including the further steps of:

automatically disconnecting said connection to said selected off-line form application in response to a user signal; and
reconfiguring said terminal device to enable implementation of a pre-selected connection between said terminal device and an identified data network.

22. The method for managing a transaction of Claim 21 wherein said step of automatically reconfiguring said terminal device includes the further step of effecting said implementation of said pre-selected connection.

23. A method for managing connections between a terminal device and at least one information source/processor wherein at least two of said connections are implemented via separate communications networks, comprising the steps of:

recognizing a signal for connection to an information source/processor via a communications network other than a communications network for which a predetermined connection is configured;
causing said terminal device to implement a connection to said information source/processor via said other communications network; and
upon termination of said information source/processor connection via said other communications network, automatically reconfiguring a connection criteria in said terminal device to enable said terminal device to implement, in response to user instruction, a connection via an alternative one of said communications networks.

24. The method for managing connections of Claim 23 wherein said recognizing step occurs at a point when said terminal device is connected to a given source/processor.

25. The method for managing connections of Claim 23 wherein information items may be selected by a user at said terminal device from said given source/processor, and including the further step of causing said selected information items to be downloaded from said source/processor to said terminal device.

26. The method for managing connections of Claim 25 wherein said step of effecting connection includes the further step of uploading said selected information items from said terminal device to said other information source/processor.

27. The method for managing connections of Claim 26 wherein said selected information items are processed by said user at said terminal device prior to uploading to said other information source/processor.

28. The method for managing connections of Claim 24 including the further step of causing said given source/processor to download to said terminal device configuration data for enabling said step of effecting connection to said other information source/processor.

29. The method for managing connections of Claim 24 including the further step of causing said other source/processor to download to said terminal device configuration data for enabling said step of automatically restoring a prior connection criteria in said terminal device.

30. A method for enhancing security of certain data in an on-line information transaction comprising the steps of:

bifurcating said information transaction into a first portion comprising said certain data and a remaining portion, wherein said remaining portion is carried out via a public on-line communications connection between a terminal device and a public information server;
causing said first portion to be carried out via a secure private on-line communications connection between said terminal device and a private information server; and
automatically reconfiguring network access means in said terminal device to switch between said public connection and said private connection.

Electronic Acknowledgement Receipt

EFS ID:	15024602
Application Number:	13336790
International Application Number:	
Confirmation Number:	6217
Title of Invention:	SYSTEM AND METHOD EMPLOYING AN AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES
First Named Inventor/Applicant Name:	Victor Larson
Customer Number:	23630
Filer:	Toby H. Kusmer./Kerrie Jones
Filer Authorized By:	Toby H. Kusmer.
Attorney Docket Number:	77580-151(VRNK-1CP3CNFT1)
Receipt Date:	22-FEB-2013
Filing Date:	23-DEC-2011
Time Stamp:	12:19:36
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Non Patent Literature	D1265part1.pdf	3257426 e10fa16cdf3dcc4c801e3c43b1076cfb845f7222	no	100

Warnings:

Information:

2	Non Patent Literature	D1265part2.pdf	4401034 0f6c7db8be993084987583de124a46502cf133b7	no	100
Warnings:					
Information:					
3	Non Patent Literature	D1265part3.pdf	3009164 a79e672b0a35ab1c4061dec70c92ee842790db53	no	100
Warnings:					
Information:					
4	Non Patent Literature	D1265part4.pdf	4214786 01c33769697efb1897305b52fc45d7f8e62bb982	no	100
Warnings:					
Information:					
5	Non Patent Literature	D1265part5.pdf	4113443 a3af25096f2ee9d952c4d9458c6f187d5f1dcff4	no	100
Warnings:					
Information:					
6	Non Patent Literature	D1265part6.pdf	3897956 640a8bf16b4e3192285eb10013a9bda1c349843e	no	100
Warnings:					
Information:					
7	Non Patent Literature	D1265part7.pdf	2164810 343b367b0c48be1264b3e566a452512819d19bb4	no	100
Warnings:					
Information:					
8	Non Patent Literature	D1265part8.pdf	3233153 d368aff43c60b3c2584edf77d5d5e500e020ba56	no	100
Warnings:					
Information:					
9	Non Patent Literature	D1265part9.pdf	2825989 3a267e18a93361908301ce620a7846feaf186d94	no	100
Warnings:					
Information:					
10	Non Patent Literature	D1265part10.pdf	2972212 6f17595814d2d60bf176852f8752c7a359e8cd29	no	100
Warnings:					
Information:					

11	Non Patent Literature	D1265part11.pdf	2160112 4fe7498a9594c1d8647be0d40219460cce69fc81	no	100
Warnings:					
Information:					
12	Non Patent Literature	D1265part12.pdf	3309637 cba2cab7755944754f6f66aedd00edcb01f9be94	no	100
Warnings:					
Information:					
13	Non Patent Literature	D1265part13.pdf	3572344 1838143d07c78fc93a7944f8fbb540f627be2438	no	100
Warnings:					
Information:					
14	Non Patent Literature	D1265part14.pdf	3352910 a9b6228a0fae1c13c1868502f029428c664bf9c0	no	100
Warnings:					
Information:					
15	Non Patent Literature	D1265part15.pdf	3496255 240ee7a5796f70e1f8d6a2691f4a440bd12b3562	no	100
Warnings:					
Information:					
			Total Files Size (in bytes):	49981231	

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

23630 7590 01/10/2013

McDermott Will & Emery
The McDermott Building
500 North Capitol Street, N.W.
Washington, DC 20001

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)

(Signature)

(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
13/336,790	12/23/2011	Victor Larson	77580-151(VRNK-1CP3CNFT1)	6217

TITLE OF INVENTION: SYSTEM AND METHOD EMPLOYING AN AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1770	$0	$0	$1770	04/10/2013

EXAMINER	ART UNIT	CLASS-SUBCLASS
LIM, KRISNA	2453	709-204000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).
☐ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
☐ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list
(1) the names of up to 3 registered patent attorneys or agents OR, alternatively,
(2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 McDermott Will & Emery LLP
2
3

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: VirnetX, Inc.

