

FIG. 12

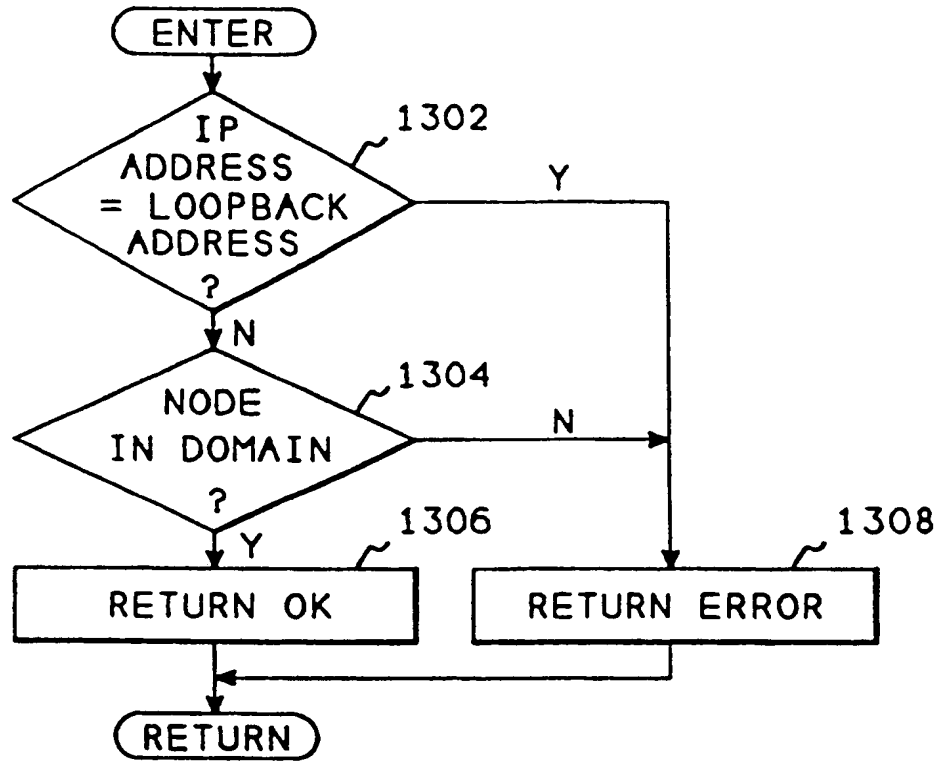


FIG. 13

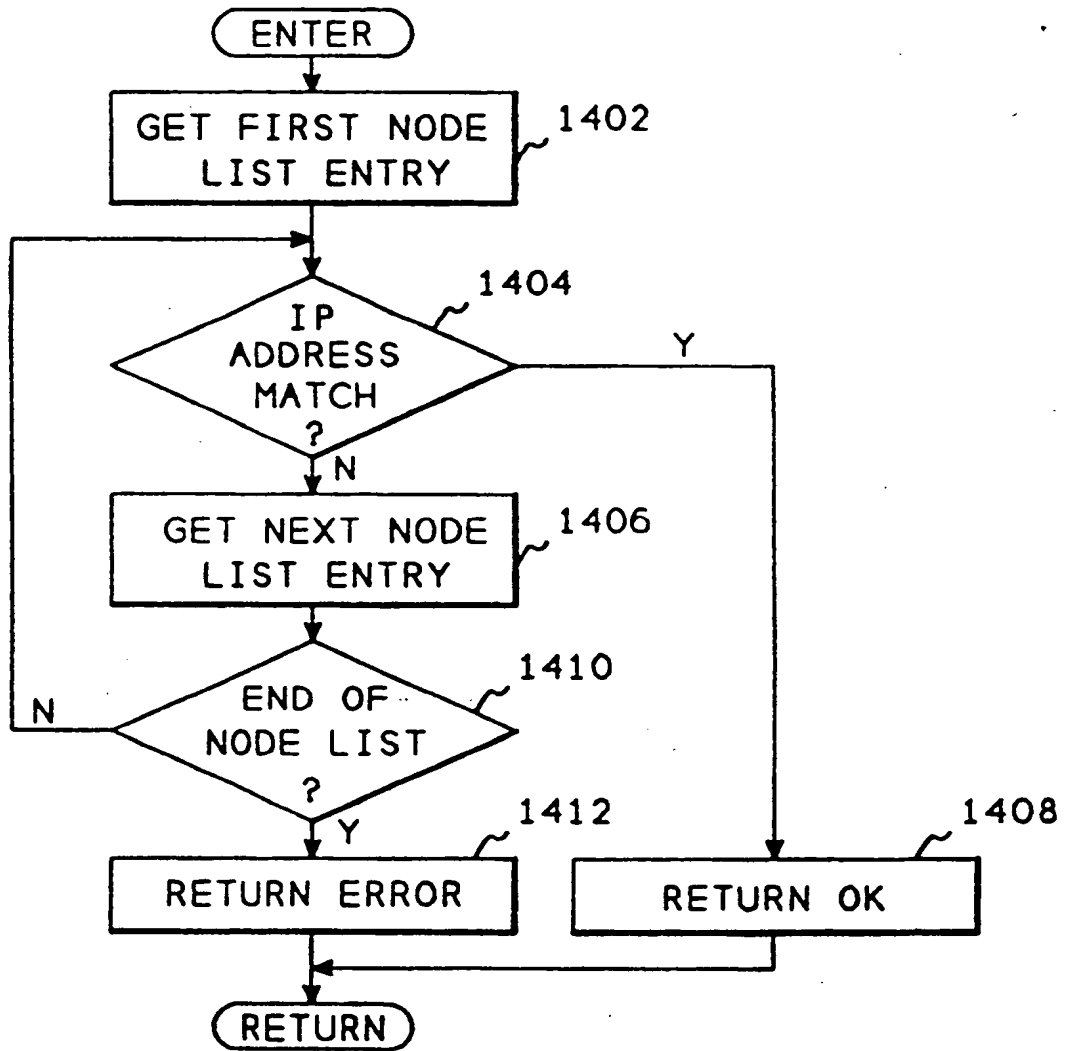


FIG. 14

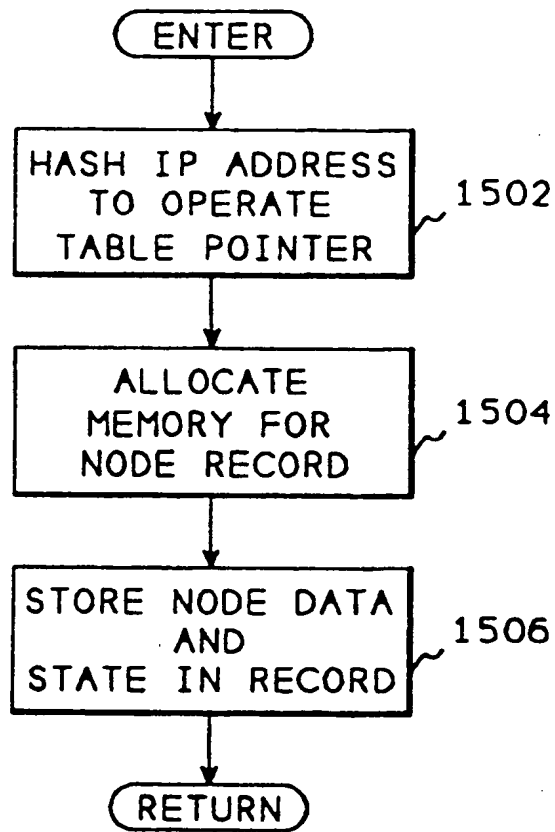


FIG. 15

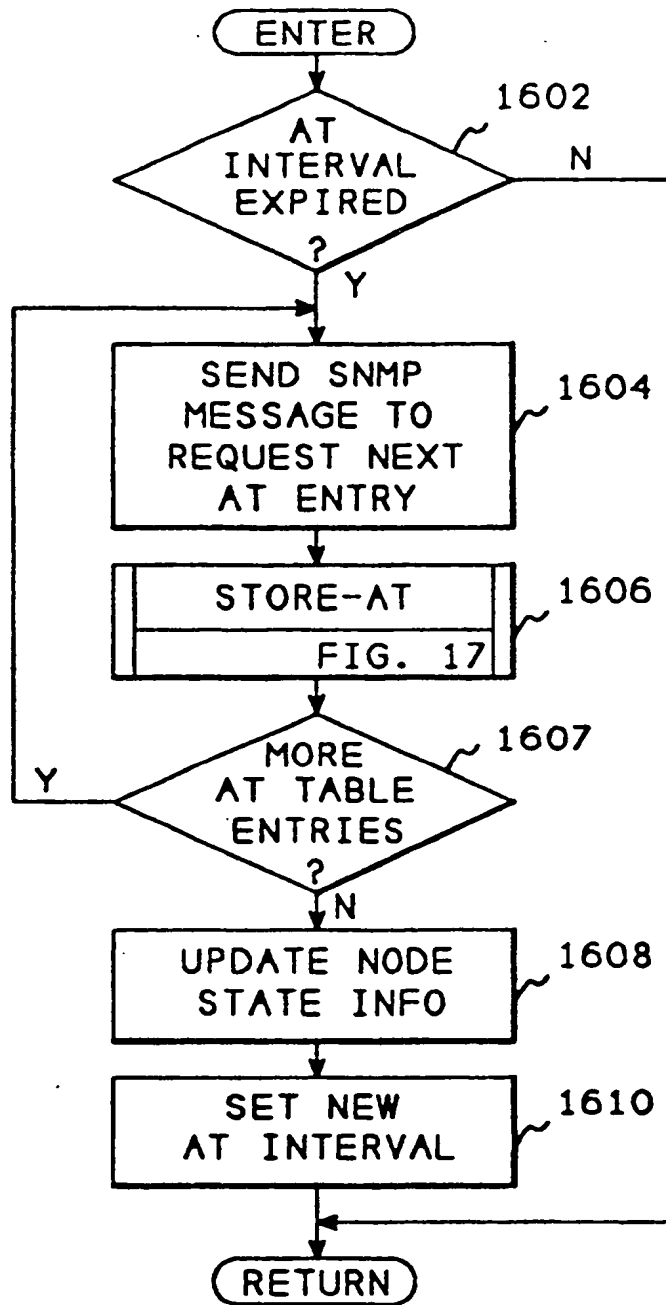


FIG. 16

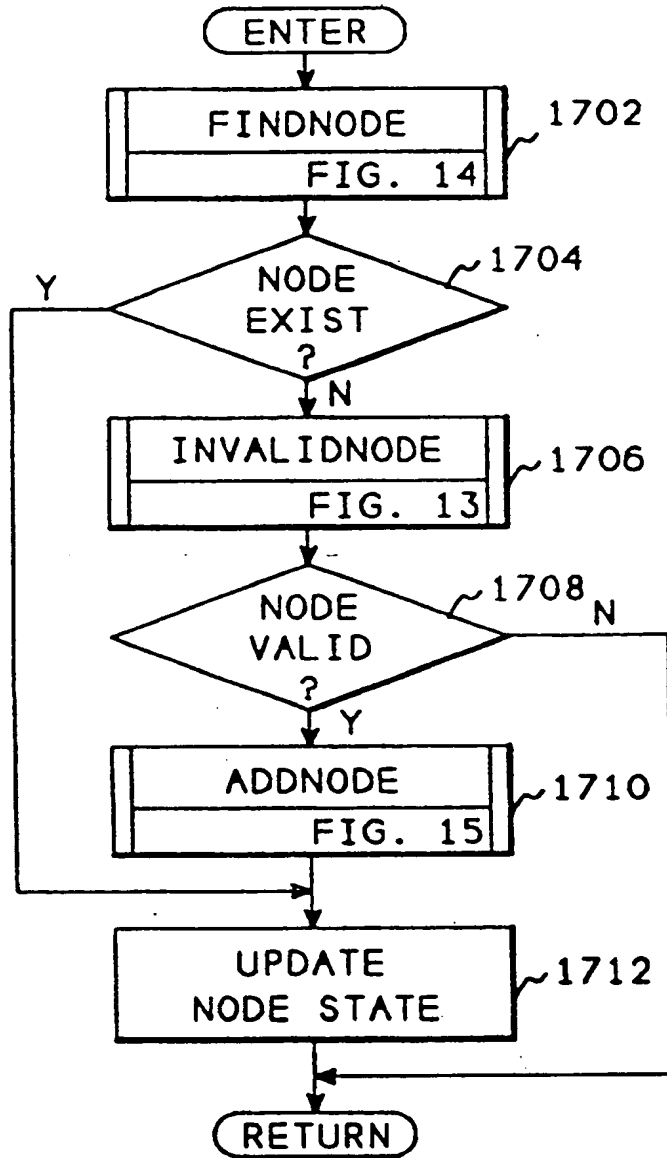


FIG. 17



EUROPEAN PATENT APPLICATION

Application number : **91303643.0**

Int. Cl.⁵ : **H04L 12/24**

Date of filing : **23.04.91**

Priority : **03.05.90 US 519187**

Inventor : **Wu, Jeff C.**
2630 Wapiti Road
Fort Collins, Colorado 80525 (US)

Date of publication of application :
06.11.91 Bulletin 91/45

Designated Contracting States :
DE FR GB

Representative : **Colgan, Stephen James et al**
CARMAELS & RANSFORD
43 Bloomsbury Square
London WC1A 2RA (GB)

Date of deferred publication of search report :
11.01.95 Bulletin 95/02

Applicant : **Hewlett-Packard Company**
Mail Stop 20 B-O,
3000 Hanover Street
Palo Alto, California 94304 (US)

Automatic discovery of network elements.

Disclosed is a computer network node discovery system (120) that provides a general way of discovering network elements, or nodes (100), connected to a computer network (118), and a specific algorithm for discovering nodes connected to a TCP/IP network, using the SNMP protocol available within the TCP/IP network software. Some nodes on a network, called discovery agents, can convey knowledge of the existence of other nodes on the network. The network discovery system queries these agents and obtains the information they have about other nodes on the network. It then queries each of the nodes obtained to determine if that node is also a discovery agent. In this manner, most of the nodes on a network can be discovered. The process of querying discovery agents to obtain a list of nodes known to the discovery agents is repeated at timed intervals to obtain information about nodes that are not always active. In a TCP/IP network, discovery agents are nodes that respond to queries for an address translation table which translates internet protocol (IP) addresses to physical addresses. The data from each node's address translation table is used to obtain both the IP and the physical address of other nodes on the network. These nodes are then queried to obtain additional information. After all the nodes on a network are discovered, the list of nodes is written to a database where it can be displayed by the network manager or other users of the network.

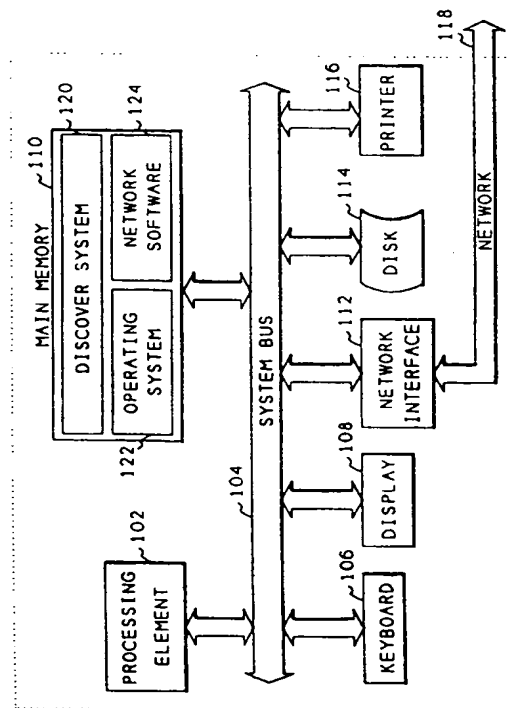


FIG. 1

EP 0 455 402 A3



European Patent
Office

EUROPEAN SEARCH REPORT

Application Number
EP 91 30 3643

DOCUMENTS CONSIDERED TO BE RELEVANT		
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim
Y	GB-A-2 217 488 (RACAL DATA COMMUNICATIONS INC.) * page 10, line 14 - page 14, line 2 * * page 15, line 18 - page 22, line 13 * * abstract * ---	1,7,9
Y	IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATION., vol.7, no.7, September 1989, NEW YORK US pages 1104 - 1114, XP54539 L.N.CASSEL ET AL 'NETWORK MANAGEMENT ARCHITECTURES AND PROTOCOLS: PROBLEMS AND APPROACHES' * paragraph V * -----	1,7,9
		CLASSIFICATION OF THE APPLICATION (Int.CI.5)
		H04L12/24
		TECHNICAL FIELDS SEARCHED (Int.CI.5)
		H04L
The present search report has been drawn up for all claims		
Place of search	Date of completion of the search	Examiner
THE HAGUE	31 October 1994	Canosa Areste, C
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

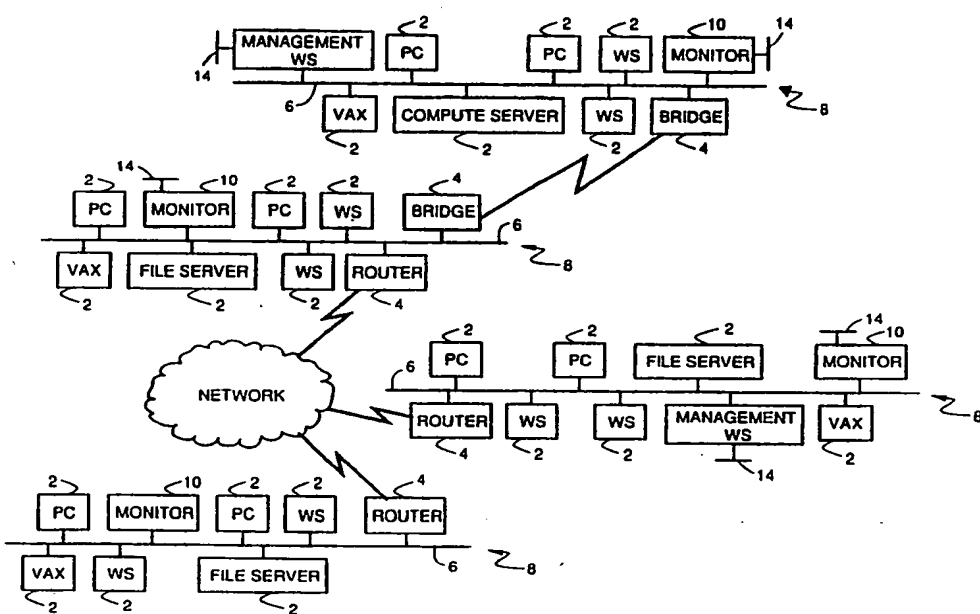
EPO FORM 1503 (03.82) (P04C01)



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

<p>(51) International Patent Classification ⁵ : H04J 3/14, 3/24, H04L 12/56</p>	<p>A1</p>	<p>(11) International Publication Number: WO 92/19054 (43) International Publication Date: 29 October 1992 (29.10.92)</p>
<p>(21) International Application Number: PCT/US92/02995 (22) International Filing Date: 10 April 1992 (10.04.92) (30) Priority data: 684,695 12 April 1991 (12.04.91) US (71) Applicant: CONCORD COMMUNICATIONS, INC. [US/US]; 753 Forest Street, Marlboro, MA 01752 (US). (72) Inventors: FERDINAND, Engel ; 21 Joseph Road, Northborough, MA 01532 (US). JONES, Kendall, S. ; 90 Boulder Road, Newton Center, MA 02159 (US). ROBERTSON, Kary ; 398 North Road, Bedford, MA 01739 (US). THOMPSON, David, M. ; 5127 243rd Road, Redmond, WA 98053 (US). WHITE, Gerard ; 133 Massapoag Road, Tyngsborough, MA 01879 (US).</p>	<p>(74) Agent: PRAHL, Eric, L.; Fish & Richardson, 225 Franklin Street, Boston, MA 02110-2804 (US). (81) Designated States: AT (European patent), BE (European patent), CA, CH (European patent), DE (European patent), DK (European patent), ES (European patent), FR (European patent), GB (European patent), GR (European patent), IT (European patent), JP, LU (European patent), MC (European patent), NL (European patent), SE (European patent). Published <i>With international search report.</i></p>	

(54) Title: NETWORK MONITORING



(57) Abstract

Monitoring is done of communications which occur in a network of nodes (2), each communication being effected by a transmission of one or more packets among two or more communicating nodes (2), each communication complying with a predefined communication protocol selected from among protocols available in the network. The contents of packets are detected passively and in real time, communication information (130, 152, 178) associated with multiple protocols is derived from the packet contents.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCI on the front pages of pamphlets publishing international applications under the PCI.