(B) RESIDENCE: (CITY and STATE OR COUNTRY): Zephyr Cove, Nevada

Please check the appropriate assignee category or categories (will not be printed on the patent) : ☐ Individual ☒ Corporation or other private group entity ☐ Government

4a. The following fee(s) are submitted:
☒ Issue Fee
☐ Publication Fee (No small entity discount permitted)
☐ Advance Order - # of Copies ______

4b. Payment of Fee(s): **(Please first reapply any previously paid issue fee shown above)**
☐ A check is enclosed.
☐ Payment by credit card. Form PTO-2038 is attached.
☒ The Director is hereby authorized to charge the required fee(s), any deficiency, or credit any overpayment, to Deposit Account Number 501133 (enclose an extra copy of this form).

5. **Change in Entity Status** (from status indicated above)
☐ a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.
☐ b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature [illegible] Date 2/25/13

Typed or printed name Toby H. Kusmer Registration No. 26,418

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

Electronic Patent Application Fee Transmittal

Application Number:	13336790
Filing Date:	23-Dec-2011
Title of Invention:	SYSTEM AND METHOD EMPLOYING AN AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES
First Named Inventor/Applicant Name:	Victor Larson
Filer:	Toby H. Kusmer./Tricia Tedesco
Attorney Docket Number:	77580-151(VRNK-1CP3CNFT1)

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Utility Appl issue fee	1501	1	1770	1770
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
		Total in USD ($)		**1770**

Electronic Acknowledgement Receipt

EFS ID:	15039288
Application Number:	13336790
International Application Number:	
Confirmation Number:	6217
Title of Invention:	SYSTEM AND METHOD EMPLOYING AN AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES
First Named Inventor/Applicant Name:	Victor Larson
Customer Number:	23630
Filer:	Toby H. Kusmer./Tricia Tedesco
Filer Authorized By:	Toby H. Kusmer.
Attorney Docket Number:	77580-151(VRNK-1CP3CNFT1)
Receipt Date:	25-FEB-2013
Filing Date:	23-DEC-2011
Time Stamp:	14:39:01
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Deposit Account
Payment was successfully received in RAM	$1770
RAM confirmation Number	14882
Deposit Account	501133
Authorized User	

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

Charge any Additional Fees required under 37 C.F.R. Section 1.16 (National application filing, search, and examination fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.17 (Patent application and reexamination processing fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.19 (Document supply fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.20 (Post Issuance fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.21 (Miscellaneous fees and charges)

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Issue Fee Payment (PTO-85B)	IssueFeePayment.pdf	132002 47150c219b53e4f80802823daa1e934830b9a628	no	1

Warnings:

The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing

Information:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
2	Fee Worksheet (SB06)	fee-info.pdf	30560 8d4c93fd0455155c90d4ccd304a215a3b4988d23	no	2

Warnings:

Information:

Total Files Size (in bytes):	162562

Docket No.: 077580-0151
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of: Victor Larson et al.	Customer Number: 23630
Application No.: 13/336,790	Confirmation No.: 6217
Filed: December 23, 2011	Art Unit: 2453
For: SYSTEM AND METHOD EMPLOYING AN AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES	Examiner: Krisna Lim

INFORMATION DISCLOSURE STATEMENT (IDS)

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Madam:

An Information Disclosure Statement was filed on February 19, 2013 and each item of information contained in the information disclosure statement became known to applicants not more than three months prior to the filing of the Information Disclosure Statement, however, the fee required under 37 CFR 1.17(P) was inadvertently omitted from the submission

The Commissioner is hereby authorized to charge the fee pursuant to 37 CFR 1.17(P) in the amount of $180.00, or further fees which may be due, to Deposit Account 50-1133.

Respectfully submitted,

MCDERMOTT WILL & EMERY LLP

Toby H. Kusmer, P.C.
Registration No. 26,418

28 State Street
Boston, MA 02109
Phone: (617) 535-4065
Facsimile: (617) 535-3800
Date: March 4, 2013

Please recognize our Customer No. 26630 as our correspondence address.

DM_US 40544584-1.077580.0161

Electronic Patent Application Fee Transmittal

Application Number:	13336790
Filing Date:	23-Dec-2011
Title of Invention:	SYSTEM AND METHOD EMPLOYING AN AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES
First Named Inventor/Applicant Name:	Victor Larson
Filer:	Toby H. Kusmer./Kerrie Jones
Attorney Docket Number:	77580-151(VRNK-1CP3CNFT1)

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
Submission- Information Disclosure Stmt	1806	1	180	180
		Total in USD ($)		**180**

Electronic Acknowledgement Receipt

EFS ID:	15106741
Application Number:	13336790
International Application Number:	
Confirmation Number:	6217
Title of Invention:	SYSTEM AND METHOD EMPLOYING AN AGILE NETWORK PROTOCOL FOR SECURE COMMUNICATIONS USING SECURE DOMAIN NAMES
First Named Inventor/Applicant Name:	Victor Larson
Customer Number:	23630
Filer:	Toby H. Kusmer./Kerrie Jones
Filer Authorized By:	Toby H. Kusmer.
Attorney Docket Number:	77580-151(VRNK-1CP3CNFT1)
Receipt Date:	04-MAR-2013
Filing Date:	23-DEC-2011
Time Stamp:	14:44:31
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Deposit Account
Payment was successfully received in RAM	$180
RAM confirmation Number	1349
Deposit Account	501133
Authorized User	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)

1	Information Disclosure Statement (IDS) Form (SB08)	Fee.pdf	34893 338b7ed0d1b18d34d847dcee5ef1b54387684496	no	1
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing					
2	Fee Worksheet (SB06)	fee-info.pdf	30612 569474a443b1cb0d690c64ed6607b828d791c13f	no	2
Warnings:					
Information:					
			Total Files Size (in bytes):	65505	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Subst. for form 1449/PTO	Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT *(Use as many sheets as necessary)*	Application Number	**13/336,790**
	Filing Date	**12-23-2011**
	First Named Inventor	**Victor Larson**
	Art Unit	**2453**
	Examiner Name	**Krisna Lim**
	Docket Number	**77580-151(VRNK-0001CP3CNFT1)**

U.S. PATENTS

EXAMINER'S INITIALS	CITE NO.	Patent Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
	A166	5,007,051	04/09/1991	Dolkas et al.	

U.S. PATENT APPLICATION PUBLICATIONS

EXAMINER'S INITIALS	CITE NO.	Patent Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear

FOREIGN PATENT DOCUMENTS

EXAMINER'S INITIALS	CITE NO.	Foreign Patent Document Country Codes - Number 4 - Kind Codes *(if known)*	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines Where Relevant Figures Appear	Translation	
						Yes	No
	C25	JP 09-270803	10/14/1997	Furukawa Electric Co. Ltd.		English Abstract	
	C26	JP 10-111848	04/28/1998	AT&T Corp.		English Abstract	
	C27	JP 10-215244	08/11/1998	Sony Corp.		English Abstract	
	C28	JP 04-117826	04/17/1992	Matsushita Electric Ind. Co. Ltd.		English Abstract	

OTHER ART (Including Author, Title, Date, Pertinent Pages, Etc.)

EXAMINER'S INITIALS	CITE NO.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	
	D1254	Eastlake, "Domain Name System Security Extensions," Network Working Group, RFC: 2535 pages 2-11 (March 1999)	
	D1255	Press Release; VirnetX and Aastra Sign a Patent License Agreement, 4 pages, May 2012, Printed from Website: http://virnetx.com/virnetx-and-aastra-sign-a-patent-license-agreement/	
	D1256	Press Release; VirnetX and Mitel Networks Corporation Sign a Patent License Agreement, 5 pages, July 2012, Printed from Website: http://virnetx.com/virnetx-and-mitel-networks-corporation-sign-a-patent-license-agreement/	
	D1257	Press Relese; Virnetx and NEC Corporation and NEC Corporation of America Sign a Patent License Agreement, 5 pages, August 2012, Printed from Website: http://virnetx.com/virnetx-and-nec-corporation-and-nec-corporation-of-america-sign-a-patent-license-agreement/	
	D1258	Supplemental Declaration of Angelos D. Keromytis, Ph.D from Control No.: 95001789 pp. 1-18, dated December 20, 2012	
	D1259	Supplemental Declaration of Angelos D. Keromytis, Ph.D from Control No.: 95001851 pp. 1-13, dated December 30, 2012	

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /K.L./

Subst. for form 1449/PTO **INFORMATION DISCLOSURE STATEMENT BY APPLICANT** *(Use as many sheets as necessary)*	**Complete if Known**
Application Number	**13/336,790**
Filing Date	**12-23-2011**
First Named Inventor	**Victor Larson**
Art Unit	**2453**
Examiner Name	**Krisna Lim**
Docket Number	**77580-151(VRNK-0001CP3CNFT1)**

	D1260	Supplemental Declaration of Angelos D. Keromytis, Ph.D from Control No.: 95001788 pp. 1-18, dated December 18, 2012	
	D1261	Supplemental Declaration of Angelos D. Keromytis, Ph.D from Control No.: 95001856 pp. 1-13, dated December 30, 2012	
	D1262	VirnetX vs Apple Transcript of Trial, Afternoon Session, 12:05 p.m., dated November 5, 2012	
	D1263	Certified Copy dated September 18, 2012 of U.S. Patent Number 6,502,135, 73 pages	
	D1264	Certified Copy dated December 30, 2009 of Assignment for Patent Application Number 95/047,83 12 pages	
	D1265	Certified Copy dated March 11, 2008 of Patent Application Number 09/504,783, 1500 pages	
	D1266	Certified Copy dated March 30, 2011 of U.S. Patent Number 7,418,504, 74 pages	
	D1267	Certified Copy dated October 17, 2012 of Assignment for Patent Application Number: 10/714,849, 10 pages	
	D1268	Certified Copy dated April 4, 2011 of Patent Application Number 10/714,849, 1170 pages	
	D1269	Certified Copy dated March 30, 2011 of U.S. Patent Number 7,490,151, 63 pages	
	D1270	Certified Copy dated October 17, 2012 of Assignment for Patent Application Number 10/259,494, 19 pages	
	D1271	Certified Copy dated April 4, 2011 of Application Number 10/259,454, 1359 pages	
	D1272	Certified Copy dated April 12, 2011 of U.S. Patent Number 7,921,211, 78 pages	
	D1273	Certified Copy dated October 17, 2012 of Assignment for Application Number 11/840,560, 12 pages	
	D1274	Certified Copy dated April 20, 2011 of Application Number 11/840,560, 3 pages	
	D1275	iPhone User Guide for iPhone OS 3.1 Software, 217 pages, 2009	
	D1276	iPhone User Guide for iOS 4.2 and 4.3 Software, 274 pages, 2011	
	D1277	iPhone User Guide for iPhone and iPhone 3G, 154 pages, 2008	
	D1278	iPhone User Guide for iOS 5.0 Software, 163 pages, 2011	
	D1279	iPad User Guide for iOS 5.0 Software, 141 pages, 2011	
	D1280	iPad User Guide for iOS 4.2 Software, 181 pages, 2010	
	D1281	iPad User Guide for iOS 4.3 Software, 198 pages, 2011	
	D1282	iPad User Guide, 154 pages, 2010	
	D1283	iPod Touch User Guide for iOS 5.0 Software, 143 pages, 2011	
	D1284	iPod Touch User Guide, 122 pages, 2008	
	D1285	iPod Touch User Guide for iPhone OS 3.0 Software, 153 pages, 2009	
	D1286	iPod Touch User Guide for iPhone OS 3.1 Software, 169 pages, 2009	
	D1287	iPod Touch User Guide for iOS 4.3 Software, 230 pages, 2011	
	D1288	iPod Touch Features Guide, 98 pages, 2008	