AT	Austria	ES	Spain	MG	Madagascar
AU	Australia	FI	Finland	ML	Mali
BB	Barbados	FR	France	MN	Mongolia
BE	Belgium	GA	Gabon	MR	Mauritania
BF	Burkina Faso	GB	United Kingdom	MW	Malawi
BG	Bulgaria	GN	Guinea	NL	Netherlands
BJ	Benin	GR	Greece	NO	Norway
BR	Brazil	HU	Hungary	PL	Poland
CA	Canada	IT	Italy	RO	Romania
CF	Central African Republic	JP	Japan	RU	Russian Federation
CG	Congo	KP	Democratic People's Republic of Korea	SD	Sudan
CH	Switzerland	KR	Republic of Korea	SE	Sweden
CI	Côte d'Ivoire	LI	Liechtenstein	SN	Senegal
CM	Cameroon	LK	Sri Lanka	SU	Soviet Union
CS	Czechoslovakia	LU	Luxembourg	TD	Chad
DE	Germany	MC	Monaco	TG	Togo
DK	Denmark			US	United States of America

- 1 -

NETWORK MONITORINGBackground of the Invention

The invention relates to monitoring and managing communication networks for computers.

5 Today's computer networks are large complex systems with many components from a large variety of vendors. These networks often span large geographic areas ranging from a campus-like setting to world wide networks. While the network itself can be used by many different types of
10 organizations, the purpose of these networks is to move information between computers. Typical applications are electronic mail, transaction processing, remote database, query, and simple file transfer. Usually, the organization that has installed and is running the
15 network needs the network to be running properly in order to operate its business. Since these networks are complex systems, there are various controls provided by the different equipment to control and manage the network. Network management is the task of planning,
20 engineering, securing and operating a network.

To manage the network properly, the Network Manager has some obvious needs. First, the Network Manager must trouble shoot problems. As the errors develop in a running network, the Network Manager must
25 have some tools that notify him of the errors and allow him to diagnose and repair these errors. Second, the Network Manager needs to configure the network in such a manner that the network loading characteristics provide the best service possible for the network users. To do
30 this the Network Manager must have tools that allow him visibility into access patterns, bottlenecks and general loading. With such data, the Network Manager can reconfigure the network components for better service.

There are many different components that need to
35 be managed in the network. These elements can be, but

- 2 -

are not limited to: routers, bridges, PC's, workstations, minicomputers, supercomputers, printers, file servers, switches and pbx's. Each component provides a protocol for reading and writing the management variables in the machine. These variables are usually defined by the component vendor and are usually referred to as a Management Information Base (MIB). There are some standard MIB's, such as the IETF (Internet Engineering Task Force) MIB I and MIB II standard definitions.

Through the reading and writing of MIB variables, software in other computers can manage or control the component. The software in the component that provides remote access to the MIB variables is usually called an agent. Thus, an individual charged with the responsibility of managing a large network often will use various tools to manipulate the MIB's of various agents on the network.

Unfortunately, the standards for accessing MIBs are not yet uniformly provided nor are the MIB definitions complete enough to manage an entire network. The Network Manager must therefore use several different types of computers to access the agents in the network. This poses a problem, since the errors occurring on the network will tend to show up in different computers and the Network Manager must therefore monitor several different screens to determine if the network is running properly. Even when the Network Manager is able to accomplish this task, the tools available are not sufficient for the Network Manager to function properly.

Furthermore, there are many errors and loadings on the network that are not reported by agents. Flow control problems, retransmissions, on-off segment loading, network capacities and utilizations are some of the types of data that are not provided by the agents.

- 3 -

Simple needs like charging each user for actual network usage are impossible.

Summary of the Invention

In general, in one aspect, the invention features
5 monitoring communications which occur in a network of nodes, each communication being effected by a transmission of one or more packets among two or more communicating nodes, each communication complying with a predefined communication protocol selected from among
10 protocols available in the network. The contents of packets are detected passively and in real time, communication information associated with multiple protocols is derived from the packet contents.

Preferred embodiments of the invention include the
15 following features. The communication information derived from the packet contents is associated with multiple layers of at least one of the protocols.

In general, in another aspect, the invention features monitoring communication dialogs which occur in
20 a network of nodes, each dialog being effected by a transmission of one or more packets among two or more communicating nodes, each dialog complying with a predefined communication protocol selected from among protocols available in the network. Information about
25 the states of dialogs occurring in the network and which comply with different selected protocols available in the network is derived from the packet contents.

Preferred embodiments of the invention include the following features. A current state is maintained for
30 each dialog, and the current state is updated in response to the detected contents of transmitted packets. For each dialog, a history of events is maintained based on information derived from the contents of packets, and the history of events is analyzed to derive information about
35 the dialog. The analysis of the history includes

- 4 -

counting events and gathering statistics about events. The history is monitored for dialogs which are inactive, and dialogs which have been inactive for a predetermined period of time are purged. For example, the current
5 state is updated to data state in response to observing the transmission of at least two data related packets from each node. Sequence numbers of data related packets stored in the history of events are analyzed and retransmissions are detected based on the sequence
10 numbers. The the current state is updated based on each new packet associated with the dialog; if an updated current state cannot be determined, information about prior packets associated with the dialog is consulted as an aid in updating the state. The history of events may
15 be searched to identify the initiator of a dialog.

The full set of packets associated with a dialog up to a point in time completely define a true state of the dialog at that point in time, and the step of updating the current state in response to the detected
20 contents of transmitted packets includes generating a current state (e.g., "unknown") which may not conform to the true state. The current state may be updated to the true state based on information about prior packets transmitted in the dialog.

25 Each communication may involve multiple dialogs corresponding to a specific protocol. Each protocol layer of the communication may be parsed and analyzed to isolate each dialog and statistics may be kept for each dialog. The protocols may include a connectionless-type
30 protocol in which the state of a dialog is implicit in transmitted packets, and the step of deriving information about the states of dialogs includes inferring the states of the dialogs from the packets. Keeping statistics for protocol layers may be temporarily suspended when parsing

- 5 -

and statistics gathering is not rapid enough to match the rate of packets to be parsed.

In general, in another aspect, the invention features monitoring the operation of the network with respect to specific items of performance during normal operation, generating a model of the network based on the monitoring, and setting acceptable threshold levels for the specific items of performance based on the model. In preferred embodiments, the operation of the network is monitored with respect to the specific items of performance during periods which may include abnormal operation.

In general, in another aspect, the invention features the combination of a monitor connected to the network medium for passively, and in real time, monitoring transmitted packets and storing information about dialogs associated with the packets, and a workstation for receiving the information about dialogs from the monitor and providing an interface to a user. In preferred embodiments, the workstation includes means for enabling a user to observe events of active dialogs.

In general, in another aspect, the invention features apparatus for monitoring packet communications in a network of nodes in which communications may be in accordance with multiple protocols. The apparatus includes a monitor connected to a communication medium of the network for passively, and in real time, monitoring transmitted packets of different protocols and storing information about communications associated with the packets, the communications being in accordance with different protocols, and a workstation for receiving the information about the communications from the monitor and providing an interface to a user. The monitor and the workstation include means for relaying the information about multiple protocols with respect to communication in

- 6 -

the different protocols from the monitor to the workstation in accordance with a single common network management protocol.

In general, in another aspect, the invention features diagnosing communication problems between two nodes in a network of nodes interconnected by links. The operation of the network is monitored with respect to specific items of performance during normal operation. A model of normal operation of the network is generated based on the monitoring. Acceptable threshold levels are set for the specific items of performance based on the model. The operation of the network is monitored with respect to the specific items of performance during periods which may include abnormal operation. When abnormal operation of the network with respect to communication between the two nodes is detected, the problem is diagnosed by separately analyzing the performance of each of the nodes and each of the links connecting the two nodes to isolate the abnormal operation.

In general, in another aspect, the invention features a method of timing the duration of a transaction of interest occurring in the course of communication between nodes of a network, the beginning of the transaction being defined by the sending of a first packet of a particular kind from one node to the other, and the end of the transaction being defined by the sending of another packet of a particular kind between the nodes. In the method, packets transmitted in the network are monitored passively and in real time. The beginning time of the transaction is determined based on the appearance of the first packet. A determination is made of when the other packet has been transmitted. The timing of the duration of the transaction is ended upon the appearance of the other packet.

- 7 -

In general, in another aspect, the invention features, tracking node address to node name mappings in a network of nodes of the kind in which each node has a possibly nonunique node name and a unique node address within the network and in which node addresses can be assigned and reassigned to node names dynamically using a name binding protocol message incorporated within a packet. In the method, packets transmitted in the network are monitored, and a table linking node names to node addresses is updated based on information contained in the name binding protocol messages in the packets.

One advantage of the invention is that it enables a network manager to passively monitor multi-protocol networks at multiple layers of the communications. In addition, it organizes and presents network performance statistics in terms of dialogs which are occurring at any desired level of the communication. This technique of organizing and displaying network performance statistics provides an effective and useful view of network performance and facilitates a quick diagnosis of network problems.

Other advantages and features will become apparent from the following description of the preferred embodiment and from the claims.

25 Description of the Preferred Embodiments

Fig. 1 is a block diagram of a network;

Fig. 2 shows the layered structure of a network communication and a protocol tree within that layered environment;

30 Fig. 3 illustrates the structure of an ethernet/IP/TCP packet;

Fig. 4 illustrates the different layers of a communication between two nodes;

Fig. 5 shows the software modules within the
35 Monitor;

- 8 -

Fig. 6 shows the structure of the Monitor software in terms of tasks and intertask communication mechanisms;

Figs. 7a-c show the STATS data structures which store performance statistics relating to the the data
5 link layer;

Fig. 8 is a event/state table describing the operation of the state machine for a TCP connection;

Fig. 9a is a history data structure that is identified by a pointer found in the appropriate dialog
10 statistics data within STATS;

Fig. 9b is a record from the history table;

Fig. 10 is a flow diagram of the Look_for_Data_State routine;

Fig. 11 is a flow diagram of the Look_for_Initiator routine that is called by the
15 Look_for_Data_State routine;

Fig. 12 is a flow diagram of the Look_for_Retransmission routine which is called by the
Look_at_History routine;

Fig. 13 is a diagram of the major steps in
20 processing a frame through the Real Time Parser (RTP);

Fig. 14 is a diagram of the major steps in the processing a statistics threshold event;

Fig. 15 is a diagram of the major steps in the
25 processing of a database update;

Fig. 16 is a diagram of the major steps in the processing of a monitor control request;

Fig. 17 is a logical map of the network as displayed by the Management Workstation;

Fig. 18 is a basic summary tool display screen;
30

Fig. 19 is a protocol selection menu that may be invoked through the summary tool display screen;

Figs. 20a-g are examples of the statistical variables which are displayed for different protocols;

- 9 -

Fig. 21 is an example of information that is displayed in the dialogs panel of the summary tool display screen;

Fig. 22 is a basic data screen presenting a rate values panel, a count values panel and a protocols seen panel;

Fig. 23 is a traffic matrix screen;

Fig. 24 is a flow diagram of the algorithm for adaptively establishing network thresholds based upon actual network performance;

Fig. 25 is a simple multi-segment network;

Fig. 26 is a flow diagram of the operation of the diagnostic analyzer algorithm;

Fig. 27 is a flow diagram of the source node analyzer algorithm;

Fig. 28 is a flow diagram of the sink node analyzer algorithm;

Fig. 29 is a flow diagram of the link analysis logic;

Fig. 30 is a flow diagram of the DLL problem checking routine;

Fig. 31 is a flow diagram of the IP problem checking routine;

Fig. 32 is a flow diagram of the IP link component problem checking routine;

Fig. 33 is a flow diagram of the DLL link component problem checking routine;

Fig. 34 shows the structure of the event timing database;

Fig. 35 is a flow diagram of the operation of the event timing module (ETM) in the Network Monitor;

Fig. 36 is a network which includes an Appletalk® segment;

Fig. 37 is a Name Table that is maintained by the Address Tracking Module (ATM);

- 10 -

Fig. 38 is a flow diagram of the operation of the ATM; and

Fig. 39 is a flow diagram of the operation of the ATM.

5 Also attached hereto before the claims are the following appendices:

Appendix I identifies the SNMP MIB subset that is supported by the Monitor and the Management Workstation (2 pages);

10 Appendix II defines the extension to the standard MIB that are supported by the Monitor and the Management Workstation (25 pages);

Appendix III is a summary of the protocol variables for which the Monitor gathers statistics and a
15 brief description of the variables, where appropriate (17 pages);

Appendix IV is a list of the Summary Tool Values Display Fields with brief descriptions (2 pages); and

20 Appendix V is a description of the actual screens for the Values Tool (34 pages).

Structure and Operation

The Network:

A typical network, such as the one shown in Fig. 1, includes at least three major components, namely,
25 network nodes 2, network elements 4 and communication lines 6. Network nodes 2 are the individual computers on the network. They are the very reason the network exists. They include but are not limited to workstations (WS), personal computers (PC), file servers (FS), compute
30 servers (CS) and host computers (e.g., a VAX), to name but a few. The term server is often used as though it was different from a node, but it is, in fact, just a node providing special services.

In general, network elements 4 are anything that
35 participate in the service of providing data movement in

- 11 -

a network, i.e., providing the basic communications. They include, but are not limited to, LAN's, routers, bridges, gateways, multiplexors, switches and connectors. Bridges serve as connections between different network
5 segments. They keep track of the nodes which are connected to each of the segments to which they are connected. When they see a packet on one segment that is addressed to a node on another of their segments, they grab the packet from the one segment and transfer it to
10 the proper segment. Gateways generally provide connections between different network segments that are operating under different protocols and serve to convert communications from one protocol to the other. Nodes send packets to routers so that they may be directed over
15 the appropriate segments to the intended destination node.

Finally, network or communication lines 6 are the components of the network which connect nodes 2 and
elements 4 together so that communications between nodes 2
20 may take place. They can be private lines, satellite lines or Public Carrier lines. They are expensive resources and are usually managed as separate entities. Often networks are organized into segments 8 that are connected by network elements 4. A segment 8 is a
25 section of a LAN connected at a physical level (this may include repeaters). Within a segment, no protocols at layers above the physical layer are needed to enable signals from two stations on the same segment to reach each other (i.e., there are no routers, bridges,
30 gateways...).

The Network Monitor and the Management Workstation:

In the described embodiment, there are two basic elements to the monitoring system which is to be described, namely, a Network Monitor 10 and a Management

- 12 -

Workstation 12. Both elements interact with each other over the local area network (LAN).

Network Monitor 10 (referred to hereinafter simply as Monitor 10) is the data collection module which is
5 attached to the LAN. It is a high performance real time front end processor which collects packets on the network and performs some degree of analysis to search for actual or potential problems and to maintain statistical
10 information for use in later analysis. In general, it performs the following functions. It operates in a promiscuous mode to capture and analyze all packets on the segment and it extracts all items of interest from the frames. It generates alarms to notify the Management
Workstation of the occurrence of significant events. It
15 receives commands from the Management Workstation, processes them appropriately and returns responses.

Management Workstation 12 is the operator interface. It collects and presents troubleshooting and performance information to the user. It is based on the
20 SunNet Manager (SNM) product and provides a graphical network-map-based interface and sophisticated data presentation and analysis tools. It receives information from Monitor 10, stores it and displays the information in various ways. It also instructs Monitor 10 to perform
25 certain actions. Monitor 10, in turn, sends responses and alarms to Management Workstation 12 over either the primary LAN or a backup serial link 14 using SNMP with the MIB extensions defined later.

These devices can be connected to each other over
30 various types of networks and are not limited to connections over a local area network. As indicated in Fig. 1, there can be multiple Workstations 12 as well as multiple Monitors 10.

Before describing these components in greater
35 detail, background information will first be reviewed

- 13 -

regarding communication protocols which specify how communications are conducted over the network and regarding the structure of the packets.

The Protocol Tree:

5 As shown in Fig. 2, communication over the network is organized as a series of layers or levels, each one built upon the next lower one, and each one specified by one or more protocols (represented by the boxes). Each layer is responsible for handling a different phase of
10 the communication between nodes on the network. The protocols for each layer are defined so that the services offered by any layer are relatively independent of the services offered by the neighbors above and below. Although the identities and number of layers may differ
15 depending on the network (i.e., the protocol set defining communication over the network), in general, most of them share a similar structure and have features in common.

For purposes of the present description, the Open Systems Interconnection (OSI) model will be presented as
20 representative of structured protocol architectures. The OSI model, developed by the International Organization for Standardization, includes seven layers. As indicated in Fig. 2, there is a physical layer, a data link layer (DLL), a network layer, a transport layer, a session
25 layer, a presentation layer and an application layer, in that order. As background for what is to follow, the function of each of these layers will be briefly described.

The physical layer provides the physical medium
30 for the data transmission. It specifies the electrical and mechanical interfaces of the network and deals with bit level detail. The data link layer is responsible for ensuring an error-free physical link between the communicating nodes. It is responsible for creating and
35 recognizing frame boundaries (i.e., the boundaries of the

- 14 -

packets of data that are sent over the network.) The network layer determines how packets are routed within the network. The transport layer accepts data from the layer above it (i.e., the session layer), breaks the

5 packets up into smaller units, if required, and passes these to the network layer for transmission over the network. It may insure that the smaller pieces all arrive properly at the other end. The session layer is the user's interface into the network. The user must

10 interface with the session layer in order to negotiate a connection with a process in another machine. The presentation layer provides code conversion and data reformatting for the user's application. Finally, the application layer selects the overall network service for

15 the user's application.