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /K.L./

Subst. for form 1449/PTO **INFORMATION DISCLOSURE STATEMENT BY APPLICANT** ***(Use as many sheets as necessary)***	**Complete if Known**
Application Number	**13/336,790**
Filing Date	**12-23-2011**
First Named Inventor	**Victor Larson**
Art Unit	**2453**
Examiner Name	**Krisna Lim**
Docket Number	**77580-151(VRNK-0001CP3CNFT1)**

	D1289	VPN Server Configuration for iOS; Networking & Internet Enterprise Deployment, 12 pages, 2011	
	D1290	iPhone Configuration Utility User Guide, 26 pages, 2010	
	D1291	iPhone Configuration Utility; Networking & Internet: Enterprise Deployment, 26 pages, 2011	
	D1292	iPhone Configuration Utility; Networking>Internet & Web, 24 pages, 2010	
	D1293	iOS Configuration Profile Reference; Networking & Internet: Enterprise Deployment, 24 pages, 2011	
	D1294	iPhone OS Enterprise Deployment Guide; Second Edition, for Version 3.1 or Later, 91 pages, 2009	
	D1295	iPhone OS; Enterprise Deployment Guide; Second Edition, for Version 3.2 or Later, 90 pages, 2010	
	D1296	CFHost Reference; Developer, 20 pages, 2008	
	D1297	CFNetwork Programming Guide; Developer, 60 pages, 2011	
	D1298	CFStream Socket Additions; Developer, 22 pages, 2010	
	D1299	Mac OS X Devloper Library; CFHostSample.c, 1 page	
	D1300	Mac OS X Developer Library; CFHostSample, 1 page, 2004	
	D1301	Mac OS X Developer Library; Document Revision History, 1 page, 2004	
	D1302	CFStream Socket Additions; Developer, 22 pages, 2010	
	D1303	Apple Push Notification Service; Distribution Service, Version 1.0, 6 pages, 2009	
	D1304	iOS Human Interface Guidelines; Developer, 184 pages, 2012	
	D1305	Networking & Internet Starting Point, 3 pages, 2011	
	D1306	Server Admin. 10.5 Help; Viewing a VPN Overview, 1 page	
	D1307	iOS: Supported Protocols for VPN, 2 pages, 2010	
	D1308	IPhone in Business Virtual Private Networks (VPN), 3 pages, 2010	
	D1309	iPhone and iPad in Business Deployment Scenarios, 26 pages, 2011	
	D1310	Deploying iPhone and iPad Virtual Private Networks, 3 pages, 2011	
	D1311	Deploying iPhone and iPad; Security Overview, 6 pages, 2011	
	D1312	Pad in Business; "Ready for Work," 2012, 5 pages	
	D1313	iOS: Using FaceTime, 2 pages, 2011, Printed from website http://support.apple.com/kb/HT4317	
	D1314	MobileMe: "Secure Chat" is Unavailable in OS X Lion, 2 pages, 2012, Printed from Website: http://support.apple.com/kb/TS3902	
	D1315	iPhone 4 and iPod Touch (4th Generation): Using FaceTime, 2 pages, 2010, Printed from Website: http://support.apple.com/kb/HT4319	
	D1316	IPhone; "Picking Up Where Amazing Left Off," 11 pages, 2012, Printed from Website: http://www.apple.com/iPhone/features/facetime	
	D1317	FaceTime for Mac; "Say Hello to FaceTime for Mac," 4 pages, 2012, Printed from Website: http://www.apple.com/mac/facetime	

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /K.L./

Subst. for form 1449/PTO	Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT *(Use as many sheets as necessary)*	Application Number	13/336,790
	Filing Date	12-23-2011
	First Named Inventor	Victor Larson
	Art Unit	2453
	Examiner Name	Krisna Lim
	Docket Number	77580-151(VRNK-0001CP3CNFT1)