Fig. 2 also shows the protocol tree which is implemented by the described embodiment. A protocol tree shows the protocols that apply to each layer and it identifies by the tree structure which protocols at each

20 layer can run "on top of" the protocols of the next lower layer. Though standard abbreviations are used to identify the protocols, for the convenience of the reader, the meaning of the abbreviations are as follows:

25	ARP	Address Resolution Protocol
	ETHERNET	Ethernet Data Link Control
	FTP	File Transfer Protocol
	ICMP	Internet Control Message Protocol
	IP	Internet Protocol
	LLC	802.2 Logical Link Control
30	MAC	802.3 CSMA/CD Media Access Control
	NFS	Network File System
	NSP	Name Server Protocol
	RARP	Reverse Address Resolution Protocol
	SMTP	Simple Mail Transfer Protocol
35	SNMP	Simple Network Management Protocol

- 15 -

TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
UDP	User Datagram Protocol

Two terms are commonly used to describe the protocol
5 tree, namely, a protocol stack and a protocol family (or
suite). A protocol stack generally refers to the
underlying protocols that are used when sending a message
over a network. For example, FTP/TCP/IP/LLC is a
protocol stack. A protocol family is a loose association
10 of protocols which tend to be used on the same network
(or derive from a common source). Thus, for example, the
TCP/IP family includes IP, TCP, UDP, ARP, TELNET and FTP.
The Decnet family includes the protocols from Digital
Equipment Corporation. And the SNA family includes the
15 protocols from IBM.

The Packet:

The relevant protocol stack defines the structure
of each packet that is sent over the network. Fig. 3,
which shows an TCP/IP packet, illustrates the typical
20 structure of a packet. In general, each level of the
protocol stack takes the data from the next higher level
and adds header information to form a protocol data unit
(PDU) which it passes to the next lower level. That is,
as the data from the application is passed down through
25 the protocol layers in preparation for transmission over
the network, each layer adds its own information to the
data passed down from above until the complete packet is
assembled. Thus, the structure of a packet resembles
that of an onion, with each PDU of a given layer wrapped
30 within the PDU of the adjacent lower level.

At the ethernet level, the PDU includes a
destination address (DEST MAC ADDR), a source address
(SRC MAC ADDR), a type (TYPE) identifying the protocol
which is running on top of this layer, and a DATA field
35 for the PDU from the IP layer.

- 16 -

Like the ethernet packet, the PDU for the IP layer includes an IP header plus a DATA field. The IP header includes a type field (TYPE) for indicating the type of service, a length field (LGTH) for specifying the total
5 length of the PDU, an identification field (ID), a protocol field (PROT) for identifying the protocol which is running on top of the IP layer (in this case, TCP), a source address field (SRC ADDR) for specifying the IP address of the sender, a destination address field (DEST
10 ADDR) for specifying the IP address of the destination node, and a DATA field.

The PDU built by the TCP protocol also consists of a header and the data passed down from the next higher layer. In this case the header includes a source port
15 field (SRC PORT) for specifying the port number of the sender, a destination port field (DEST PORT) for specifying the port number of the destination, a sequence number field (SEQ NO.) for specifying the sequence number of the data that is being sent in this packet, and an
20 acknowledgment number field (ACK NO.) for specifying the number of the acknowledgment being returned. It also includes bits which identify the packet type, namely, an acknowledgment bit (ACK), a reset connection bit (RST), a synchronize bit (SYN), and a no more data from sender bit
25 (FIN). There is also a window size field (WINDOW) for specifying the size of the window being used.

The Concept of a Dialog:

The concept of a dialog is used throughout the following description. As will become apparent, it is a
30 concept which provides a useful way of conceptualizing, organizing and displaying information about the performance of a network - for any protocol and for any layer of the multi-level protocol stack.

As noted above, the basic unit of information in
35 communication is a packet. A packet conveys meaning

- 17 -

between the sender and the receiver and is part of a larger framework of packet exchanges. The larger exchange is called a dialog within the context of this document. That is, a dialog is a communication between a sender and a receiver, which is composed of one or more packets being transmitted between the two. There can be multiple senders and receivers which can change roles. In fact, most dialogs involve exchanges in both directions.

10 Stated another way, a dialog is the exchange of messages and the associated meaning and state that is inherent in any particular exchange at any layer. It refers to the exchange between the peer entities (hardware or software) in any communication. In those situations where there is a layering of protocols, any particular message exchange could be viewed as belonging to multiple dialogs. For example, in Fig. 4 Nodes A and B are exchanging packets and are engaged in multiple dialogs. Layer 1 in Node A has a dialog with Layer 1 in Node B. For this example, one could state that this is the data link layer and the nature of the dialog deals with the message length, number of messages, errors and perhaps the guarantee of the delivery. Simultaneously, Layer n of Node A is having a dialog with Layer n of node B. For the sake of the example, one could state that this is an application layer dialog which deals with virtual terminal connections and response rates. One can also assume that all of the other layers (2 through n-1) are also having simultaneous dialogs.

30 In some protocols there are explicit primitives that deal with the dialog and they are generally referred to as connections or virtual circuits. However, dialogs exist even in stateless and connectionless protocols. Two more examples will be described to help clarify the concept further, one dealing with a connection oriented

35

- 18 -

protocol and the other dealing with a connectionless protocol.

In a typical connection oriented protocol, Node A sends a connection request (CR) message to Node B. The
5 CR is an explicit request to form a connection. This is the start of a particular dialog, which is no different from the start of the connection. Nodes A and B could have other dialogs active simultaneously with this particular dialog. Each dialog is seen as unique. A
10 connection is a particular type of dialog.

In a typical connectionless protocol, Node A sends Node B a message that is a datagram which has no connection paradigm, in fact, neither do the protocol(s) at higher layers. The application protocol designates
15 this as a request to initiate some action. For example, a file server protocol such as Sun Microsystems' Network File System (NFS) could make a mount request. A dialog comes into existence once the communication between Nodes
A and B has begun. It is possible to determine that
20 communication has occurred and to determine the actions being requested. If in fact there exists more than one communication thread between Nodes A and B, then these would represent separate, different dialogs.

Inside the Network Monitor:

25 Monitor 10 includes a MIPS R3000 general purpose microprocessor (from MIPS Computer Systems, Inc.) running at 25 MHz. It is capable of providing 20 mips processing power. Monitor 10 also includes a 64Kbyte instruction cache and a 64Kbyte data cache, implemented by SRAM.

30 The major software modules of Monitor 10 are implemented as a mixture of tasks and subroutine libraries as shown in Fig. 5. It is organized this way so as to minimise the context switching overhead incurred during critical processing sequences. There is NO
35 PREEMPTION of any module in the monitor subsystem. Each

- 19 -

module is cognizant of the fact that it should return control to the kernel in order to let other tasks run. Since the monitor subsystem is a closed environment, the software is aware of real time constraints.

5 Among the major modules which make up Monitor 10 is a real time kernel 20, a boot/load module 22, a driver 24, a test module 26, an SNMP Agent 28, a Timer module 30, a real time parser (RTP) 32, a Message Transport Module (MTM) 34, a statistics database (STATS) 36, an
10 Event Manager (EM) 38, an Event Timing Module (ETM) 40 and a control module 42. Each of these will now be described in greater detail.

 Real Time Kernel 20 takes care of the general housekeeping activities in Monitor 10. It is responsible
15 for scheduling, handling intertask communications via queues, managing a potentially large number of timers, manipulating linked lists, and handling simple memory management.

 Boot/Load Module 22, which is FProm based, enables
20 Monitor 10 to start itself when the power is turned on in the box. It initializes functions such as diagnostics, and environmental initialization and it initiates down loading of the Network Monitor Software including program and configuration files from the Management Workstation.
25 Boot/load module 22 is also responsible for reloading program and/or configuration data following internal error detection or on command from the Management Workstation. To accomplish down loading, boot/load module 22 uses the Trivial File Transfer Protocol (TFTP).
30 The protocol stack used for loading is TFTP/UDP/IP/ethernet over the LAN and TFTP/UDP/IP/SLIP over the serial line.

 Device Driver 24 manages the network controller hardware so that Monitor 10 is able to read and write
35 packets from the network and it manages the serial

- 20 -

interface. It does so both for the purposes of monitoring traffic (promiscuous mode) and for the purposes of communicating with the Management Workstation and other devices on the network. The communication
5 occurs through the network controller hardware of the physical network (e.g. Ethernet). The drivers for the LAN controller and serial line interface are used by the boot load module and the MTM. They provide access to the chips and isolate higher layers from the hardware
10 specifics.

Test module 26 performs and reports results of physical layer tests (TDR, connectivity,...) under control of the Management Workstation. It provides traffic load information in response to user requests
15 identifying the particular traffic data of interest. The load information is reported either as a percent of available bandwidth or as frame size(s) plus rate.

SNMP Agent 28 translates requests and information into the network management protocol being used to
20 communicate with the Management Workstation, e.g., the Simple Network Management Protocol (SNMP).

Control Module 42 coordinates access to monitor control variables and performs actions necessary when these are altered. Among the monitor control variables
25 which it handles are the following:

set reset monitor - transfer control to reset logic;

set time of day - modify monitor hardware clock and generate response to Management Workstation;

30 get time of day - read monitor hardware clock and generate response to Workstation;

- 21 -

set trap permit - send trap control ITM to EM and
generate response to Workstation;

get trap permit - generate response to
Workstation;

5 Control module 42 also updates parse control records
within STATS when invoked by the RTP (to be described) or
during overload conditions so that higher layers of
parsing are dropped until the overload situation is
resolved. When overload is over it restores full
10 parsing.

Timer 30 is invoked periodically to perform
general housekeeping functions. It pulses the watchdog
timer at appropriate intervals. It also takes care of
internal time stamping and kicking off routines like the
15 EM routine which periodically recalculates certain
numbers within the statistical database (i.e., STATS).

Real Time Parser (RTP) 32 sees all frames on the
network and it determines which protocols are being used
and interprets the frames. The RTP includes a protocol
20 parser and a state machine. The protocol parser parses a
received frame in the "classical" manner, layer-by-layer,
lowest layer first. The parsing is performed such that
the statistical objects in STATS (i.e., the network
parameters for which performance data is kept) are
25 maintained. Which layers are to have statistics stored
for them is determined by a parse control record that is
stored in STATS (to be described later). As each layer
is parsed, the RTP invokes the appropriate functions in
the statistics module (STATS) to update those statistical
30 objects which must be changed.

The state machine within RTP 32 is responsible for
tracking state as appropriate to protocols and
connections. It is responsible for maintaining and
updating the connection oriented statistical elements in

- 22 -

STATS. In order to track connection states and events, the RTP invokes a routine within the state machine. This routine determines the state of a connection based on past observed frames and keeps track of sequence numbers. 5 It is the routine that determines if a connection is in data transfer state and if a retransmission has occurred. The objectives of the state machine are to keep a brief history of events, state transitions, and sequence numbers per connection; to detect data transfer state so 10 that sequence tracking can begin; and to count inconsistencies but still maintain tracking while falling into an appropriate state (e.g. unknown).

RTP 32 also performs overload control by determining the number of frames awaiting processing and 15 invoking control module 42 to update the parse control records so as to reduce the parsing depth when the number becomes too large.

Statistics Module (STATS) 36 is where Monitor 10 keeps information about the statistical objects it is 20 charged with monitoring. A statistical object represents a network parameter for which performance information is gathered. This information is contained in an extended MIB (Management Information Base), which is updated by RTP 32 and EM 38.

25 STATS updates statistical objects in response to RTP invocation. There are at least four statistical object classes, namely, counters, timers, percentages (%), and meters. Each statistical object is implemented as appropriate to the object class to which it belongs. 30 That is, each statistical object behaves such that when invoked by RTP 32 it updates and then generates an alarm if its value meets a preset threshold. (Meets means that for a high threshold the value is equal to or greater than the threshold and for a low threshold the value is

- 23 -

equal to or less than the threshold. Note that a single object may have both high and low thresholds.)

STATS 36 is responsible for the maintenance and initial analysis of the database. This includes
5 coordinating access to the database variables, ensuring appropriate interlocks are applied and generating alarms when thresholds are crossed. Only STATS 36 is aware of the internal structure of the database, the rest of the system is not.

10 STATS 36 is also responsible for tracking events of interest in the form of various statistical reductions. Examples are counters, rate meters, and rate of change of rate meters. It initiates events based on
15 particular statistics reaching configured limits, i.e., thresholds. The events are passed to the EM which sends a trap (i.e., an alarm) to the Management Workstation. The statistics within STATS 36 are readable from the Management Workstation on request.

STATS performs lookup on all addressing fields.
20 It assigns new data structures to address field values not currently present. It performs any hashing for fast access to the database. More details will be presented later in this document.

Event Manager (EM) 38 extracts statistics from
25 STATS and formats it in ways that allow the Workstation to understand it. It also examines the various statistics to see if their behavior warrants a notification to the Management Workstation. If so, it uses the SNMP Agent software to initiate such
30 notifications.

If the Workstation asks for data, EM 38 gets the data from STATS and sends it to the Workstation. It also performs some level of analysis for statistical, accounting and alarm filtering and decides on further
35 action (e.g. delivery to the Management Workstation).

- 24 -

EM 38 is also responsible for controlling the delivery of events to the Management Workstation, e.g., it performs event filtering. The action to be taken on receipt of an event (e.g. threshold exceeded in STATS) is specified by the event action associated with the threshold. The event is used as an index to select the defined action (e.g. report to Workstation, run local routine xxxx, ignore). The action can be modified by commands from the Management Workstation (e.g., turn off an alarm) or by the control module in an overload situation. An update to the event action, however, does not affect events previously processed even if they are still waiting for transmission to the Management Workstation. Discarded events are counted as such by EM 38.

EM 38 also implements a throttle mechanism to limit the rate of delivery of alarms to the console based on configured limits. This prevents the rapid generation of multiple alarms. In essence, Monitor 10 is given a maximum frequency at which alarms may be sent to the Workstation. Although alarms in excess of the maximum frequency are discarded, a count is kept of the number of alarms that were discarded.

EM 38 invokes routines from the statistics module (STATS) to perform periodic updates such as rate calculations and threshold checks. It calculates time averages, e.g., average traffic by source stations, destination stations. EM 38 requests for access to monitor control variables are passed to the control module.

EM 38 checks whether asynchronous traps (i.e., alarms) to the Workstation are permitted before generating any.

EM 38 receives database update requests from the Management Workstation and invokes the statistics module (STATS) to process these.

- 25 -

Message Transport Module (MTM) 34, which is DRAM based, has two distinct but closely related functions. First, it is responsible for the conversion of Workstation commands and responses from the internal
5 format used within Monitor 10 to the format used to communicate over the network. It isolates the rest of the system from the protocol used to communicate within Management Workstation. It translates between the internal representation of data and ASN.1 used for SNMP.
10 It performs initial decoding of Workstation requests and directs the requests to appropriate modules for processing. It implements SNMP/UDP/IP/LLC or ETHERNET protocols for LAN and SNMP/UDP/IP/SLIP protocols for serial line. It receives network management commands
15 from the Management Workstation and delivers these to the appropriate module for action. Alarms and responses destined for the Workstation are also directed via this module.

Second, MTM 34 is responsible for the delivery and
20 reception of data to and from the Management Workstation using the protocol appropriate to the network. Primary and backup communication paths are provided transparently to the rest of the monitor modules (e.g. LAN and dial up link). It is capable of full duplex delivery of messages
25 between the console and monitoring module. The messages carry event, configuration, test and statistics data.

Event Timing Module (ETM) 40 keeps track of the start time and end times of user specified transactions over the network. In essence, this module monitors the
30 responsiveness of the network at any protocol or layer specified by the user.

Address Tracking Module 42 keeps track of the node name to node address bindings on networks which implement dynamic node addressing protocols.

- 26 -

Memory management for Monitor 10 is handled in accordance with following guidelines. The available memory is divided into four blocks during system initialization. One block includes receive frame buffers. They are used for receiving LAN traffic and for receiving secondary link traffic. These are organized as linked lists of fixed sized buffers. A second block includes system control message blocks. They are used for intertask messages within Monitor 10 and are organized as a linked list of free blocks and multiple linked lists of in process intertask messages. A third block includes transmit buffers. They are used for creation and transmission of workstation alarms and responses and are organized as a linked list of fixed sized buffers. A fourth block is the statistics. This is allocated as a fixed size area at system initialization and managed by the statistics module during system operation.

Task Structure of Monitor;

The structure of the Monitor in terms of tasks and intertask messages is shown in Fig. 6. The rectangular blocks represent interrupt service routines, the ovals represent tasks and the circles represent input queues.

Each task in the system has a single input queue which it uses to receive all input. All inter-process communications take place via messages placed onto the input queue of the destination task. Each task waits on a (well known) input queue and processes events or inter-task messages (i.e., ITM's) as they are received. Each task returns to the kernel within an appropriate time period defined for each task (e.g. after processing a fixed number of events).

Interrupt service routines (ISR's) run on receipt of hardware generated interrupts. They invoke task level

- 27 -

processing by sending an ITM to the input queue of the appropriate task.

The kernel scheduler acts as the base loop of the system and calls any runnable tasks as subroutines. The
5 determination of whether a task is runnable is made from the input queue, i.e., if this has an entry the task has work to perform. The scheduler scans the input queues for each task in a round robin fashion and invokes a task with input pending. Each task processes items from its
10 input queue and returns to the scheduler within a defined period. The scheduler then continues the scan cycle of the input queues. This avoids any task locking out others by processing a continuously busy input queue. A task may be given an effectively higher priority by
15 providing it with multiple entries in the scan table.

Database accesses are generally performed using access routines. This hides the internal structure of the database from other modules and also ensures that appropriate interlocks are applied to shared data.

20 The EM processes a single event from the input queue and then returns to the scheduler.

The MTM Xmit task processes a single event from its input queue and then returns control to the scheduler. The MTM Recv task processes events from the
25 input queue until it is empty or a defined number (e.g. 10) events have been processed and then returns control to the scheduler.

The timer task processes a single event from the input queue and then returns control to the scheduler.

30 RTP continues to process frames until the input queue is empty or it has processed a defined number (e.g. 10) frames. It then returns to the scheduler.

The following sections contain a more detailed description of some of the above-identified software
35 modules.

- 28 -

The Statistics Module (STATS):

The functions of the statistics module are:

- * to define statistics records;
- * to allocate and initialize statistics records;
- 5 * to provide routines to lookup statistics records,
e.g. lookup_id_addr;
- * to provide routines to manipulate the statistics
within the records, e.g. stats_age, stats_incr and
stats_rate;
- 10 * to provide routines to free statistics records,
e.g. stats_allocate and stats_deallocate

It provides these services to the Real Time Parser (RTP) module and to the Event Manager (EM) module.

STATS defines the database and it contains
15 subroutines for updating the statistics which it keeps.

STATS contains the type definitions for all
statistics records (e.g. DLL, IP, TCP statistics). It
provides an initialization routine whose major function
is to allocate statistics records at startup from
20 cacheable memory. It provides lookup routines in order
to get at the statistics. Each type of statistics record
has its own lookup routine (e.g. lookup_ip_address) which
returns a pointer to a statistics record of the
appropriate type or NULL.