	D1318	iPad; "Your New Favorite Way to do Just About Everything," 8 pages, 2012, Printed from Website" http://www.apple.com/ipad/built-in-apps/	
	D1319	iPod Touch; FaceTime, "Oh, I see what you're saying," 2 pages	
	D1320	Apple Press Info; Apple Presents iPhone 4, Printed from Website: http://www.apple.com/pr/library/apple-presents-iphone	
	D1321	iPod Touch; FaceTime, "Oh I See What You're Saying,", 3 pages, 2012, Printed from Website: http://www.apple.com/iPodtouch/built-in-apps/facetime.htm	
	D1322	IOS 4, The World's Most Advanced Mobile Operating System, 5 pages, Printed from Website: http://www.apple.com/iphone/ios4	
	D1323	Apple Press Info; Apple Reinvents the Phone with iPhone, 3 pages, 2007, Printed from Website: http://www.apple.com/pr/library/2007/01/09Apple-reinvents-the-phone	
	D1324	Apple Press Info; Apple Announces the New iPhone 3Gs-The Fastest, Most Powerful iPhone Yet, 3 pages, 2009, Printed from the Website: http://www.apple.com/pr/library/2009/06/08Apple-Announces-the-new-iphone3GS	
	D1325	Apple Press Info; Apple Launches iPhone 4S, ios 5 & iCloud, iPhone 4S Features Dual-Core A5 Chip, All New Camera, full 1080p HD Video Recording & Introduces Siri, 2011, 2 pages, Printed from website: http://www.apple.com/pr/library/2011/10/04Apple-Launches-iPhone-4S-iOS-5-iCloud.html	
	D1326	Apple Press Info; Apple Introduces New iPod Touch, Features Retina Display, A4 Chip, FaceTime Video Calling, HD Video Recording & Game Center, 2 pages, 2010, Printed from Website http://www.apple.com/pr/library/2010/09/01Apple-Introduces-New-iPod-touch.html	
	D1327	Apple Press Info; Apple Launches iPad, Magical & Revolutionary Device at an Unbelievable Price, 2 pages, 2010, Printed from Website: http://www.apple.com/pr/library/2010/01/27Apple-Launches-iPad.html	
	D1328	Apple Press Info; Apple Launces New iPad, New iPad Features Retina Display, A5X Chip, 5 Megapixel iSight Camera & Ultrafast 4G LTE, 2012, 3 pages, Printed from the Website: http://www.apple.com/pr/library/2012/03/07Apple-Launches-New-iPad.html	
	D1329	FaceTime; "Phone Calls Like You've Never Seen Before," 3 pages	
	D1330	Apple Press Info; Apple Brings FaceTime to the Mac, 1 pages, Printed from Website https://www.apple.com/pr/library/2010/10/20Apple-Brings-FaceTime-to-the-Mac.html	
	D1331	iPad at Work; "Mobile Meetings Made Easy," 4 pages, 2011	
	D1332	IPad – Technical Specifications, 49 pages, Printed from Website: http://support.apple.com/kb/sp58C	
	D1333	Stirling Design, 8 pages, 2008	
	D1334	Quick Guide: SSL VPN Technical Primer, 11 pages, 2010	
	D1335	Silva, "Secure iPhone Access to Corporate Web Applications," Technical Brief, 10 pages	
	D1336	Defendant Apple Inc.'s Third Supplemental Responses to VirnetX Inc.'s First Request for Admission to Apple Inc. dated, April 13, 2012, 207 pages	
	D1337	Apple Support Communities, 4 pages, Printed from Website https://discussions.apple.com/thread/486096?start=0&tstart=0	
	D1338	VirnetX – Products; License and Service Offerings, 1 page	

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /K.L./

Subst. for form 1449/PTO **INFORMATION DISCLOSURE STATEMENT BY APPLICANT** *(Use as many sheets as necessary)*	**Complete if Known**	
	Application Number	**13/336,790**
	Filing Date	**12-23-2011**
	First Named Inventor	**Victor Larson**
	Art Unit	**2453**
	Examiner Name	**Krisna Lim**
	Docket Number	**77580-151(VRNK-0001CP3CNFT1)**

	D1339	VirnetX Contact Information, 4 pages, 2011	
	D1340	VirnetX Launches Secure Domain Name Initiative; 4G/LTE Security, 1 page, 2010	
	D1341	VirnetX Gabriel Connection; Enabling Safe Network Neighborhoods, 2 pages, 2012	
	D1342	Baugher et al., "The Secure Real-Time Transport Protocol (SRTP)," Network Working Group, RFC:3711, 39 pages, 2004	
	D1343	Jennings et al., "Resource Location and Discovery (Reload) Draft-Bryan-P2PSIP-Reload-04," Internet-Draft, 12/12/08, pages 1-127	
	D1344	Barnes et al., "Verification Involving PSTN Reachability: Requirements and Architecture Overview," Internet Draft, 27 pages, 2012	
	D1345	April Inc. Form 10-K (Annual Report) filed 12/01/05 for the Period Ending 09/24/05, Edgar Online, 1400 pages, 2011	
	D1346	Phone, Facetime; "Be in Two Places at Once," 3 pages, Printed from the Website http://www.apple.com/ios/facetime/	
	D1347	Apple Press Info; Apple Presents iPhone 4, All-New Design with FaceTime Video Calling, Retina, Display, 5 Megapixel Camera & HD Video Recording, 3 pages, 2010	
	D1348	NYSE AMEX:VHC; Cowen and Co. 39th Annual Technology Media & Telecom Conference, 36 pages, June 2, 2011	
	D1349	Pindyck et al., "Market Power: Monopoly and Monopsony," Microeconomics, Sixth Edition, pages 370-371	
	D1350	Press Release; IpCapital Group Completes VirnetX IP Licensing Evaluation, 3 pages	
	D1351	Microsoft Real-Time Communications: Protocols and Technologies, Microsoft TechNet, 22 pages, 2010	
	D1352	Filing Receipt dated September 23, 2011 for Application Number: 13/223,259	
	D1353	Email Communications Regarding Apple Product Innovations, 6 pages, 2010	
	D1354	Mathy et al., "Traversal Using Relays Around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)," Internet Engineering Task Force (IETF), RFC: 5766, 67 pages, 2010	
	D1355	Egevang et al., "The IP Network Address Translator (NAT)," Network Working Group, RFC: 1631, 10 pages, 1994	
	D1356	Srisuresh et al., "IP Network Address Translator (NAT) Terminology and Considerations," Network Working Group, RFC:2663, 30 pages, 1999	
	D1357	Sisalem, et al., "Introduction to Cryptographic Mechanisms," SIP Security, 356 pages, 2009	
	D1358	Curriculum Vitae, Mark T Jones, 9 pages	
	D1359	Curriculum Vitae, Roy Weinstein, 5 pages	
	D1360	How To Configure IPSec Tunneling in Windows 2000, 8 pages	
	D1361	Press Relese; Virnetx and NEC Corporation and NEC Corporation of America Sign a Patent License Agreement, 5 pages, August 2012, Printed from Website: http://virnetx.com/virnetx-and-nec-corporation-and-nec-corporation-of-america-sign-a-patent-license-agreement/	