25 As a received frame is being parsed, statistics
within statistics records need to be manipulated (e.g.
incremented) to record relevant information about the
frame. STATS provides the routines to manipulate those
statistics. For example, there is a routine to update
30 counters. After the counter is incremented/decremented
and if there is a non-zero threshold associated with the
counter, the internal routine compares its value to the
threshold. If the threshold has been exceeded, the Event
Manager is signaled in order to send a trap to the
35 Workstation. Besides manipulating statistics, these

- 29 -

routines, if necessary, signal the Event Manager via an Intertask Message (ITM) to send a trap to the Management Workstation.

The following is an example of some of the
5 statistics records that are kept in STATS.

- o monitor statistics
- o mac statistics for segment
- o llc statistics for segment
- o statistics per ethernet/lsap type for segment
- 10 o ip statistics for segment
- o icmp statistics for segment
- o tcp statistics for segment
- o udp statistics for segment
- o nfs statistics for segment
- 15 o ftp control statistics for segment
- o ftp data statistics for segment
- o telnet statistics for segment
- o smtp statistics for segment
- o arp statistics for segment

- 20 o statistics per mac address
- o statistics per ethernet type/lasp per mac address
- o statistics per ip address (includes icmp)
- o statistics per tcp socket
- 25 o statistics per udp socket
- o statistics per nfs socket
- o statistics per ftp control socket
- o statistics per ftp data socket
- o statistics per telnet socket
- 30 o statistics per smtp socket
- o arp statistics per ip address

- o statistics per mac address pair
- o statistics per ip pair (includes icmp)

- 30 -

- o statistics per tcp connection
 - o statistics per udp pair
 - o statistics per nfs pair
 - o statistics per ftp control connection
 - 5 o statistics per ftp data connection
 - o statistics per telnet connection
 - o statistics per smtp connection
-
- o connection histories per udp and tcp socket

All statistics are organized similarly across protocol
10 types. The details of the data structures for the DLL
level are presented later.

As noted earlier, there are four statistical
object classes (i.e., variables), namely, counts, rates,
percentages (%), and meters. They are defined and
15 implemented as follows.

A count is a continuously incrementing variable
which rolls around to 0 on overflow. It may be reset on
command from the user (or from software). A threshold
may be applied to the count and will cause an alarm when
20 the threshold count is reached. The threshold count
fires each time the counter increments past the threshold
value. For example, if the threshold is set to 5, alarms
are generated when the count is 5, 10, 15,...

A rate is essentially a first derivative of a
25 count variable. The rate is calculated at a period
appropriate to the variable. For each rate variable, a
minimum, maximum and average value is maintained.
Thresholds may be set on high values of the rate. The
maximums and minimums may be reset on command. The
30 threshold event is triggered each time the rate
calculated is in the threshold region.

As commonly used, the % is calculated at a period
appropriate to the variable. For each % variable a

- 31 -

minimum, maximum and average value is maintained. A threshold may be set on high values of the %. The threshold event is triggered each time the % calculated is in the threshold region.

5 Finally, a meter is a variable which may take any discrete value within a defined range. The current value has no correlation to past or future values. A threshold may be set on a maximum and/or minimum value for a meter.

10 The rate and % fields of network event variables are updated differently than counter or meter fields in that they are calculated at fixed intervals rather than on receipt of data from the network.

15 Structures for statistics kept on a per address or per address pair basis are allocated at initialization time. There are several sizes for these structures. Structures of the same size are linked together in a free pool. As a new structure is needed, it is obtained from a free queue, initialized, and linked into an active list. Active lists are kept on a per statistics type
20 basis.

 As an address or address pair (e.g. mac, ip, tcp...) is seen, RTP code calls an appropriate lookup routine. The lookup routine scans active statistics structures to see if a structure has already been
25 allocated for the statistics. Hashing algorithms are used in order to provide for efficient lookup. If no structure has been allocated, the lookup routine examines the appropriate parse control records to determine whether statistics should be kept, and, if so, it
30 allocates a structure of the appropriate size, initializes it and links it into an active list.

 Either the address of a structure or a NULL is returned by these routines. If NULL is returned, the RTP does not stop parsing, but it will not be allowed to

- 32 -

store the statistics for which the structure was requested.

The RTP updates statistics within the data base as it runs. This is done via macros defined for the RTP.

- 5 The macros call on internal routines which know how to manipulate the relevant statistic. If the pointer to the statistics structure is NULL, the internal routine will not be invoked.

The EM causes rates to be calculated. The STATS
10 module supplies routines (e.g. stats_rate) which must be called by the EM in order to perform the rate calculations. It also calls subroutines to reformat the data in the database in order to present it to the Workstation (i.e., in response to a get from the
15 Workstation).

The calculation algorithms for the rate and % fields of network event variables are as follows.

The following rates are calculated in units per second, at the indicated (approximate) intervals:

- 20 1. 10 second intervals:
e.g. DLL frame, byte, ethernet, 802.3, broadcast, multicast rates
2. 60 second intervals
e.g., all DLL error, ethertype/dsap rates
25 all IP rates.
TCP packets, bytes, errors, retransmitted packets, retransmitted bytes, acks, rsts
UDP packet, error, byte rates
FTP file transfer, byte transfer, error rates
- 30 For these rates, the new average replaces the previous value directly. Maximum and minimum values are retained until reset by the user.

The following rates are calculated in units per hour at the indicated time intervals:

- 35 1. 15 minute interval.

- 33 -

e.g., TCP - connection rate
Telnet connection rate
FTP session rate

The hourly rate is calculated from a sum of the
5 last twelve 5 minute readings, as obtained from the
buckets for the pertinent parameter. Each new reading
replaces the oldest of the twelve values maintained.
Maximum and minimum values are retained until reset by
the user.

10 There are a number of other internal routines in
STATS. For example, all statistical data collected by
the Monitor is subject to age out. Thus, if no activity
is seen for an address (or address pair) in the time
period defined for age out, then the data is discarded
15 and the space reclaimed so that it may be recycled. In
this manner, the Monitor is able to use the memory for
active elements rather than stale data. The user can
select the age out times for the different components.
The EM periodically kicks off the aging mechanism to
20 perform this recycling of resources. STATS provides the
routines which the EM calls, e.g. stats_age.

There are also routines in STATS to allocate and
de-allocate Statistics, e.g., stats_allocate and
stats_de-allocate. The allocate routine is called when
25 stations and dialogs are picked up by the Network
Monitor. The de-allocate routine is called by the aging
routines when a structure is to be recycled.

The Data Structures in STATS

The general structure of the database within STATS
30 is illustrated by Figs. 7a-c, which shows information
that is maintained for the Data Link Layer (DLL) and its
organization. A set of data structures is kept for each
address associated with the layer. In this case there
are three relevant addresses, namely a segment address,
35 indicating which segment the node is on, a MAC address

- 34 -

for the node on the segment, and an address which identifies the dialog occurring over that layer. The dialog address is the combination of the MAC addresses for the two nodes which make up the dialog. Thus, the overall data structure has three identifiable components: a segment address data structure (see Fig. 7a), a MAC address data structure (see Fig. 7b) and a dialog data structure (see Fig. 7c).

The segment address structure includes a doubly linked list 102 of segment address records 104, each one for a different segment address. Each segment address record 104 contains a forward and backward link (field 106) for forward and backward pointers to neighboring records and a hash link (field 108). In other words, the segment address records are accessed by either walking down the doubly linked list or by using a hashing mechanism to generate a pointer into the doubly linked list to the first record of a smaller hash linked list. Each record also contains the address of the segment (field 110) and a set of fields for other information. Among these are a flags field 112, a type field 114, a parse_control field 116, and an EM_control field 118. Flags field 112 contains a bit which indicates whether the identified address corresponds to the address of another Network Monitor. This field only has meaning in the MAC address record and not in the segment or dialog address record. Type field 114 identifies the MIB group which applies to this address. Parse control field 116 is a bit mask which indicates what subgroups of statistics from the identified MIB group are maintained, if any. Flags field 112, type field 114 and parse control field 116 make up what is referred to as the parse control record for this MAC address. The Network Monitor uses a default value for parse control field 116 upon initialization or whenever a new node is detected.

- 35 -

The default value turns off all statistics gathering. The statistics gathering for any particular address may subsequently be turned on by the Workstation through a Network Monitor control command that sets the appropriate 5 bits of the parse control field to one.

EM_control field 118 identifies the subgroups of statistics within the MIB group that have changed since the EM last serviced the database to update rates and other variables. This field is used by the EM to 10 identify those parts of STATS which must be updated or for which recalculations must be performed when the EM next services STAT.

Each segment address record 104 also contains three fields for time related information. There is a 15 start_time field 120 for the time that is used to perform some of the rate calculations for the underlying statistics; a first_seen field 122 for the time at which the Network Monitor first saw the communication; and a last_seen field 124 for the time at which the last 20 communication was seen. The last_seen time is used to age out the data structure if no activity is seen on the segment after a preselected period of time elapses. The first_seen time is a statistic which may be of interest to the network manager and is thus retrievable by the 25 Management Workstation for display.

Finally, each segment address record includes a stats_pointer field 126 for a pointer to a DLL segment statistics data structure 130 which contains all of the statistics that are maintained for the segment address. 30 If the bits in parse_control field 116 are all set to off, indicating that no statistics are to be maintained for the address, then the pointer in stats_pointer field 126 is a null pointer.

The list of events shown in data structure 130 of 35 Fig. 7a illustrates the type of data that is collected

- 36 -

for this address when the parse control field bits are set to on. Some of the entries in DLL segment statistics data structure 130 are pointers to buckets for historical data. In the case where buckets are maintained, there
5 are twelve buckets each of which represents a time period of five minutes duration and each of which generally contains two items of information, namely, a count for the corresponding five minute time period and a MAX rate for that time period. MAX rate records any spikes which
10 have occurred during the period and which the user may not have observed because he was not viewing that particular statistic at the time.

At the end of DLL segment statistics data structure 130, there is a protocol_Q pointer 132 to a
15 linked list 134 of protocol statistics records 136 identifying all of the protocols which have been detected running on top of the DLL layer for the segment. Each record 136 includes a link 138 to the next record in the list, the identity of the protocol (field 140), a frames
20 count for the number of frames detected for the identified protocol (field 142); and a frame rate (field 144).

The MAC address data structure is organized in a similar manner to that of the segment data structure (see
25 Fig. 7b). There is a doubly linked list 146 of MAC address records 148, each of which contains the same type of information as is stored in DLL segment address records 104. A pointer 150 at the end of each MAC address record 148 points to a DLL address statistics
30 data structure 152, which like the DLL segment address data structure 130, contains fields for all of the statistics that are gathered for that DLL MAC address. Examples of the particular statistics are shown in Fig. 7b.

- 37 -

At the end of DLL address statistics data structure 152, there are two pointer fields 152 and 154, one for a pointer to a record 158 in a dialog link queue 160, and the other for a pointer to a linked list 162 of protocol statistics records 164. Each dialog link queue entry 158 contains a pointer to the next entry (field 168) in the queue and it contains a dialog_addr pointer 170 which points to an entry in the DLL dialog queue which involves the MAC address. (see Fig. 7c). Protocol statistics records 164 have the same structure and contain the same categories of information as their counterparts hanging off of DLL segment statistics data structure 130.

The above-described design is repeated in the DLL dialog data structures. That is, dialog record 172 includes the same categories of information as its counterpart in the DLL segment address data structure and the MAC address data structure. The address field 174 contains the addresses of both ends of the dialog concatenated together to form a single address. The first and second addresses within the single address are arbitrarily designated nodes 1 and 2, respectively. In the stats_pointer field 176 there is a pointer to a dialog statistics data structure 178 containing the relevant statistics for the dialog. The entries in the first two fields in this data structure (i.e., fields 180 and 182) are designated protocol entries and protocols. Protocol entries is the number of different protocols which have been seen between the two MAC addresses. The protocols that have been seen are enumerated in the protocols field 182.

DLL dialog statistics data structure 178, illustrated by Fig. 7c, includes several additional fields of information which only appear in these structures for dialogs for which state information can be

- 38 -

kept (e.g. TCP connection). The additional fields identify the transport protocol (e.g., TCP) (field 184) and the application which is running on top of that protocol (field 186). They also include the identity of
5 the initiator of the connection (field 188), the state of the connection (field 190) and the reason that the connection was closed, when it is closed (field 192). Finally, they also include a state_pointer (field 194) which points to a history data structure that will be
10 described in greater detail later. Suffice it to say, that the history data structure contains a short history of events and states for each end of the dialog. The state machine uses the information contained in the history data structure to loosely determine what the
15 state of each of the end nodes is throughout the course of the connection. The qualifier "loosely" is used because the state machine does not closely shadow the state of the connection and thus is capable of recovering from loss of state due to lost packets or missed
20 communications.

The above-described structures and organization are used for all layers and all protocols within STATS.
Real Time Parser (RTP)

The RTP runs as an application task. It is
25 scheduled by the Real Time Kernel scheduler when received frames are detected. The RTP parses the frames and causes statistics, state tracking, and tracing operations to be performed.

The functions of the RTP are:

- 30 * obtain frames from the RTP Input Queue;
- * parse the frames;
- * maintain statistics using routines supplied by the STATS module;
- * maintain protocol state information;

- 39 -

- * notify the MTM via an ITM if a frame has been received with the Network Monitor's address as the destination address; and
- * notify the EM via an ITM if a frame has been received with any Network Monitor's address as the source address.

The design of the RTP is straightforward. It is a collection of routines which perform protocol parsing. The RTP interfaces to the Real Time Kernel in order to perform RTP initialization, to be scheduled in order to parse frames, to free frames, to obtain and send an ITM to another task; and to report fatal errors. The RTP is invoked by the scheduler when there is at least one frame to parse. The appropriate parse routines are executed per frame. Each parse routine invokes the next level parse routine or decides that parsing is done. Termination of the parse occurs on an error or when the frame has been completely parsed.

Each parse routine is a separately compilable module. In general, parse routines share very little data. Each knows where to begin parsing in the frame and the length of the data remaining in the frame.

The following is a list of the parse routines that are available within RTP for parsing the different protocols at the various layers.

Data Link Layer Parse - rtp_dll_parse:

This routine handles Ethernet, IEEE 802.3, IEEE 802.2, and SNAP: See RFC 1010, Assigned Numbers for a description of SNAP (Subnetwork Access Protocol).

Address Resolution Protocol Parse - rtp_arp_parse

ARP is parsed as specified in RFC 826.

Internet Protocol Parse - rtp_ip_parse

IP Version 4 is parsed as specified in RFC 791 as amended by RFC 950, RFC 919, and RFC 922.

- 40 -

Internet Control Message Protocol Parse - rtp_icmp_parse

ICMP is parsed as specified in RFC 792.

Unit Data Protocol Parse - rtp_udp_parse

UDP is parsed as specified in RFC 768.

5 Transmission Control Protocol Parse - rtp_tcp_parse

TCP is parsed as specified in RFC 793.

Simple Mail Transfer Protocol Parse - rtp_smtp_parse

SMTP is parsed as specified in RFC 821.

File Transfer Protocol Parse - rtp_ftp_parse

10 FTP is parsed as specified in RFC 959.

Telnet Protocol Parse - rtp_telnet_parse

The Telnet protocol is parsed as specified in RFC
854.

Network File System Protocol Parse - rpt_nfs_parse

15 The NFS protocol is parsed as specified in RFC
1094.

The RTP calls routines supplied by STATS to look
up data structures. By calling these lookup routines,
global pointers to data structures are set up. Following
20 are examples of the pointers to statistics data
structures that are set up when parse routines call
Statistics module lookup routines.

mac_segment, mac_dst_segment, mac_this_segment,
mac_src, mac_dst, mac_dialog
25 ip_src_segment, ip_dst_segment, ip_this_segment,
ip_src, ip_dst, ip_dialog
tcp_src_segment, tcp_dst_segment,
tcp_this_segment,
tcp_src, tcp_dst, tcp_src_socket, tcp_dst_socket,
30 tcp_connection

The mac_src and mac_dst routines return pointers
to the data structures within STATS for the source MAC
address and the destination MAC address, respectively.
The lookup_mac_dialog routine returns a pointer to the
35 data structure within STATS for the dialog between the

- 41 -

two nodes on the MAC layer. The other STATS routines supply similar pointers for data structures relevant to other protocols.

The RTP routines are aware of the names of the
5 statistics that must be manipulated within the data base (e.g. frames, bytes) but are not aware of the structure of the data. When a statistic is to be manipulated, the RTP routine invokes a macro which manipulates the
10 appropriate statistics in data structures. The macros use the global pointers which were set up during the lookup process described above.

After a frame has been parsed (whether the parse was successful or not), the RTP routine examines the destination mac and ip addresses. If either of the
15 addresses is that of the Network Monitor, RTP obtains a low priority ITM, initializes it, and sends the ITM to the MTM task. One of the fields of the ITM contains the address of the buffer containing the frame.

The RTP must hand some received frames to the EM
20 in order to accomplish the autotopology function (described later). After a frame has been parsed (whether the parse was successful or not), the RTP routine examines the source mac and ip addresses. If either of the addresses is that of another Network
25 Monitor, RTP obtains a low priority ITM, initializes it and sends the ITM to the EM task. The address data structure (in particular, the flags field of the parse control record) within STATS for the MAC or the IP address indicates whether the source address is that of
30 another Network Monitor. One of the fields of the ITM contains the address of the buffer containing the frame.

The RTP receives traffic frames from the network for analysis. RTP operation may be modified by sending control messages to the Monitor. RTP first parses these
35 messages, then detects that the messages are destined for

- 42 -

the Monitor and passes them to the MTM task. Parameters which affect RTP operation may be changed by such control messages.

The general operation of the RTP upon receipt of a
5 traffic frame is as follows:

```

    Get next frame from input queue
    get address records for these stations
    For each level of active parsing
    {
10   get pointer to start of protocol header
      call layer parse routine
      determine protocol at next level
      set pointer to start of next layer protocol

      }end of frame parsing
15   if this is a monitor command add to MTM input
      queue
      if this frame is from another monitor, pass
      to EM
      check for overload -if yes tell control

```

20 The State Machine:

In the described embodiment, the state machine determines and keeps state for both addresses of all TCP connections. TCP is a connection oriented transport
25 protocol, and TCP clearly defines the connection in terms of states of the connection. There are other protocols which do not explicitly define the communication in terms of state, e.g. connectionless protocols such as NFS. Nevertheless, even in the connectionless protocols there
30 is implicitly the concept of state because there is an expected order to the events which will occur during the course of the communication. That is, at the very least, one can identify a beginning and an end of the communication, and usually some sequence of events which will occur during the course of the communication. Thus,

- 43 -

even though the described embodiment involves a connection oriented protocol, the principles are applicable to many connectionless protocols or for that matter any protocol for which one can identify a beginning and an end to the communication under that protocol.

Whenever a TCP packet is detected, the RTP parses the information for that layer to identify the event associated with that packet. It then passes the identified event along with the dialog identifier to the state machine. For each address of the two parties to the communication, the state machine determines what the current state of the node is. The code within the state machine determines the state of a connection based upon a set of rules that are illustrated by the event/state table shown in Fig. 8.

The interpretation of the event/state table is as follows. The top row of the table identifies the six possible states of a TCP connection. These states are not the states defined in the TCP protocol specification. The left most column identifies the eight events which may occur during the course of a connection. Within the table is an array of boxes, each of which sits at the intersection of a particular event/state combination. Each box specifies the actions taken by the state machine if the identified event occurs while the connection is in the identified state. When the state machine receives a new event, it may perform three types of action. It may change the recorded state for the node. The state to which the node is changed is specified by the S="STATE" entry located at the top of the box. It may increment or decrement the appropriate counters to record the information relevant to that event's occurrence. (In the table, incrementing and decrementing are signified by the ++ and the -- symbols, respectively, located after the

- 44 -

identity of the variable being updated.) Or the state machine may take other actions such as those specified in the table as start close timer, Look_for_Data_State, or Look_at_History (to be described shortly). The

5 particular actions which the state machine takes are specified in each box. An empty box indicates that no action is taken for that particular event/state combination. Note, however, that the occurrence of an

10 event is also likely to have caused the update of statistics within STATS, if not by the state machine, then by some other part of the RTP. Also note that it may be desirable to have the state machine record other events, in which case the state table would be modified to identify those other actions.

15 Two events appearing on the table deserve further explanation, namely, close timer expires and inactivity timer expires. The close timer, which is specified by TCP, is started at the end of a connection and it establishes a period during which any old packets for the

20 connection which are received are thrown away (i.e., ignored). The inactivity timer is not specified by TCP but rather is part of the Network Monitor's resource management functions. Since keeping statistics for dialogs (especially old dialogs) consumes resources, it

25 is desirable to recycle resources for a dialog if no activity has been seen for some period of time. The inactivity timer provides the mechanism for accomplishing this. It is restarted each time an event for the connection is received. If the inactivity timer expires

30 (i.e., if no event is received before the timer period ends), the connection is assumed to have gone inactive and all of the resources associated with the dialog are recycled. This involves freeing them up for use by other dialogs.

- 45 -

The other states and events within the table differ from but are consistent with the definitions provided by TCP and should be self evident in view of that protocol specification.

5 The event/state table can be read as follows. Assume, for example, that node 1 is in DATA state and the RTP receives another packet from node 1 which it determines to be a TCP FIN packet. According to the entry in the table at the intersection of FIN/DATA (i.e.,
10 event/state), the state machine sets the state of the connection for node 1 to CLOSING, it decrements the active connections counter and it starts the close timer. When the close timer expires, assuming no other events over that connection have occurred, the state machine
15 sets node 1's state to CLOSED and it starts the inactivity timer. If the RTP sends another SYN packet to reinitiate a new connection before the inactive timer expires, the state machine sets node 1's state to CONNECTING (see the SYN/CLOSED entry) and it increments
20 an after close counter.

When a connection is first seen, the Network Monitor sets the state of both ends of the connection to UNKNOWN state. If some number of data and acknowledgment frames are seen from both connection ends, the states of
25 the connection ends may be promoted to DATA state. The connection history is searched to make this determination as will be described shortly.

Referring to Figs. 9a-b, within STATS there is a history data structure 200 which the state machine uses
30 to remember the current state of the connection, the state of each of the nodes participating in the connection and a short history of state related information. History data structure 200 is identified by a state_pointer found at the end of the associated dialog
35 statistics data structure in STATS (see Fig. 7c). Within

- 46 -

history data structure 200, the state machine records the current state of node 1 (field 202), the current state of node 2 (field 206) and other data relating to the corresponding node (fields 204 and 208). The other data
5 includes, for example, the window size for the receive and transmit communications, the last detected sequence numbers for the data and acknowledgment frames, and other data transfer information.

History data structure 200 also includes a history
10 table (field 212) for storing a short history of events which have occurred over the connection and it includes an index to the next entry within the history table for storing the information about the next received event (field 210). The history table is implemented as a
15 circular buffer which includes sufficient memory to store, for example, 16 records. Each record, shown in Fig. 9b, stores the state of the node when the event was detected (field 218), the event which was detected (i.e., received) (field 220), the data field length (field 222),
20 the sequence number (field 224), the acknowledgment sequence number (field 226) and the identity of the initiator of the event, i.e., either node 1 or node 2 or 0 if neither (field 228).

Though the Network Monitor operates in a
25 promiscuous mode, it may occasionally fail to detect or it may, due to overload, lose a packet within a communication. If this occurs the state machine may not be able to accurately determine the state of the connection upon receipt of the next event. The problem
30 is evidenced by the fact that the next event is not what was expected. When this occurs, the state machine tries to recover state by relying on state history information stored in the history table in field 212 to deduce what the state is. To deduce the current state from
35 historical information, the state machine uses one of the

- 47 -

two previously mentioned routines, namely, Look_for_Data_State and Look_at_History.

Referring to Fig. 10, Look_for_Data_State routine 230 searches back through the history one record at a time until it finds evidence that the current state is DATA state or until it reaches the end of the circular buffer (step 232). Routine 230 detects the existence of DATA state by determining whether node 1 and node 2 each have had at least two data events or two acknowledgment combinations with no intervening connect, disconnect or abort events (step 234). If such a sequence of events is found within the history, routine 230 enters both node 1 and node 2 into DATA state (step 236), it increments the active connections counter (step 238) and then it calls a Look_for_Initiator routine to look for the initiator of the connection (step 240). If such a pattern of events is not found within the history, routine 230 returns without changing the state for the node (step 242).

As shown in Fig. 11, Look_for_Initiator routine 240 also searches back through the history to detect a telltale event pattern which identifies the actual initiator of the connection (step 244). More specifically, routine 240 determines whether nodes 1 and 2 each sent connect-related packets. If they did, routine 240 identifies the initiator as the first node to send a connect-related packet (step 246). If the search is not successful, the identity of the connection initiator remains unknown (step 248).

The Look_at_History routine is called to check back through the history to determine whether data transmissions have been repeated. In the case of retransmissions, the routine calls a Look_for_Retransmission routine 250, the operation of which is shown in Fig. 12. Routine 250 searches back through the history (step 252) and checks whether the

- 48 -

same initiator node has sent data twice (step 254). It detects this by comparing the current sequence number of the packet as provided by the RTP with the sequence numbers of data packets that were previously sent as reported in the history table. If a retransmission is spotted, the retransmission counter in the dialog statistics data structure of STATS is incremented (step 256). If the sequence number is not found within the history table, indicating that the received packet does not represent a retransmission, the retransmission counter is not incremented (step 258).

Other statistics such as Window probes and keep alives may also be detected by looking at the received frame, data transfer variables, and, if necessary, the history.

Even if frames are missed by the Network Monitor, because it is not directly "shadowing" the connection, the Network Monitor still keeps useful statistics about the connection. If inconsistencies are detected the Network Monitor counts them and, where appropriate, drops back to UNKNOWN state. Then, the Network Monitor waits for the connection to stabilize or deteriorate so that it can again determine the appropriate state based upon the history table.

25 Principal Transactions of Network Monitor Modules:

The transactions which represent the major portion of the processing load within the Monitor, include monitoring, actions on threshold alarms, processing database get/set requests from the Management Workstation, and processing monitor control requests from the Management Workstation. Each of these mechanisms will now be briefly described.

Monitoring involves the message sequence shown in Fig. 13. In that figure, as in the other figures involving message sequences, the numbers under the

- 49 -

heading SEQ. identify the major steps in the sequence.
The following steps occur:

1. ISR puts Received traffic frame ITM on RTP input queue
- 5 2. request address of pertinent data structure from STATS (get parse control record for this station)
3. pass pointer to RTP
4. update statistical objects by call to statistical update routine in STATS using pointer to pertinent data structure
- 10 5. parse completed - release buffers

The major steps which follow a statistics threshold event (i.e., an alarm event) are shown in Fig.

14. The steps are as follows:

- 15 1. statistical object update causes threshold alarm
2. STATS generates threshold event ITM to event manager (EM)
3. look up appropriate action for this event
4. perform local event processing
- 20 5. generate network alarm ITM to MTM Xmit (if required)
6. format network alarm trap for Workstation from event manager data
7. send alarm to Workstation

25 The major steps in processing of a database update request (i.e., a get/set request) from the Management Workstation are shown in Fig. 15. The steps are as follows:

- 30 1. LAN ISR receives frame from network and passes it to RTP for parsing
2. RTP parses frame as for any other traffic on segment.
3. RTP detects frame is for monitor and sends received Workstation message over LAN ITM to MTM Recv.
- 35

- 50 -

4. MTM Recv processes protocol stack.
5. MTM Recv sends database update request ITM to EM.
6. EM calls STATS to do database read or database write with appropriate IMPB
- 5 7. STATS performs database access and returns response to EM.
8. EM encodes response to Workstation and sends database update response ITM to MTM Xmit
9. MTM Xmit transmits.

10 The major steps in processing of a monitor control request from the Management Workstation are shown in Fig. 16. The steps are as follows:

1. Lan ISR receives frame from network and passes received frame ITM to RTP for parsing.
- 15 2. RTP parses frame as for any other traffic on segment.
3. RTP detects frame is for monitor and sends received workstation message over LAN ITM to MTM Recv.
- 20 4. MTM Recv processes protocol stack and decodes workstation command.
5. MTM Recv sends request ITM to EM.
6. EM calls Control with monitor control IMPB.
- 25 7. Control performs requested operation and generates response to EM.
8. EM sends database update response ITM to MTM Xmit.
9. MTM Xmit encodes response to Workstation and transmits.

The Monitor/Workstation Interface:

30 The interface between the Monitor and the Management Workstation is based on the SNMP definition (RFC 1089 SNMP; RFC 1065 SMI; RFC 1066 SNMP MIB - Note: RFC means Request for Comments). All five SNMP PDU types are supported:

35 get-request

- 51 -

get-next-request
get-response
set-request
trap

5 The SNMP MIB extensions are designed such that where possible a user request for data maps to a single complex MIB object. In this manner, the get-request is simple and concise to create, and the response should contain all the data necessary to build the screen. Thus, if the
10 user requests the IP statistics for a segment this maps to an IP Segment Group.

The data in the Monitor is keyed by addresses (MAC, IP) and port numbers (telnet, FTP). The user may wish to relate his data to physical nodes entered into
15 the network map. The mapping of addresses to physical nodes is controlled by the user (with support from the Management Workstation system where possible) and the Workstation retains this information so that when a user requests data for node 'Joe' the Workstation asks the
20 Monitor for the data for the appropriate address(es). The node to address mapping need not be one to one.

Loading and dumping of monitors uses TFTP (Trivial File Transfer Protocol). This operates over UDP as does SNMP. The Monitor to Workstation interface follows the
25 SNMP philosophy of operating primarily in a polled mode. The Workstation acts as the master and polls the Monitor slaves for data on a regular (configurable) basis.

The information communicated by the SNMP is represented according to that subset of ASN.1 (ISO 8824
30 Specification of ASN.1) defined in the Internet standard Structure of Management Information (SMI - RFC 1065). The subset of the standard Management Information Base (MIB) (RFC 1066 SNMP MIB) which is supported by the Workstation is defined in Appendix III. The added value
35 provided by the Workstation is encoded as enterprise

- 52 -

specific extensions to the MIB as defined in Appendix IV. The format for these extensions follows the SMI recommendations for object identifiers so that the Workstation extensions fall in the subtree

5 1.3.6.1.4.1.x.1. where x is an enterprise specific node identifier assigned by the IAB.

Appendix V is a summary of the network variables for which data is collected by the Monitor for the extended MIB and which can be retrieved by the Workstation. The summary includes short descriptions of the meaning and significance of the variables, where appropriate.

The Management Workstation:

The Management Workstation is a SUN Sparcstation (also referred to as a Sun) available from Sun Microsystems, Inc. It is running the Sun flavor of Unix and uses the Open Look Graphical User Interface (GUI) and the SunNet Manager as the base system. The options required are those to run SunNet Manager with some additional disk storage requirement.

The network is represented by a logical map illustrating the network components and the relationships between them, as shown in Fig. 17. A hierarchical network map is supported with navigation through the layers of the hierarchy, as provided by SNM. The Management Workstation determines the topology of the network and informs the user of the network objects and their connectivity so that he can create a network map. To assist with the map creation process, the Management Workstation attempts to determine the stations connected to each LAN segment to which a Monitor is attached. Automatic determination of segment topology by detecting stations is performed using the autotopology algorithms as described in copending U.S. Patent Application S.N. ***,** entitled "Automatic Topology Monitor for Multi-

- 53 -

Segment Local Area Network" filed on January 14, 1991 (Attorney Docket No. 13283-NE.APP), incorporated herein by reference.

In normal operation, each station in the network is monitored by a single Monitor that is located on its local segment. The initial determination of the Monitor responsible for a station is based on the results of the autotopology mechanism. The user may override this initial default if required.

The user is informed of new stations appearing on any segment in the network via the alarm mechanism. As for other alarms, the user may select whether stations appearing on and disappearing from the network segment generate alarms and may modify the times used in the aging algorithms. When a new node alarm occurs, the user must add the new alarm to the map using the SNM tools. In this manner, the SNM system becomes aware of the nodes.

The sequence of events following the detection of a new node is:

1. the location of the node is determined automatically for the user.
2. the Monitor generates an alarm for the user indicating the new node and providing some or all of the following information:
 - mac address of node
 - ip address of node
 - segment that the node is believed to be located on
 - Monitor to be responsible for the node
3. the user must select the segment and add the node manually using the SNM editor

- 54 -

4. The update to the SNM database will be detected and the file reread. The Workstation database is reconstructed and the parse control records for the Monitors updated if required.
5. The Monitor responsible for the new node has its parse control record updated via SNMP set request(s).

An internal record of new nodes is required for the autotopology. When a new node is reported by a Network Monitor, the Management Workstation needs to have the previous location information in order to know which Network Monitors to involve in autotopology. For example, two nodes with the same IP address may exist in separate segments of the network. The history makes possible the correlation of the addresses and it makes possible duplicate address detection.

Before a new Monitor can communicate with the Management Workstation via SNMP it needs to be added to the SNM system files. As the SNM files are cached in the database, the file must be updated and the SNM system forced to reread it.

Thus, on the detection of a new Monitor the following events need to occur in order to add the Monitor to the Workstation:

1. The Monitor issues a trap to the Management Workstation software and requests code to be loaded from the Sun Microsystems boot/load server.
2. The code load fails as the Monitor is not known to the unix networking software at this time.
3. The Workstation confirms that the new Monitor does not exceed the configured system limits (e.g. 5 Monitors per

- 55 -

- Workstation) and terminates the initialization sequence if limits are exceeded. An alarm is issued to the user indicating the presence of the new Monitor and whether it can be supported.
- 5 4. The user adds the Monitor to the SNMP.HOSTS file of the SNM system, to the etc/hosts file of the Unix networking system and to the SNM map.
 - 10 5. When the files have been updated the user resets the Monitor using the set tool (described later).
 6. The Monitor again issues a trap to the Management Workstation software and
 - 15 requests code to be loaded from the Sun boot/load server.
 7. The code load takes place and the Monitor issues a trap requesting data from the Management Workstation.
 - 20 8. The Monitor data is issued using SNMP set requests.

Note that on receiving the set request, the SNMP proxy rereads in the (updated) SNMP.HOSTS file which now includes the new Monitor. Also note that the SNMP hosts

25 file need only contain the Monitors, not the entire list of nodes in the system.

9. On completion of the set request(s) the Monitor run command is issued by the Workstation to bring the Monitor on line.
- 30 The user is responsible for entering data into the SNM database manually. During operation, the Workstation monitors the file write date for the SNM database. When this is different from the last date read, the SNM database is reread and the Workstation database
- 35 reconstructed. In this manner, user updates to the SNM

- 56 -

database are incorporated into the Workstation database as quickly as possible without need for the user to take any action.

When the Workstation is loaded, the database is
5 created from the data in the SNM file system (which the user has possibly updated). This data is checked for consistency and for conformance to the limits imposed by the Workstation at this time and a warning is generated to the user if any problems are seen. If the data errors
10 are minor the system continues operation; if they are fatal the user is asked to correct them and Workstation operation terminates.

The monitoring functions of the Management Workstation are provided as an extension to the SNM
15 system. They consist of additional display tools (i.e., summary tool, values tool, and set tool) which the user invokes to access the Monitor options and a Workstation event log in which all alarms are recorded.

As a result of the monitoring process, the Monitor
20 makes a large number of statistics available to the operator. These are available for examination via the Workstation tools that are provided. In addition, the Monitor statistics (or a selected subset thereof) can be made visible to any SNMP manager by providing it with
25 knowledge of the extended MIB. A description of the statistics maintained are described elsewhere.

Network event statistics are maintained on a per network, per segment and per node basis. Within a node, statistics are maintained on a per address (as
30 appropriate to the protocol layer - IP address, port number, ...) and per connection basis. Per network statistics are always derived by the Workstation from the per segment variables maintained by the Monitors. Subsets of the basic statistics are maintained on a node
35 to node and segment to segment basis.

- 57 -

If the user requests displays of segment to segment traffic, the Workstation calculates this data as follows. The inter segment traffic is derived from the node to node statistics for the intersecting set of
 5 nodes. Thus, if segment A has nodes 1, 2, and 3 and segment B has nodes 20, 21, and 22, then summing the node to node traffic for

1 -> 20,21,22

2 -> 20,21,22

10 3 -> 20,21,22

produces the required result. On-LAN/off-LAN traffic for segments is calculated by a simply summing node to node traffic for all stations on the LAN and then subtracting this from total segment counts.

15 Alarms are reported to the user in the following ways:

1. Alarms received are logged in a Workstation log.
2. The node which the alarm relates to is highlighted on the map.
- 20 3. The node status change is propagated up through the (map) hierarchy to support the case where the node is not visible on the screen. This is as provided by SNM.

Summary Tool

25 After the user has selected an object from the map and invokes the display tools, the summary tool generates the user's initial screen at the Management Workstation. It presents a set of statistical data selected to give an overview of the operational status of the object (e.g., a
 30 selected node or segment). The Workstation polls the Monitor for the data required by the Summary Tool display screens.

The Summary Tool displays a basic summary tool screen such as is shown in Fig. 18. The summary tool
 35 screen has three panels, namely, a control panel 602, a

- 58 -

values panel 604, and a dialogs panel 606. The control panel includes the indicated mouse activated buttons. The functions of each of the buttons is as follows. The file button invokes a traditional file menu. The view
5 button invokes a view menu which allows the user to modify or tailor the visual properties of the tool. The properties button invokes a properties menu containing choices for viewing and sometimes modifying the properties of objects. The tools button invokes a tools
10 menu which provides access to the other Workstation tools, e.g. Values Tool.

The Update Interval field allows the user to specify the frequency at which the displayed statistics are updated by polling the Monitor. The Update Once
15 button enables the user to retrieve a single screen update. When the Update Once button is invoked not only is the screen updated but the update interval is automatically set to "none".

The type field enables the user to specify the
20 type of network objects on which to operate, i.e., segment or node.

The name button invokes a pop up menu containing an alphabetical list of all network objects of the type selected and apply and reset buttons. The required name
25 can then be selected from the (scrolling) list and it will be entered in the name field of the summary tool when the apply button is invoked. Alternatively, the user may enter the name directly in the summary tool name field.