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /K.L./

Subst. for form 1449/PTO	Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT *(Use as many sheets as necessary)*	Application Number	**13/336,790**
	Filing Date	**12-23-2011**
	First Named Inventor	**Victor Larson**
	Art Unit	**2453**
	Examiner Name	**Krisna Lim**
	Docket Number	**77580-151(VRNK-0001CP3CNFT1)**

	D1362	iPhone, FaceTime; "Be in Two Places at Once," 3 pages, Printed from Website: http://www.apple.com/ios/facetime/	
	D1363	iPhone, "It Does Everything Better,"6 pages, Printed from Website: http://www.apple.com/iPhone/built-in-apps	
	D1364	My Apple ID, "What's an Apple ID," 1 pages, Printed from Website: https://appleid.apple.com/cgi-bin/webobjects/myappleid.woa	
	D1365	Rosenberg et al., "Session Initiation Protocol (SIP): Locating SIP Servers," Network Working Group, RFC: 3263, 17 pages, 2002	
	D1366	Certified Copy dated September 21, 2012 of Reexamination Certificate Number 6,502,135 issued June 6, 2011, 11 pages	
	D1367	Certified Copy dated September 20, 2012 of Patent Application Number 95/001,269, 4999 pages	
	D1368	Chatterjee et al., "Bargaining Under Incomplete Information," Operations Research, 31:835-851, 1983	
	D1369	Nash, "The Bargaining Problem," Econometrica, 18:155-162, 1950	
	D1370	Nash, "Two-Person Cooperative Games," Econometrica, 21:128-140, 1953	
	D1371	Choi et al., "An Analytical Solution to Reasonable Royalty Rate Calculations," IDEA: The Journal of Law and Technology, 13 pages, 2001	
	D1372	The Prize in Economics 1994 - Press Release dated October 11, 1994, 4 pages, Printed from Website: http://www.nobelprize.org/nobel_prizes/economics/laureates/1994/press.html	
	D1373	Putnam et al., "Bargaining and the Construction of Economically Consistent Hypothetical License Negotiations," The Licensing Journal, pages 8-15, 2004	
	D1374	Scherling et al., "Rational Reasonable Royalty Damages: A Return to the Roots," Landslide, Volume 4, 4 pages, 2011	
	D1375	Jarosz et al., "Application of Game Theory to Intellectual Property Royalty Negotiations," Chapter 17, pages 241-265	
	D1376	Goldscheider, Licensing Best Practices; Strategic, Territorial, and Technology Issues, 2 pages, 2006	
	D1377	iPhone Configuration Utility, 19 pages, 2012	
	D1378	VPN Server Configuration for iOS Devices, 6 pages, 2012	
	D1379	Samuelson et al., Economics, Fourteenth Edition, pages 258-259, 1992	
	D1380	Stigler et al., The Theory of Price, Forth Edition, pages 215-216, 1987	
	D1381	Truett et al., "Joint Profit Maximization, Negotiation, and the Determinacy of Price in Bilateral Monopoly," Journal of Economic Education, pages 260-270	
	D1382	Binmore et al., "Noncooperative Models of Bargaining," The Handbook of Game Theory, 1:(7)181-225,1992	
	D1383	Spindler et al., "Endogenous Bargaining Power in Bilateral Monopoly and Bilateral Exchange," Canadian Journal of Economics-Revue Canadienne D Economie, pages 464-474, 1974	
	D1384	Myerson, "Game Theory; Analysis or Conflict," Harvard University Press, pages 375-392	

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /K.L./

Subst. for form 1449/PTO **INFORMATION DISCLOSURE STATEMENT BY APPLICANT** ***(Use as many sheets as necessary)***	**Complete if Known**		
	Application Number	**13/336,790**	
	Filing Date	**12-23-2011**	
	First Named Inventor	**Victor Larson**	
	Art Unit	**2453**	
	Examiner Name	**Krisna Lim**	
	Docket Number	**77580-151(VRNK-0001CP3CNFT1)**	

	D1385	Binmore, "The Nash Bargaining Solution in Economic Modelling," The Rand Journal of Economics, 17:176-188, 1996	
	D1386	Rubinstein et al., "On the Interpretation of the Nash Bargaining Solution and its Extension to Non-Expected Utility Preferences," Econometrica, 60:1171-1186, 1992	
	D1387	Greenleaf et al., "Guarantees in Auctions: The Auction House as Negotiator and Managerial Decision Maker," Management Science, 39:1130-1145, 1993	
	D1388	Chan, "Trade Negotiations in a Nash Bargaining Model," Journal of International Economics, 25:253-363, 1987	
	D1389	Lemley et al., "Patent Holdup and Royalty Stacking," Texas Law Review, 85:1991-2049	
	D1390	Cauley, "Winning the Patent Damages Case; A Litigator's Guide to Economic Models and Other Damage Strategies," Oxford Press, pages 29-30, 2044	
	D1391	Degnan et al., "A Survey of Licensed Royalties," Les Nouvelles, pages 91, 93, 94, 1997	
	D1392	Kahn, "The Review of Economics and Statistics," pages 157-164, 1993	
	D1393	Microsoft Company Information; Including Stocks and Financial Information, 83 pages	
	D1394	Apple Press Info: Apple Updates MacBook Pro with Next Generation Processors, Graphics & Thunderbolt I/O Technology, 3 pages, Printed from Website: http://www.apple.com/pr/library/2011/02/24Apple-Updates-MacBook-Pro-with-Next-Generation-Processors-Graphics-Thunderbolt-I-O-Technology.html	
	D1395	Apple Press Info: Apple to Ship Mac OS X Snow Leopard on August 28, 2 pages, Printed from the Website: http://www.apple.com/pr/library/2009/08/24/apple-to-ship-mac-os-x	
	D1396	iPad, Facetime; "Once Again, iPad gets the World Talking," 3 pages, Printed from the Website: http://www.apple.com/ipad/built-in-apps/facetime/html	
	D1397	Apple iOS: Setting up VPN, 2 pages, Printed from Website: http/support.apple.com/kb/HT1424	
	D1398	Apple iPhone User Guide for iOS 5.1 Software, 179 pages, 2012	
	D1399	Apple, Communicating with HTTP Servers, CFNetworking Programming Guide, 6 pages, 2011, Printed from the Website: https://developer.apple.com/library/ios/documentation/networking/conceptual/CFNetwork/CFHT	
	D1400	VirnetX, Gabriel Connection Technology ^{TM} White Paper, 7 pages, 2012	