30 The protocol button invokes a pop up menu which provides an exclusive set of protocol layers which the user may select. Selection of a layer copies the layer name into the displayed field of the summary tool when the apply operation is invoked. An example of a protocol
35 selection menu is shown in Fig. 19. It displays the

- 59 -

available protocols in the form of a protocol tree with multiple protocol families. The protocol selection is two dimensional. That is, the user first selects the protocol family and then the particular layer within that
5 family.

As indicated by the protocol trees shown in Fig. 19, the capabilities of the Monitor can be readily extended to handle other protocol families. The particular ones which are implemented depend upon the
10 needs of the particular network environment in which the Monitor will operate.

The user invokes the apply button to indicate that the selection process is complete and the type, name, protocol, etc. should be applied. This then updates the
15 screen using the new parameter set that the user selected. The reset button is used to undo the selections and restore them to their values at the last apply operation.

The set of statistics for the selected parameter
20 set is displayed in values panel 604. The members of the sets differ depending upon, for example, what protocol was selected. Figs. 20a-g present examples of the types of statistical variables which are displayed for the DLL, IP, UDP, TCP, ICMP, NFS, and ARP/RARP protocols,
25 respectively. The meaning of the values display fields are described in Appendix I, attached hereto.

Dialogs panel 606 contains a display of the connection statistics for all protocols for a selected node. Within the Management Workstation, connection
30 lists are maintained per node, per supported protocol. When connections are displayed, they are sorted on "Last Seen" with the most current displayed first. A single list returned from the Monitor contains all current connection. For TCP, however, each connection also
35 contains a state and TCP connections are displayed as

- 60 -

Past and Present based upon the returned state of the connection. For certain dialogs, such as TCP and NFS over UDP, there is an associated direction to the dialog, i.e., from the initiator (source) to the receiver (sink).
5 For these dialogs, the direction is identified in a DIR. field. A sample of information that is displayed in dialogs panel 606 is presented in Fig. 21 for current connections.

Values Tool

10 The values tool provides the user with the ability to look at the statistical database for a network object in detail. When the user invokes this tool, he may select a basic data screen containing a rate values panel 620, a count values panel 622 and a protocols seen panel
15 626, as shown in Fig. 22, or he may select a traffic matrix screen 628, as illustrated in Fig. 23.

In rate values and count values panels 620 and 622, value tools presents the monitored rate and count statistics, respectively, for a selected protocol. The
20 parameters which are displayed for the different protocols (i.e., different groups) are listed in Appendix II. In general, a data element that is being displayed for a node shows up in three rows, namely, a total for the data element, the number into the data element, and
25 the number out of the data element. Any exceptions to this are identified in Appendix II. Data elements that are displayed for segments, are presented as totals only, with no distinction between Rx and Tx.

When invoked the Values Tool displays a primary
30 screen to the user. The primary screen contains what is considered to be the most significant information for the selected object. The user can view other information for the object (i.e., the statistics for the other parameters) by scrolling down.

- 61 -

The displayed information for the count values and rate values panels 620 and 622 includes the following. An alarm field reports whether an alarm is currently active for this item. It displays as "*" if active alarm is present. A Current Value/Rate field reports the current rate or the value of the counter used to generate threshold alarms for this item. This is reset following each threshold trigger and thus gives an idea of how close to an alarm threshold the variable is. A Typical Value field reports what this item could be expected to read in a "normal" operating situation. This field is filled in for those items where this is predictable and useful. It is maintained in the Workstation database and is modifiable by the user using the set tool. An Accumulated Count field reports the current accumulated value of the item or the current rate. A Max Value field reports the highest value recently seen for the item. This value is reset at intervals defined by a user adjustable parameter (default 30 minutes). This is not a rolling cycle but rather represents the highest value since it was reset which may be from 1 to 30 minutes ago (for a rest period of 30 minutes). It is used only for rates. A Min Value field reports the lowest value recently seen for the item. This operates in the same manner as Max Value field and is used only for rates.

A Percent (%) field reports only for the following variables:

off seg counts:

100(in count / total off seg count)
 100(out count / total off seg count)
 100(transit count / total off seg count)
 100(local count / total off seg count)

off seg rates

100(transit rate / total off seg rate), etc.
 protocols

- 62 -

100(frame rate this protocol / total frame
rate)

On the right half of the basic display, there the
following additional fields: a High Threshold field and a
5 Sample period for rates field.

Set Tool

The set tool provides the user with the ability to
modify the parameters controlling the operation of the
Monitors and the Management Workstation. These
10 parameters affect both user interface displays and the
actual operation of the Monitors. The parameters which
can be operated on by the set tool can be divided into
the following categories: alarm thresholds, monitoring
control, segment Monitor administration, and typical
15 values.

The monitoring control variables specify the
actions of the segment Monitors and each Monitor can have
a distinct set of control variables (e.g., the parse
control records that are described elsewhere). The user
20 is able to define those nodes, segments, dialogs and
protocols in which he is interested so as to make the
best use of memory space available for data storage.
This mechanism allows for load sharing, where multiple
Monitors on the same segment can divide up the total
25 number of network objects which are to be monitored so
that no duplication of effort between them takes place.

The monitor administration variables allow the
user to modify the operation of the segment Monitor in a
more direct manner than the monitoring control variables.
30 Using the set tool, the user can perform those operations
such as reset, time changes etc. which are normally the
prerogative of a system administrator.

Note that the above descriptions of the tools
available through the Management Workstation are not
35 meant to imply that other choices may not be made

- 63 -

regarding the particular information which is displayed and the manner in which it is displayed.

Adaptively Setting Network Monitor Thresholds:

The Workstation sets the thresholds in the Network
5 Monitor based upon the performance of the system as
observed over an extended period of time. That is, the
Workstation periodically samples the output of the
Network Monitors and assembles a model of a normally
10 functioning network. Then, the Workstation sets the
thresholds in the Network Monitors based upon that model.
If the observation period is chosen to be long enough and
since the model represents the "average" of the network
performance over the observation period, temporary
15 undesired deviations from normal behavior are smoothed
out over time and model tends to accurately reflect
normal network behavior.

Referring the Fig. 24, the details of the training
procedure for adaptively setting the Network Monitor
thresholds are as follows. To begin training, the
20 Workstation sends a start learning command to the Network
Monitors from which performance data is desired (step
302). The start learning command disables the thresholds
within the Network Monitor and causes the Network Monitor
to periodically send data for a predefined set of network
25 parameters to the Management Workstation. (Disabling the
thresholds, however, is not necessary. One could have
the learning mode operational in parallel with monitoring
using existing thresholds.) The set of parameters may be
any or all of the previously mentioned parameters for
30 which thresholds are or may be defined.

Throughout the learning period, the Network
Monitor sends "snapshots" of the network's performance to
the Workstation which, in turn, stores the data in a
performance history database 306 (step 304). The network
35 manager sets the length of the learning period.

- 64 -

Typically, it should be long enough to include the full range of load conditions that the network experiences so that a representative performance history is generated. It should also be long enough so that short periods of overload or faulty behavior do not distort the resulting averages.

After the learning period has expired, the network manager, through the Management Workstation, sends a stop learning command to the Monitor (step 308). The Monitor ceases automatically sending further performance data updates to the Workstation and the Workstation processes the data in its performance history database (step 310). The processing may involve simply computing averages for the parameters of interest or it may involve more sophisticated statistical analysis of the data, such as computing means, standard deviations, maximum and minimum values, or using curve fitting to compute rates and other pertinent parameter values.

After the Workstation has statistically analyzed the performance data, it computes a new set of thresholds for the relevant performance parameters (step 312). To do this, it uses formulas which are appropriate to the particular parameter for which a threshold is being computed. That is, if the parameter is one for which one would expect to see wide variations in its value during network monitoring, then the threshold should be set high enough so that the normal expected variations do not trigger alarms. On the other hand, if the parameter is of a type for which only small variations are expected and larger variations indicate a problem, then the threshold should be set to a value that is close to the average observed value. Examples of formulae which may be used to compute thresholds are:

- * Highest value seen during learning period;

- 65 -

- * Highest value seen during learning period + 10%;
- * Highest value seen during learning period + 50%;
- 5 * Highest value seen during learning period + user-defined percent;
- * Any value of the parameter other than zero;
- * Average value seen during learning period + 50%; and
- 10 * Average value seen during learning period + user-defined percent.

As should be evident from these examples, there is a broad range of possibilities regarding how to compute a particular threshold. The choice, however, should
 15 reflect the parameter's importance in signaling serious network problems and its normal expected behavior (as may be evidenced from the performance history acquired for the parameter during the learning mode).

After the thresholds are computed, the Workstation
 20 loads them into the Monitor and instructs the Monitor to revert to normal monitoring using the new thresholds (step 314).

This procedure provides a mechanism enabling the network manager to adaptively reset thresholds in
 25 response to changing conditions on the network, shifting usage patterns and evolving network topology. As the network changes over time, the network manager merely invokes the adaptive threshold setting feature and updates the thresholds to reflect those changes.

30 The Diagnostic Analyzer Module:

The Management Workstation includes a diagnostic analyzer module which automatically detects and diagnoses the existence and cause of certain types of network problems. The functions of the diagnostic module may
 35 actually be distributed among the Workstation and the

- 66 -

Network Monitors which are active on the network. In principle, the diagnostic analyzer module includes the following elements for performing its fault detection and analysis functions.

5 The Management Workstation contains a reference model of a normally operating network. The reference model is generated by observing the performance of the network over an extended period of time and computing averages of the performance statistics that were observed
10 during the observation period. The reference model provides a reference against which future network performance can be compared so as to diagnose and analyze potential problems. The Network Monitor (in particular, the STATS module) includes alarm thresholds on a selected
15 set of the parameters which it monitors. Some of those thresholds are set on parameters which tend to be indicative of the onset or the presence of particular network problems.

 During monitoring, when a Monitor threshold is
20 exceeded, thereby indicating a potential problem (e.g. in a TCP connection), the Network Monitor alerts the Workstation by sending an alarm. The Workstation notifies the user and presents the user with the option of either ignoring the alarm or invoking a diagnostic
25 algorithm to analyze the problem. If the user invokes the diagnostic algorithm, the Workstation compares the current performance statistics to its reference model to analyze the problem and report its results. (Of course, this may also be handled automatically so as to not
30 require user intervention.) The Workstation obtains the data on current performance of the network by retrieving the relevant performance statistics from all of the segment Network Monitors that may have information useful to diagnosing the problem.

- 67 -

The details of a specific example involving poor TCP connection performance will now be described. This example refers to a typical network on which the diagnostic analyzer resides, such as the network
 5 illustrated in Fig. 25. It includes three segments labelled S1, S2, and S3, a router R1 connecting S1 to S2, a router R2 connecting S2 to S3, and at least two nodes, node A on S1 which communicates with node B on S3. On each segment there is also a Network Monitor 324 to
 10 observe the performance of its segment in the manner described earlier. A Management Workstation 320 is also located on S1 and it includes a diagnostic analyzer module 322. For this example, the symptom of the network problem is degraded performance of a TCP connection
 15 between Nodes A and B.

A TCP connection problem may manifest itself in a number of ways, including, for example, excessively high numbers for any of the following:

errors
 20 packets with bad sequence numbers
 packets retransmitted
 bytes retransmitted
 out of order packets
 out of order bytes
 25 packets after window closed
 bytes after window closed
 average and maximum round trip times
 or by an unusually low value for the current window size. By setting the appropriate thresholds, the Monitor is
 30 programmed to recognize any one or more of these symptoms. If any one of the thresholds is exceeded, the Monitor sends an alarm to the Workstation. The Workstation is programmed to recognize the particular alarm as related to an event which can be further
 35 analyzed by its diagnostic analyzer module 322. Thus,

- 68 -

the Workstation presents the user with the option of invoking its diagnostic capabilities (or automatically invokes the diagnostic capabilities).

In general terms, when the diagnostic analyzer is
5 invoked, it looks at the performance data that the
segment Monitors produce for the two nodes, for the
dialogs between them and for the links that interconnect
them and compares that data to the reference model for
the network. If a significant divergence from the
10 reference model is identified, the diagnostic analyzer
informs the Workstation (and the user) about the nature
of the divergence and the likely cause of the problem.
In conducting the comparison to "normal" network
performance, the network circuit involved in
15 communications between nodes A and B is decomposed into
its individual components and diagnostic analysis is
performed on each link individually in the effort to
isolate the problem further.

The overall structure of the diagnostic algorithm
20 400 is shown in Fig. 26. When invoked for analyzing a
possible TCP problem between nodes A and B, diagnostic
analyzer 322 checks for a TCP problem at node A when it
is acting as a source node (step 402). To perform this
check, diagnostic algorithm 400 invokes a source node
25 analyzer algorithm 450 shown in Fig. 27. If a problem is
identified, the Workstation reports that there is a high
probability that node A is causing a TCP problem when
operating as a source node and it reports the results of
the investigation performed by algorithm 450 (step 404).

30 If node A does not appear to be experiencing a TCP
problem when acting as a source node, diagnostic analyzer
322 checks for evidence of a TCP problem at node B when
it is acting as a sink node (step 406). To perform this
check, diagnostic algorithm 400 invokes a sink node
35 analyzer algorithm 470 shown in Fig. 28. If a problem is

- 69 -

identified, the Workstation reports that there is a high probability that node B is causing a TCP problem when operating as a sink node and it reports the results of the investigation performed by algorithm 470 (step 408).

5 Note that source and sink nodes are concepts which apply to those dialogs for which a direction of the communication can be defined. For example, the source node may be the one which initiated the dialog for the purpose of sending data to the other node, i.e., the sink
10 node.

 If node B does not appear to be experiencing a TCP problem when acting as a sink node, diagnostic analyzer 322 checks for evidence of a TCP problem on the link between Node A and Node B (step 410). To perform this
15 check, diagnostic algorithm 400 invokes a link analysis algorithm 550 shown in Fig. 29. If a problem is identified, the Workstation reports that there is a high probability that a TCP problem exists on the link and it reports the results of the investigation performed by
20 link analysis algorithm 550 (step 412).

 If the link does not appear to be experiencing a TCP problem, diagnostic analyzer 322 checks for evidence of a TCP problem at node B when it is acting as a source node (step 414). To perform this check, diagnostic
25 algorithm 400 invokes the previously mentioned source algorithm 450 for Node B. If a problem is identified, the Workstation reports that there is a medium probability that node B is causing a TCP problem when operating as a source node and it reports the results of
30 the investigation performed by algorithm 450 (step 416).

 If node B does not appear to be experiencing a TCP problem when acting as a source node, diagnostic analyzer 322 checks for a TCP problem at node A when it is acting as a sink node (step 418). To perform this check,
35 diagnostic algorithm 400 invokes sink node analyzer

- 70 -

algorithm 470 for Node A. If a problem is identified, the Network Monitor reports that there is a medium probability that node A is causing a TCP problem when operating as a sink node and it reports the results of the investigation performed by algorithm 470 (step 420).

Finally, if node A does not appear to be experiencing a TCP problem when acting as a sink node, diagnostic analyzer 322 reports that it was not able to isolate the cause of a TCP problem (step 422).

10 The algorithms which are called from within the above-described diagnostic algorithm will now be described. Referring to Fig. 27, source node analyzer algorithm 450 checks whether a particular node is causing a TCP problem when operating as a source node. The
15 strategy is as follows. To determine whether a TCP problem exists at this node which is the source node for the TCP connection, look at other connections for which this node is a source. If other TCP connections are
20 okay, then there is probably not a problem with this node. This is an easy check with a high probability of being correct. If no other good connections exist, then look at the lower layers for possible reasons. Start at DLL and work up as problems at lower layers are more
25 fundamental, i.e., they cause problems at higher layers whereas the reverse is not true.

In accordance with this approach, algorithm 450 first determines whether the node is acting as a source node in any other TCP connection and, if so, whether the other connection is okay (step 452). If the node is
30 performing satisfactorily as a source node in another TCP connection, algorithm 450 reports that there is no problem at the source node and returns to diagnostic algorithm 400 (step 454). If algorithm 450 cannot
35 identify any other TCP connections involving this node that are okay, it moves up through the protocol stack

- 71 -

checking each level for a problem. In this case, it then checks for DLL problems at the node when it is acting as a source node by calling an DLL problem checking routine 510 (see Fig. 30) (step 456). If a DLL problem is found, that fact is reported (step 458). If no DLL problems are found, algorithm 450 checks for an IP problem at the node when it is acting as a source by calling an IP problem checking routine 490 (see Fig. 31) (step 460). If an IP problem is found, that fact is reported (step 462). If no IP problems are found, algorithm 450 checks whether any other TCP connection in which the node participates as a source is not okay (step 464). If another TCP connection involving the node exists and it is not okay, algorithm 450 reports a TCP problem at the node (step 466). If no other TCP connections where the node is acting as a source node can be found, algorithm 450 exits.

Referring to Fig. 28, sink node analyzer algorithm 470 checks whether a particular node is causing a TCP problem when operating as a sink node. It first determines whether the node is acting as a sink node in any other TCP connection and, if so, whether the other connection is okay (step 472). If the node is performing satisfactorily as a sink node in another TCP connection, algorithm 470 reports that there is no problem at the source node and returns to diagnostic algorithm 400 (step 474). If algorithm 470 cannot identify any other TCP connections involving this node that are okay, it then checks for DLL problems at the node when it is acting as a sink node by calling DLL problem checking routine 510 (step 476). If a DLL problem is found, that fact is reported (step 478). If no DLL problems are found, algorithm 470 checks for an IP problem at the node when it is acting as a sink by calling IP problem checking routine 490 (step 480). If an IP problem is found, that

- 72 -

fact is reported (step 482). If no IP problems are found, algorithm 470 checks whether any other TCP connection in which the node participates as a sink is not okay (step 484). If another TCP connection involving
5 the node as a sink exists and it is not okay, algorithm 470 reports a TCP problem at the node (step 486). If no other TCP connections where the node is acting as a sink node can be found, algorithm 470 exits.

Referring to Fig. 31, IP problem checking routine
10 490 checks for IP problems at a node. It does this by comparing the IP performance statistics for the node to the reference model (steps 492 and 494). If it detects any significant deviations from the reference model, it reports that there is an IP problem at the node (step
15 496). If no significant deviations are noted, it reports that there is no IP problem at the node (step 498).

As revealed by examining Fig. 30, DLL problem checking routine 510 operates in a similar manner to IP problem checking routine 490, with the exception that it
20 examines a different set of parameters (i.e., DLL parameters) for significant deviations.

Referring the Fig. 29, link analysis logic 550 first determines whether any other TCP connection for the link is operating properly (step 552). If a properly
25 operating TCP connection exists on the link, indicating that there is no link problem, link analysis logic 550 reports that the link is okay (step 554). If a properly operating TCP connection cannot be found, the link is decomposed into its constituent components and an IP link
30 component problem checking routine 570 (see Fig. 32) is invoked for each of the link components (step 556). IP link component problem routine 570 evaluates the link component by checking the IP layer statistics for the relevant link component.

- 73 -

The decomposition of the link into its components arranges them in order of their distance from the source node and the analysis of the components proceeds in that order. Thus, for example, the link components which make up the link between nodes A and B include in order:

5 segment S1, router R1, segment S2, router R2, and segment S3. The IP data for these various components are analyzed in the following order:

10 IP data for segment S1
 IP data for address R1
 IP data for source node to R1
 IP data for S1 to S2
 IP data for S2
 IP data for address R2
 15 IP data for S3
 IP data for S2 to S3
 IP data for S1 to S3

As shown in Fig. 32, IP link component problem checking routine 570 compares IP statistics for the link component to the reference model (step 572) to determine whether network performance deviates significantly from that specified by the model (step 574). If significant deviations are detected, routine 570 reports that there is an IP problem at the link component (step 576).
 25 Otherwise, it reports that it found no IP problem (step 578).

Referring back to Fig. 29, after completing the IP problem analysis for all of the link components, logic 550 then invokes a DLL link component problem checking routine 580 (see Fig. 33) for each link component to check its DLL statistics (step 558).
 30

DLL link problem routine 580 is similar to IP link problem routine 570. As shown in Fig. 33, DLL link problem checking routine 580 compares DLL statistics for the link to the reference model (step 582) to determine
 35

- 74 -

whether network performance at the DLL deviates significantly from that specified by the model (step 584). If significant deviations are detected, routine 580 reports that there is a DLL problem at the link 5 component (step 586). Otherwise, it reports that no DLL problems were found (step 588).

Referring back to Fig. 29, after completing the DLL problem analysis for all of the link components, logic 550 checks whether there is any other TCP on the 10 link (step 560). If another TCP exists on the link (which implies that the other TCP is also not operating properly), logic 550 reports that there is a TCP problem on the link (step 562). Otherwise, logic 550 reports that there was not enough information from the existing 15 packet traffic to determine whether there was a link problem (step 564)

If the analysis of the link components does not isolate the source of the problem and if there were components for which sufficient information was not 20 available (due possibly to lack of traffic over through that component), the user may send test messages to those components to generate the information needed to evaluate its performance.

The reference model against which comparisons 25 are made to detect and isolate malfunctions may be generated by examining the behavior of the network over an extended period of operation or over multiple periods of operation. During those periods of operation, average values and maximum excursions (or standard deviations) 30 for observed statistics are computed. These values provide an initial estimate of a model of a properly functioning system. As more experience with the network is obtained and as more historical data on the various statistics is accumulated the thresholds for detecting 35 actual malfunctions or imminent malfunctions and the

- 75 -

reference model can be revised to reflect the new experience.

What constitutes a significant deviation from the reference model depends upon the particular parameter
 5 involved. Some parameters will not deviate from the expected norm and thus any deviation would be considered to be significant, for example, consider ICMP messages of type "destination unreachable," IP errors, TCP errors. Other parameters will normally vary within a wide range
 10 of acceptable values, and only if they move outside of that range should the deviation be considered significant. The acceptable ranges of variation can be determined by watching network performance over a sustained period of operation.

15 The parameters which tend to provide useful information for identifying and isolating problems at the node level for the different protocols and layers include the following.

TCP

20 error rate
 header byte rate
 packets retransmitted
 bytes retransmitted
 packets after window closed
 25 bytes after window closed

UDP

error rate
 header byte rate

IP

30 error rate
 header byte rate
 fragmentation rate
 all ICMP messages of type destination

- 76 -

unreachable, parameter problem,
redirection

DLL

error rate

5

runts

For diagnosing network segment problems, the above-identified parameters are also useful with the addition of the alignment rate and the collision rate at the DLL. All or some subset of these parameters may be included
10 among the set of parameters which are examined during the diagnostic procedure to detect and isolate network problems.

The above-described technique can be applied to a wide range of problems on the network, including among
15 others, the following:

TCP Connection fails to establish
UDP Connection performs poorly
UDP not working at all
IP poor performance/high error rate
20 IP not working at all
DLL poor performance/high error rate
DLL not working at all

For each of these problems, the diagnostic approach would be similar to that described above, using, of course,
25 different parameters to identify the potential problem and isolate its cause.

The Event Timing Module

Referring again to Fig. 5, the RTP is programmed to detect the occurrence of certain transactions for
30 which timing information is desired. The transactions typically occur within a dialog at a particular layer of the protocol stack and they involve a first event (i.e., an initiating event) and a subsequent partner event or response. The events are protocol messages that arrive

- 77 -

at the Network Monitor, are parsed by the RTP and then passed to Event Timing Module (ETM) for processing. A transaction of interest might be, for example, a read of a file on a server. In that case, the initiating event
5 is the read request and the partner event is the read response. The time of interest is the time required to receive a response to the read request (i.e., the transaction time). The transaction time provides a useful measure of network performance and if measured at
10 various times throughout the day under different load conditions gives a measure of how different loads affect network response times. The layer of the communication protocol at which the relevant dialog takes place will of course depend upon the nature of the event.

15 In general, when the RTP detects an event, it transfers control to the ETM which records an arrival time for the event. If the event is an initiating event, the ETM stores the arrival time in an event timing database 300 (see Fig. 34) for future use. If the event
20 is a partner event, the ETM computes a difference between that arrival time and an earlier stored time for the initiating event to determine the complete transaction time.

Event timing database 300 is an array of records
25 302. Each record 302 includes a dialog field 304 for identifying the dialog over which the transactions of interest are occurring and it includes an entry type field 306 for identifying the event type of interest. Each record 302 also includes a start time field 308 for
30 storing the arrival time of the initiating event and an average delay time field 310 for storing the computed average delay for the transactions. A more detailed description of the operation of the ETM follows.

Referring to Fig. 35, when the RTP detects the
35 arrival of a packet of the type for which timing

- 78 -

information is being kept, it passes control to the ETM along with relevant information from the packet, such as the dialog identifier and the event type (step 320). The ETM then determines whether it is to keep timing

5 information for that particular event by checking the event timing database (step 322). Since each event type can have multiple occurrences (i.e., there can be multiple dialogs at a given layer), the dialog identifier is used to distinguish between events of the same type

10 for different dialogs and to identify those for which information has been requested. All of the dialog/events of interest are identified in the event timing database. If the current dialog and event appear in the event timing database, indicating that the event should be

15 timed, the ETM determines whether the event is a starting event or an ending event so that it may be processed properly (step 324). For certain events, the absence of a start time in the entry field of the appropriate record

20 302 in event timing database 300 is one indicator that the event represents a start time; otherwise, it is an end time event. For other events, the ETM determines if the start time is to be set by the event type as specified in the packet being parsed. For example, if the event is a file read a start time is stored. If the

25 event is the read completion it represents an end time. In general, each protocol event will have its own intrinsic meaning for how to determine start and end times.

Note that the arrival time is only an estimate of

30 the actual arrival time due to possible queuing and other processing delays. Nevertheless, the delays are generally so small in comparison to the transaction times being measured that they are of little consequence.

In step 324, if the event represents a start time,

35 the ETM gets the current time from the kernal and stores

- 79 -

it in start time field 308 of the appropriate record in event timing database 300 (step 326). If the event represents an end time event, the ETM obtains the current time from the kernel and computes a difference between
5 that time and the corresponding start time found in event timing database 300 (step 328). This represents the total time for the transaction of interest. It is combined with the stored average transaction time to compute a new running average transaction time for that
10 event (step 330).

Any one of many different methods can be used to compute the running average transaction time. For example, the following formula can be used:

15
$$\text{New Avg.} = [(5 * \text{Stored Avg.}) + \text{Transaction Time}] / 6.$$

After six transaction have been timed, the computed new average becomes a running average for the transaction times. The ETM stores this computed average in the appropriate record of event timing database 300,
20 replacing the previous average transaction time stored in that record, and it clears start time entry field 308 for that record in preparation for timing the next transaction.

After processing the event in steps 322, 326, and
25 330, the ETM checks the age of all of the start time entries in the event timing database 300 to determine if any of them are too "old" (step 332). If the difference between the current time and any of the start times exceeds a preselected threshold, indicating that a
30 partner event has not occurred within a reasonable period of time, the ETM deletes the old start time entry for that dialog/event (step 334). This insures that a missed packet for a partner event does not result in an erroneously large transaction time which throws off the
35 running average for that event.

- 80 -

If the average transaction time increases beyond a preselected threshold set for timing events, an alarm is sent to the Workstation.

Two examples will now be described to illustrate the operation of the ETM for specific event types. In the first example, Node A of Fig. 25 is communicating with Node B using the NFS protocol. Node A is the client while Node B is the server. The Network Monitor resides on the same segment as node A, but this is not a requirement. When Node A issues a read request to Node B, the Network Monitor sees the request and the RTP within the Network Monitor transfers control to the ETM. Since it is a read, the ETM stores a start time in the Event Timing Database. Thus, the start time is the time at which the read was initiated.

After some delay, caused by the transmission delays of getting the read message to node B, node B performs the read and sends a response back to node A. After some further transmission delays in returning the read response, the Network Monitor receives the second packet for the event. At the time, the ETM recognizes that the event is an end time event and updates the average transaction time entry in the appropriate record with a new computed running average. The ETM then compares the average transaction time with the threshold for this event and if it has been exceeded, issues an alarm to the Workstation.

In the second example, node A is communicating with Node B using the Telnet protocol. Telnet is a virtual terminal protocol. The events of interest take place long after the initial connection has been established. Node A is typing at a standard ASCII (VT100 class) terminal which is logically (through the network) connected to Node B. Node B has an application which is receiving the characters being typed on Node A and, at

- 81 -

appropriate times, indicated by the logic of the applications, sends characters back to the terminal located on Node A. Thus, every time node A sends characters to B, the Network Monitor sees the
5 transmission.

In this case, there are several transaction times which could provide useful network performance information. They include, for example, the amount of time it takes to echo characters typed at the keyboard
10 through the network and back to the display screen, the delay between typing an end of line command and seeing the completion of the application event come back or the network delays incurred in sending a packet and receiving acknowledgment for when it was received.

15 In this example, the particular time being measured is the time it takes for the network to send a packet and receive an acknowledgement that the packet has arrived. Since Telnet runs on top of TCP, which in turn runs on top of IP, the Network Monitor monitors the TCP
20 acknowledge end-to-end time delays.

Note that this is a design choice of the implementation and that all events visible to the Network Monitor by virtue of the fact that information is in the packet could be measured.

25 When Node A transmits a data packet to Node B, the Network Monitor receives the packet. The RTP recognizes the packet as being part of a timed transaction and passes control to the ETM. The ETM recognizes it as a start time event, stores the start time in the event
30 timing database and returns control to the RTP after checking for aging.

When Node B receives the data packet from Node A, it sends back an acknowledgment packet. When the Network Monitor sees that packet, it delivers the event to the
35 ETM, which recognizes it as an end time event. The ETM

- 82 -

calculates the delay time for the complete transaction and uses that to update the average transaction time. The ETM then compares the new average transaction time with the threshold for this event. If it has been
5 exceeded, the ETM issues an alarm to the Workstation.

Note that this example is measuring something very different than the previous example. The first example measures the time it takes to traverse the network, perform an action and return that result to the
10 requesting node. It measures performance as seen by the user and it includes delay times from the network as well as delay times from the File Server.

The second example is measuring network delays without looking at the service delays. That is, the ETM
15 is measuring the amount of time it takes to send a packet to a node and receive the acknowledgement of the receipt of the message. In this example, the ETM is measuring transmissions delays as well as processing delays associated with network traffic, but not anything having
20 to do with non-network processing.

As can be seen from the above examples, the ETM can measure a broad range of events. Each of these events can be measured passively and without the cooperation of the nodes that are actually participating
25 in the transmission.

The Address Tracker Module (ATM)

Address tracker module (ATM) 43, one of the software modules in the Network Monitor (see Fig. 5), operates on networks on which the node addresses for
30 particular node to node connections are assigned dynamically. An Appletalk® Network, developed by Apple Computer Company, is an example of a network which uses dynamic node addressing. In such networks, the dynamic change in the address of a particular service causes
35 difficulty troubleshooting the network because the

- 83 -

network manager may not know where the various nodes are and what they are called. In addition, foreign network addresses (e.g., the IP addresses used by that node for communication over an IP network to which it is
5 connected) can not be relied upon to point to a particular node. ATM 43 solves this problem by passively monitoring the network traffic and collecting a table showing the node address to node name mappings.

In the following description, the network on which
10 the Monitor is located is assumed to be an Appletalk® Network. Thus, as background for the following discussion, the manner in which the dynamic node addressing mechanism operates on that network will first be described.

15 When a node is activated on the Appletalk® Network, it establishes its own node address in accordance with protocol referred to as the Local Link Access Protocol (LLAP). That is, the node guesses its own node address and then verifies that no other node on
20 the network is using that address. The node verifies the uniqueness of its guess by sending an LLAP Enquiry control packet informing all other nodes on the network that it is going to assign itself a particular address unless another node responds that the address has already
25 been assigned. If no other node claims that address as its own by sending an LLAP acknowledgment control packet, the first node uses the address which it has selected. If another node claims the address as its own, the first node tries another address. This continues until, the
30 node finds an unused address.

When the first node wants to communicate with a second node, it must determine the dynamically assigned node address of the second node. It does this in accordance with another protocol referred to as the Name
35 Binding Protocol (NBP). The Name Binding Protocol is

- 84 -

used to map or bind human understandable node names with machine understandable node addresses. The NBP allows nodes to dynamically translate a string of characters (i.e., a node name) into a node address. The node
5 needing to communicate with another node broadcasts an NBP Lookup packet containing the name for which a node address is being requested. The node having the name being requested responds with its address and returns a
10 Lookup Reply packet containing its address to the original requesting node. The first node then uses that address its current communications with the second node.

Referring to Fig. 36, the network includes an Appletalk® Network segment 702 and a TCP/IP segment 704, each of which are connected to a larger network 706
15 through their respective gateways 708. A Monitor 710, including a Real Time Parser (RTP) 712 and an Address Tracking Module (ATM) 714, is located on Appletalk network segment 702 along with other nodes 711. A
20 Management Workstation 716 is located on segment 704. It is assumed that Monitor 710 has the features and capabilities previously described; therefore, those features not specifically related to the dynamic node addressing capability will not be repeated here but rather the reader is referred to the earlier discussion.
25 Suffice it to say that Monitor 710 is, of course, adapted to operate on Appletalk Network segment 702, to parse and analyze the packets which are transmitted over that segment according to the Appletalk® family of protocols and to communicate the information which it extracts from
30 the network to Management Workstation 716 located on segment 704.

Within Monitor 710, ATM 714 maintains a name table data structure 730 such as is shown in Fig. 37. Name Table 720 includes records 722, each of which has a node
35 name field 724, a node address field 726, an IP address

- 85 -

field 728, and a time field 729. ATM 714 uses Name Table 720 to keep track of the mappings of node names to node address and to IP address. The relevance of each of the fields of records 722 in Name Table 720 are explained in 5 the following description of how ATM 714 operates.

In general, Monitor 710 operates as previously described. That is, it passively monitors all packet traffic over segment 702 and sends all packets to RTP 712 for parsing. When RTP 712 recognizes an Appletalk 10 packet, it transfers control to ATM 714 which analyzes the packet for the presence of address mapping information.

The operation of ATM 714 is shown in greater detail in the flow diagram of Fig. 38. When ATM 714 15 receives control from RTP 712, it takes the packet (step 730 and strips off the lower layers of the protocol until it determines whether there is a Name Binding Protocol message inside the packet (step 732). If it is a NBP message, ATM 714 then determines whether it is new name 20 Lookup message (step 734). If it is a new name Lookup message, ATM 714 extracts the name from the message (i.e., the name for which a node address is being requested) and adds the name to the node name field 724 of a record 722 in Name Table 720 (step 736).

25 If the message is an NBP message but it is not a Lookup message, ATM 714 determines whether it is a Lookup Reply (step 738). If it is a Lookup Reply, signifying that it contains a node name/node address binding, ATM 714 extracts the name and the assigned node address from 30 the message and adds this information to Name Table 720. ATM 714 does this by searching the name fields of records 722 in Name Table 720 until it locates the name. Then, it updates the node address field of the identified record to contain the node address which was extracted 35 from the received NBP packet. ATM 714 also updates time

- 86 -

field 729 to record the time at which the message was processed.

After ATM 714 has updated the address field of the appropriate record, it determines whether any records 722
5 in Name Table 720 should be aged out (step 742). ATM 714 compares the current time to the times recorded in the time fields. If the elapsed time is greater than a preselected time period (e.g. 48 hours), ATM 714 clears the record of all information (step 744). After that, it
10 awaits the next packet from RTP 712.

As ATM 714 is processing each a packet and it determines either that it does not contain an NBP message (step 732) or it does not contain a Lookup Reply message (step 738), ATM 714 branches to step 742 to perform the
15 age out check before going on to the next packet from RTP 712.

The Appletalk to IP gateways provide services that allow an Appletalk Node to dynamically connect to an IP address for communicating with IP nodes. This service
20 extends the dynamic node address mechanism to the IP world for all Appletalk nodes. While the flexibility provided is helpful to the users, the network manager is faced with the problem of not knowing which Appletalk Nodes are currently using a particular IP address and
25 thus, they can not easily track down problems created by the particular node.

ATM 714 can use passive monitoring of the IP address assignment mechanisms to provide the network manager a Name-to-IP address mapping.

30 If ATM 714 is also keeping IP address information, it implements the additional steps shown in Fig. 39 after completing the node name to node address mapping steps. ATM 714 again checks whether it is an NBP message (step 748). If it is an NBP message, ATM 714 checks whether it
35 is a response to an IP address request (step 750). IP

- 87 -

address requests are typically implied by an NBP Lookup request for an IP gateway. The gateway responds by supplying the gateway address as well as an IP address that is assigned to the requesting node. If the NBP
5 message is an IP address response, ATM 714 looks up the requesting node in Name Table 720 (step 752) and stores the IP address assignment in the IP address field of the appropriate record 722 (step 754).

After storing the IP address assignment
10 information, ATM 714 locates all other records 722 in Name Table 720 which contain that IP address. Since the IP address has been assigned to a new node name, those old entries are no longer valid and must be eliminated. Therefore, ATM 714 purges the IP address fields of those
15 records (step 756). After doing this cleanup step, ATM 714 returns control to RTP 712.

Other embodiments are within the following claims. For example, the Network Monitor can be adapted to identify node types by analyzing the type of packet
20 traffic to or from the node. If the node being monitored is receiving mount requests, the Monitor would report that the node is behaving like node a file server. If the node is issuing routing requests, the Monitor would report that the node is behaving like a router. In
25 either case, the network manager can check a table of what nodes are permitted to provide what functions to determine whether the node is authorized to function as either a file server or a router, and if not, can take appropriate action to correct the problem.

- 88 -

APPENDIX I

SNMP MIB Subset Supported

This is the subset of the standard MIB which can be obtained by monitoring.

Refer to RFC 1066 Management Information Base for an explanation on the items which follow.