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /K.L./

Subst. for form 1449/PTO **INFORMATION DISCLOSURE STATEMENT BY APPLICANT** *(Use as many sheets as necessary)*	**Complete if Known**
Application Number	**13/336,790**
Filing Date	**12-23-2011**
First Named Inventor	**Victor Larson**
Art Unit	**2453**
Examiner Name	**Krisna Lim**
Docket Number	**77580-151(VRNK-0001CP3CNFT1)**

	D1401	VirnetX, Technology, 4 pages, 2012	
	D1402	Certified Copy dated January 15, 2008 of U.S. Patent Number 6,502,135, 64 pages	
	D1403	Inter Partes Reexamination Certificate dated June 7, 2011 for Patent Number 6,502,135	
	D1404	Proceedings of The Symposium on Network and Distributed System Security, 7 pages, February 22-23, 1996	
	D1405	In-Q-Tel; Corporate Overview, 2 pages, 2004	
	D1406	Davies, Supervisor of Translation: Tadahiro Uezono, Security for Computer Networks, Japan, Nikkei-McGraw-Hill Inc., First Edition, First Copy, p 126-129 (December 5, 1985) – (English Version and Japanese Version Submitted)	
	D1407	Comer, "Translated by Jun Murai and Hiroyuki Kusumoto, "Internetworking with TCP/IP Vol. 1: Principles, Protocols, and Architecture, Third Edition," Japan Kyoritsu Shuppan Co., Ltd., First Edition, First Copy, p 161-193 (August 10, 1997) (English Version and Japanese Version Submitted)	
	D1408	Lynch et al., Supervisor of Translation: Jun Murai, "Internet System Handbook," Japan Impress Co. Ltd. First Edition p 152-157 and p 345-351 (August 11, 1996) (English Version and Japanese Version Submitted)	
	D1409	Office Action dated December 27, 2012 from Corresponding Canadian Patent Application Number 2723504	
	D1410	Office Action dated December 5, 2012 from Corresponding Japanese Patent Application Number 2011-081417	
	D1411	Office Action dated December 13, 2012 from Corresponding Japanese Patent Application Number 2011-085052	
	D1412	Office Action dated December 13, 2012 from Corresponding Japanese Patent Application Number 2011-083415	

EXAMINER /Krisna Lim/	DATE CONSIDERED 04/28/2013

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.
1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /K.L./

Subst. for form 1449/PTO		Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT *(Use as many sheets as necessary)*		Application Number	**13/336,790**
		Filing Date	**12-23-2011**
		First Named Inventor	**Victor Larson**
		Art Unit	**2453**
		Examiner Name	**Krisna Lim**
		Docket Number	**77580-151(VRNK-0001CP3CNFT1)**

CERTIFICATION STATEMENT

This Information Disclosure Statement is being filed to replace the Information Disclosure Statement filed with the United States Patent and Trademark Office on February 15, 2013.

The following items were added:

1. Japanese Office Action dated December 13, 2012 from Corresponding Japanese Patent Application Number 2011-083415 has been added (DI412) to note where prior art reference C25 came from; and

2. The certification was added to note that the remaining references contained in the information disclosure statement were cited in a communication from a foreign patent office in a counterpart foreign application, and, to the was known to any individual designated in § 1.56(c) more than three months prior to the filing of the information disclosure statement.

Please See 37 CFR 1.97 and 1.98 to make the appropriate selection(s)

[ ] Information Disclosure Statement is being filed with the filing of the application or before the receipt of a first office action.

[X] That each item of information contained in the information disclosure statement was first cited in any communication from a foreign patent office in a counterpart foreign application not more than three months prior to the filing of the information disclosure statement; or; Cited reference A166 from Canadian office action dated December 27, 2012; Cited reference C25 from Japanese office action dated 12/13/12; Cited references C26, D1254 from Japanese office action dated 12/13/12; C27-C28, D1406-1408 from Japanese office action dated 12/05/12.

[X] That no item of information contained in the information disclosure statement was cited in a communication from a foreign patent office in a counterpart foreign application, and, to the knowledge of the person signing the certification after making reasonable inquiry, no item of information contained in the information disclosure statement was known to any individual designated in § 1.56(c) more than three months prior to the filing of the information disclosure statement. Cited references D1255-D1405 all received by the client on January 31, 2013.

[ ] The Commissioner is hereby authorized to charge any required fees to Deposit Account 50-1133.

[ ] Information Disclosure Statement is being filed with the Request for Continued Examination. The Commissioner is hereby authorized to charge the fee pursuant to 37 CFR 1.17(P) in the amount of $810.00, or further fees which may be due, to Deposit Account 50-1133.

SIGNATURE

A signature of the applicant or representative is required in accordance with CFR 1.33, 10.18. Please see CFR 1.4(d) for the form of the signature.

/Toby H. Kusmer/
Toby H. Kusmer; Reg. No.:26,418
McDermott Will & Emery LLP
28 State Street
Boston, MA 02109
Tel. (617) 535-4000
Fax (617) 535-3800

Date: February 19, 2013

DM_US 41241724-1.077580.0151

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /K.L./

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
13/336,790	12/23/2011	Victor Larson	77580-151(VRNK-1CP3CNFT1)	6217

23630 7590 05/03/2013
McDermott Will & Emery
The McDermott Building
500 North Capitol Street, N.W.
Washington, DC 20001

EXAMINER	
LIM, KRISNA	
ART UNIT	**PAPER NUMBER**
2453	
NOTIFICATION DATE	**DELIVERY MODE**
05/03/2013	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

mweipdocket@mwe.com

PTOL-90A (Rev. 04/07)

UNITED STATES DEPARTMENT OF COMMERCE
U.S. Patent and Trademark Office
Address : COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450

APPLICATION NO./ CONTROL NO.	FILING DATE	FIRST NAMED INVENTOR / PATENT IN REEXAMINATION	ATTORNEY DOCKET NO.
13/336,790	23 December, 2011	SHORT ET AL.	77580-151(VRNK-1CP3CNFT1)

		EXAMINER	
McDermott Will & Emery The McDermott Building 500 North Capitol Street, N.W. Washington, DC 20001		KRISNA LIM	
		ART UNIT	PAPER
		2453	20130429

DATE MAILED:

Please find below and/or attached an Office communication concerning this application or proceeding.

Commissioner for Patents

The 1449 of the IDS filed 02/19/2013 with Examiner's initial, sign and date is attached herewith.

/Krisna Lim/
Primary Examiner, Art Unit 2453

PTO-90C (Rev.04-03)

~~13336790 GAU:~~ 2453

Subst. for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	13/336,790
Filing Date	12-23-2011
First Named Inventor	Victor Larson
Art Unit	2165
Examiner Name	Krisna Lim
Docket Number	77580-151(VRNK-0001CP3CNFT1)

Change(s) applied to document /P.H./ 3/4/2013

	A108		6,353,614	03/05/2002	Borella et al.	
	A109		6,425,003	07/23/2002	Herzog et al.	
	A110		6,430,155	08/06/2002	Davie et al	
	A111		6,430,610	08/06/2002	Carter	
	A112		6,487,598	11/26/2002	Valencia	
	A113		6,496,867	12/17/2002	Beser et al.	
	A114		6,502,135	12/2002	Munger et al.	
	A115		6,505,232	01/07/2003	Mighdoll et al	
	A116		6,510,154	01/21/2003	Mayes et al	
	A117		6,549,516	04/15/2003	Albert et al	
	A118		6,557,037	04/2003	Provino, Joseph E.	
	A119		6,560,634	05/06/2003	Broadhurst	
	A120		6,571,296	05/27/2002	Dillon	
	A121		6,571,338	05/27/2003	Shaio et al.	
	A122		6,581,166	~~7/17/2003~~	Hirst et al.	June 17, 2003
	A123		6,606,708	08/12/2003	Devine et al.	
	A124		6,615,357	9/2/2003	Boden et al.	
	A125		6,618,761	09/09/2003	Munger et al.	
	A126		6,671,702	12/30/2003	Kruglikov et al	
	A127		6,687,551	2/3/2004	Steindl	
	A128		6,687,746	02/03/04	Shuster, et al.	
	A129		6,701,437	03/02/2004	Hoke et al.	
	A130		6,714,970	3/30/2004	Fiveash et al.	
	A131		6,717,949	4/6/2004	Boden et al.	
	A132		6,751,738	06/15/2004	Wesinger, Jr. et al..	
	A133		6,752,166	06/22/04	Lull, et al.	
	A134		6,757,740	06/29/04	Parekh, et al.	
	A135		6,760,766	7/6/2004	Sahlqvist	
	A136		6,813,777	11/2004	Weinberger et al.	
	A137		6,826,616	11/30/2004	Larson et al.	
	A138		6,839,759	1/4/2005	Larson et al.	
	A139		6,937,597	08/30/2005	Rosenberg et al.	
	A140		60/134,547	05/17/1999	Victory Sheymov	
	A141		60/151,563	08/31/1999	Bryan Whittles	
	A142		7,010,604	3/7/2006	Munger et al.	
	A143		7,039,713	05/2006	Van Gunter et al.	
	A144		7,072,964	07/04/2006	Whittle et al.	
	A145		7,133,930	11/7/2006	Munger et al.	
	A146		7,167,904	01/23/07	Devarajan, et al.	
	A147		7,188,175	03/06/07	McKeeth, James A.	
	A148		7,188,180	3/6/2007	Larson et al.	
	A149		7,197,563	3/27/2007	Sheymov et al.	
	A150		7,353,841	04/08/08	Kono, et al.	
	A151		7,418,504	08/2008	Larson et al.	
	A152		7,461,334	12/02/08	Lu, et al.	
	A153		7,490,151	02/2009	Munger et al.	
	A154		7,493,403	02/2009	Shull et al.	
	A155		7,584,500	09/2009	Dillon et al.	
	A156		7,764,231	07/27/2010	Karr et al.	
	A157		7,852,861	12/2010	Wu et al.	
	A158		7,921,211	04/2011	Larson et al.	
	A159		7,933,990	04/2011	Munger et al.	
	A160		8,051,181	11/2011	Larson et al.	

/Krisna Lim/ 07/20/2012

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /K.L./

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	ISSUE DATE	PATENT NO.	ATTORNEY DOCKET NO.	CONFIRMATION NO.
13/336,790	06/04/2013	8458341	77580-151(VRNK-1CP3CNFT1)	6217

23630 7590 05/15/2013

McDermott Will & Emery
The McDermott Building
500 North Capitol Street, N.W.
Washington, DC 20001

ISSUE NOTIFICATION

The projected patent number and issue date are specified above.

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment is 0 day(s). Any patent to issue from the above-identified application will include an indication of the adjustment on the front page.

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Application Assistance Unit (AAU) of the Office of Data Management (ODM) at (571)-272-4200.

APPLICANT(s) (Please see PAIR WEB site http://pair.uspto.gov for additional applicants):

Victor Larson, Fairfax, VA;
Robert Dunham Short III, Leesburg, VA;
Edmond Colby Munger, Crownsville, MD;
Michael Williamson, South Riding, VA;