System group:
none

Interfaces group
ifType
ifPhysAddress
ifOperStatus
ifInOctets
ifInUcastPkts
ifInNUcastPkts
ifOutOctets
ifOutUcastPkts
ifOutNUcastPkts

Address Translation group
none

IP group
ipForwarding
ipDefaultTTL
ipInReceives
ipInHdrErrors
ipInAddrErrors
ipForwDatagrams
ipReasmReqds
ipFragCreates

IP Address Table
ipAddress
ipAdEntBcastAddr

IP Routing Table
none

ICMP group
icmpInMsgs
icmpInErrors
icmpInDestUnreachs
icmpInTimeExcds
icmpInParmProbs
icmpInSrcQuenchs
icmpInRedirects
icmpInEchoes

App. I - 1

HELP

WIRED NEWS

HOTWIRED

WIRED MAGAZINE

SUCK.COM

**TurboTax**
CLICK HERElook for

SEARCH

Before: April 1, 1995

NEW SEARCH

REVISE SEARCH

Click here
amazon.comSM

[Click here for Amazon.com](#)

Returned: 226 matches.

Breakdown: internet: 11672334, phone: 6326171 ◀ 71 - 80 ▶

[The Web](#)[Usenet](#)[Top News Sites](#)[Businesses](#)[People](#)[Email Addresses](#)[Classifieds](#)[Domain Names](#)[Stocks](#)[Discussion Groups](#)[ShareWare](#)

- *Buy Computer Books, 20-40% off at [BarnesandNoble.com!](#)*
- *Need reliable sources? Use the [HotBot Research Service](#).*

71. [C&I 514 Web](#)

91% Language Arts Web Back to Student Web Welcome to C&I 514 Language Arts Web Mexican Culture Server It has Mexican recipes, songs, flags, and artists. All is in Spanish. Margaret Bruha Foreign Languages and Culture Web references The site contains a...

<http://www.cae.wisc.edu/~tojek/514language.html>, 5567 bytes, 08Mar95

72. [Jo's Homepage](#)

91% Jo's Bookmarks These are some of the resources that a typical K12 teacher found on the Internet: White House Tour Index - NMAA/WHC National Museum of American Art home page Adrien Rothschild - NMAA/WHC







<http://www.nmaa.si.edu/whc/whcgifs...>
<http://www.vt.com/edu/jobook.html>, 13996 bytes, 20Jun94

73. [Electronic Retail](#)

91% From : David Rich, sasha@ozemail.com.au On : Thu Mar 2 06:54:37 EST 1995 In reponse to why consumers may be going off buying over the internet. Most consumers are not yet on the net. Most people do not understand what the net is, nor how it works..

<http://cism.bus.utexas.edu/issues/issue2/comment11.html>, 2997 bytes, 05Mar95

74. [March Archives: Re: 28.8 not all that bad](#)

- 91% Re: 28.8 not all that bad David Winet (dwinet@qal.berkeley.edu) Tue, 14 Mar 1995 06:40:56 -0800 (PST) Messages sorted by: [date][thread][subject][author] Next message: Roie Gat: "Re: 28.8 not all that bad" Previous message: Larry Chace: ...
http://www.indstate.edu/CU-SeeMe/devl_archives/mar_95/0259.html, 3587 bytes, 02Apr95
75.  [The Web Untangled](#)
- 91% Introduction Known as WWW or just the Web, the World Wide Web is more a concept than a specific protocol. The Web provides computer users on the Internet with a consistent means of access to a variety of media in a simplified fashion. Access is via.
<http://wvnxaxa.wvnet.edu/~roman/UNTANGLED.html>, 3220 bytes, 06Jan95
76.  [March Archives: Re: 28.8 not all that bad](#)
- 90% Re: 28.8 not all that bad rrvindr@INDYVAX.IUPUI.EDU Tue, 14 Mar 1995 15:13:30 -0500 Messages sorted by: [date][thread][subject][author] Next message: Jesus Arango: "cu-seeme & iphone conference" Previous message: Dan Berrios: "Timing out."
http://www.indstate.edu/CU-SeeMe/devl_archives/mar_95/0270.html, 4071 bytes, 02Apr95
77.  [Hendry's first home page](#)
- 90% Hendry's Homepage Hot ! Add you page here! This page is best viewed with Netscape Navigator. Download Let's Surf Cool Sites Games HK Stuffs Resources Dictionary Museums Movies, TV Softwares, Icons Download Autos News, Magazines, Computer paper What.
<http://www.smart.is/link4/LINK5.HTM>, 16131 bytes, 01Jan94
78.  [NIC-News Newsletters: NIC Newsletter v 5.10](#)
- 90% NIC Newsletter v 5.10 Sheryl Erez (erez@cac.washington.edu) Tue, 23 Mar 1993 15:32:46 -0800 (PST) == || Networks & Distributed Computing || NIC - NEWS || == -- | An update on network resources for NDC staff || Volume 5, Issue 10 Edited by Sheryl...
<http://www.washington.edu/nic-news/old/text/0053.html>, 7486 bytes, 06Jan95
79.  [\(http://www.cox.smu.edu/class/mis6386/people/stort/iphone.html\)](http://www.cox.smu.edu/class/mis6386/people/stort/iphone.html)
- 90% The Internet Phone The IPHONE is a cool program to "telephone" worldwide using the Internet. System Requirements: 386 CPU or higher Windows 3.1 or later A Sound Blaster (compatible) sound card Speakers Microphone Internet Access (SLIP, e.g. PICnet..)
<http://www.cox.smu.edu/class/mis6386/people/stort/iphone.html>, 1841 bytes, 30Mar95
80.  [Spock's Other Raging Slab of Meat](#)
- 90% The PolySpock Project BBS PolySpock is the direct and spiritual successor to Rathead Systems. PSP was started on a crazy whim one day by RatSnatcher and his roommate from Ohio, Special Ed (aka Gar, Adam Capell) while they were trying to figure out..
<http://www.arlington.com/~tjames/pig/pspock2.html>, 4992 bytes, 15Mar95

Breakdown: internet: 11672334, phone: 6326171 ◀ 71 - 80 ▶

Netscape Conference and CoolTalk Meeting Room

Sponsored by:



Is your domain available?

- Home
- Register
- Unregister
- Talk Meeting Rm
- Chat Rooms
- Get Software
- Errors / FAQ

• [Home School chat](#)

• It's a **BOY!**



• [Holocaust Remembrance Day Chat](#)

  **Search Engines...**

Member of the Internet Link Exchange

Welcome to the q5 Netscape Conference and CoolTalk® Meeting Room!
now with CHAT!

(We have **NO** connection to Netscape Communications Corporation.)



Bookmark **this** page. Other pages will change as this site improves!

We have *Chat* and *Talk* meeting rooms.

Chat allows you to communicate either publicly or privately by typing to each other.

For *talk* you can use either CoolTalk or Netscape Conference.

First you register (it's free), then go to the Talk Meeting Room to see the list of people waiting for a call.

To call someone, press the *CoolTalk* () or *Conference* () icon next to their name.

For most of us, every time we log on to your Internet Service Provider you have a different IP address, so you must register here each time you log on to your Internet Service Provider.



A random user

See the people you talk to! Email us your GIF, JPEG, or BMP with a bio. No graphic? Snail mail us your picture and a bio and we will scan it in.



Netscape Conference is a program in the suite *Netscape Communicator Preview Release*. You must choose to download:

"Netscape Communicator Preview Release - All Components plus Plug-ins"
in order to get *Netscape Conference*.

There is a section of the Release notes which deals with Netscape Conference.



CoolTalk® is a plugin program for Netscape. It comes bundled with Netscape 3.0 and is available for many platforms including Windows 95, Windows NT, Windows 3.1, MacOS, SunOS, Solaris, HP-UX, Digital Unix, and IRIX.

If you are downloading Netscape, you must get the "plugin" version with a "P" in its name. When you select the "Desired Product:" make sure you select one which is called "standard plus components"

CoolTalk can also be downloaded from the Netscape FTP site.

MAC users *must* use a 28.8 or faster modem.

Check out the CoolTalk FAQ. It specifically addresses issues of compatible sound cards, upgrading to *Full Duplex*, and *MAC*.

We have NO connection with Netscape. While we may be able to answer simple questions, Netscape Technical Support is more knowledgeable than us. **Send comments about this site to coolmaster@detel.com**



This site may be uncomprehensible without
Netscape Navigator 3.0. [Download Netscape Now](#)



Copyright ©1996 Detel Communications
<http://www.detel.com/>
This site was created by:
Edward J. Weinberg webmaster@detel.com

Detel, Inc.
Suite 332
2490 Black Rock Tnpk
Fairfield, CT 06432

We can create one for you!

No programmers were harmed in the testing of this product.

Is your domain available?
Check out:
www.serverking.com

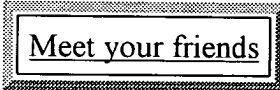
- Home
- Register
- Unregister
- Talk Meeting Rm
- Chat Rooms
- Get Software
- Errors / FAQ

Talk Meeting Room

The list



Member of the Internet Link Exchange





Visitors:



The following people have indicated they are looking for someone to talk to using Netscape Conference or Cooltalk.







You have to press the "reload" button to see the latest list of names.

If you are waiting for a call...make sure Conference and/or Cooltalk is turned on!











To call someone, press the *CoolTalk* () or *Conference* () icon next to their name.

When people disconnect, they are not removed from this list, and if you disconnect and log in again you will probably have a different IP address.

While waiting for a call try our new [Chat Room](#)

Time/Date	Dial	Name/Email	City	St	Country
03/20 13:55 EST		Guenter Z.	Tacoma	WA	USA
		Kai, ruf nochmal durch!!!!!!!!!!!!			
03/20 13:43 EST	 	LastNiceGuy	houston	tx	USA
		I'd like to try this anyone call me.			
03/20 13:35 EST	 	katherina		bc	canada
		waiting for last nice guy			
03/20 13:31 EST		Erin	York	PA	USA

03/20 13:20 EST		<u>ROBNIEDSME</u>	Tamarac	FL	USA
		USA			
03/20 13:11 EST		<u>M. ANN</u>		md	usa
		financial services,loans, maryland only.			
03/20 12:37 EST		<u>Eglin M</u>	Port Elizabeth		South Africa
		Let's talk			
03/20 12:27 EST		<u>Rik</u>	vitoria	ES	Brasil
		USA			
03/20 12:26 EST		<u>Maximo</u>	Roma		Italia
		italian			
03/20 12:04 EST		<u>peter</u>	montreal	que	Canada
03/20 12:01 EST		<u>Kai Schütrumpf</u>	Friedewald	Hessen	Germany
		Please only for Guenter!!!			
03/20 12:01 EST		<u>ROCAL</u>	Rochester	IN	USA
		KA9YQM			
03/20 11:03 EST		<u>tinkoo</u>	MUMBAI	MH	INDIA
		i can receive calls but can't call up...			
03/20 11:01 EST		<u>Armando Gonzalez</u>	Caracas	DF	Venezuela
		USA			
03/20 10:58 EST		<u>ARCH-Snd Questions</u>	will snd answer	Fl	U.S.A.
		NetConf & CT works send e-mail for info			
03/20 10:50 EST		<u>perfect_</u>			
		hi			
03/20 10:45 EST		<u>Zaxxon</u>	statesville	NC	USA
03/20 10:35 EST		<u>bob</u>	ellenton	fl	usa
		USA			
03/20 10:09 EST		<u>Don</u>	Morristown	NJ	
		My E-Mail is: Don0453@AOL.COM			
03/20 09:22 EST		<u>hema</u>	mumbai		india
		just testing			

03/20 09:14 EST		Daniel	vic		aust
USA					
03/20 09:12 EST		Corinne			Singapore
Looking for DANIEL dear....					
03/20 07:05 EST		TAZZ	Newnan	GA	USA
USA					
03/20 06:52 EST		LK	Russellville	AR	USA
USA					
03/20 06:38 EST		cjay	melbourne	vic	australia
waiting for mates					
03/20 06:26 EST		Tbone	okinawa	ap	japan
03/20 04:44 EST		Aleksander	San Fran	CA	ICQ#8336177
Hey any nasty ladies out there?...M25					
03/20 03:29 EST		KRISHY			Malaysia
USA					
03/20 02:22 EST		Lacie	Jacksonville	FL	USA
Waiting for Dave					
03/20 02:22 EST		Davester	Phoenix	AZ	US
Lacie.....					

[Home](#) [Add your IP](#)

Copyright ©1996 Detel Communications

<http://www.detel.com/>

This site was created by:

Edward J. Weinberg coolmaster@detel.com

Detel, Inc.

We can create one for you!

No programmers were harmed in the testing of this product.



[LinkExchange Member](#)

Register your IP address "telephone number" for Netscape Conference and/or CoolTalk

This will add your *current* IP address and other information to the contact list. Since your IP address will change each time you log on, you must register each time you log on.

The following information is optional, but including it will improve the chances that someone will talk to you.

Including your email address will put you on our mailing lists for updates to this site.

Register me for:

-  CoolTalk
  Netscape Conference

Name or Nick:
E-Mail:
City:
State:
Country:
Comment:

Remember this information so I do not have to enter it again!

*

Your IP address is: 151.200.96.2.

(FILE 'USPAT' ENTERED AT 15:36:24 ON 18 MAR 1998)

L1 219 S IP ADDRESS
L2 145 S L1 AND SERVER
L3 103 S L2 AND INTERNET
L4 6 S L3 AND TELEPHONY
L5 0 S (INTERNET AND (PHONE OR TELEPHONE OR TELEPHONY))/TI
L6 0 S (INTERNET AND (PHONE OR TELEPHONE OR TELEPHONY))/ABS
L7 12 S (INTERNET AND (PHONE OR TELEPHONE OR TELEPHONY))/AB
L8 32 S "IP ADDRESS" (3A) (SERVER OR DATABASE OR STORAGE OR COLLECT
ECT
L9 2 S ("IP ADDRESS" (5A) (SERVER OR DATABASE OR STORAGE OR COLLECT
L10 1 S L9 AND (TELEPHONY OR VOICE OR MULTIMEDIA OR SOUND OR TALK O

(FILE 'USPAT' ENTERED AT 18:26:56 ON 18 MAR 1998)

L1 1369 S IRC
L2 1 S "DYNAMIC IP ADDRESS" AND (SERVER OR DATABASE OR STORAGE)
L3 3 CHAT AND "IP ADDRESS"

(FILE 'USPAT' ENTERED AT 15:07:40 ON 23 MAR 1998)

L1 88 S 395*12/CCLS
L2 0 S L1 AND INTERNET
L3 19 S L1 AND TELEP?
L4 2 L1 AND "TELEPHONE INTERFACE"

(FILE 'USPAT' ENTERED AT 11:49:05 ON 24 MAR 1998)

L1 0 S "INTERNET TELEPHONY"
L2 7 S "NETWORK TELEPHONY"
L3 1037 S INTERNET
L4 511 S L3 AND TELEPHONE
L5 51 S L4 AND INTERNET (4A) TELEPHONE
L6 3 S L4 AND (INTERNET (4A) TELEPHONE)/AB
L7 2 L5 AND (INTERNET (W) TELEPHONE)

(FILE 'USPAT' ENTERED AT 14:12:11 ON 27 MAR 1998)

L1 103486 S INTERNET OR WEB OR WWW
L2 3531 S L1 AND (TELEPHON! OR VOICE OR TALK OR SPEAK OR SPEECH OR AU
L3 352 S L2 AND ((ROLLING OR NEW OR DYNAMIC OR UNIQUE OR DIFFERENT
L4 3613 S L1 AND (TELEPHON! OR PHONE OR VOICE OR TALK OR SPEAK OR SPE
L5 359 S L4 AND ((ROLLING OR NEW OR DYNAMIC OR UNIQUE OR DIFFERENT
L6 178 S L5 AND ((USER (W) INTERFACE) OR WINDOW)
L7 35 S L6 AND (IP (7A) (SERVER OR DATABASE OR "DATA BASE"))

Readme
VocalTec Internet Phone (TM)
Version 2.5 (Build 5) - February, 1995
=====

Copyright(c) 1995 by VocalTec Ltd.

In order to use Internet Phone, you need...

1. Windows 3.1 or higher (not NT).
2. 8MB of RAM recommended.
3. 486SX 25Mhz or faster recommended.
4. Windows compatible audio board, with speaker and microphone.
5. TCP/IP software with WINSOCK 1.1 support.
6. A SLIP/PPP, or direct Internet connection (14,400 baud minimum).

Installing the Internet Phone

1. Make sure that your microphone and speaker work properly, by recording yourself using the Microsoft's Sound Recorder. See "Preparing Your Audio Device" section, below.
2. Create a directory on your hard-disk
e.g.: MD C:\IPHONE
3. From that directory, execute the self-extracting archive
C:\IPHONE> IPHONE25.EXE
4. Choose File/Run and execute C:\IPHONE\ADDICONS
A new program-manager group file will be created, with icons for The
e Internet Phone and Help file
5. Double-click the Internet Phone icon.
6. The first time you run the Internet Phone Software, a Quick Tour wi
ll be suggested to you.

Readme

7. Start talking with the rest of the world!

About The Internet Phone and IRC -----

The Internet Phone uses the IRC to show the currently on-line users. The actual talk is done directly between the PC's running the Internet Phone, and NOT via the IRC.

Selecting an IRC server -----

In order to use the Internet Phone, you must be connected to an IRC server. It is best to select the one nearest to you, in order to get the best connection.

Once you're connected to the IRC, you can call any other Internet Phone user that is connected to the IRC network. There is no need for both of you to use the same IRC server.

Please note that by "nearest" we mean over the net, but usually geographically close places have a better network connection.

The first time you connect, you can select a server from the Publicly Accessible Servers. It might not be the closest to you, but it will enable you to start talking.

Later you can try and find a server better suited for you. Note that many servers accept connections from specific areas. Some are limited to a country, some to a specific campus.

Readme

Technical Support

Before calling VocalTec for technical support, please do the following
:

1. Check the Troubleshooting from the Help menu in the Internet Phone.
It contains a list of problems and solutions.
2. Select "Technical Support" from the Help menu for information on ho
w
to contact VocalTec.


END of README.TXT


INTERNET PHONE HELP

File Edit Bookmark Options Help

Contents Search Back Print

Internet Phone Help Index


 **Quick Tour**
A quick tour of Internet Phone for newcomers.

 **Using Internet Phone**

INTERNET PHONE HELP

File Edit Bookmark Options Help

Contents Search Back Print

 **Using Internet Phone**

Basics

- [What is Internet Phone?](#)
- [Starting Internet Phone](#)
- [Quitting Internet Phone](#)

The IRC network

- [Connecting to an IRC server](#)
- [Disconnecting from the IRC](#)
- [Joining and creating topics](#)
- [Leaving a topic](#)
- [Using private topics](#)

Calling and talking over the Internet

- [Making a call to a person](#)

2 Internet Phone

What is Internet Phone?

Internet Phone is a unique software product that opens a new and exciting dimension for Internet users. Up to now you could only use the Internet to transfer text and graphics. You could send e-mail to other users, or even have text "chats" with them, but didn't you ever wish you could actually speak with them?

With Internet Phone you can use the Internet to speak with your own voice with Internet users all over the globe! Yes, real-time real-voice conversations over the Internet!

All you need is Internet Phone, an Internet connection and a Windows-compatible audio device. Plug in a microphone and speaker, run Internet Phone and, by clicking a button, get in touch with Internet users all over the world at the price of a local phone call. Whether you want to meet new friends, get information personally, or make the direct business contact, Internet Phone is for you. A friendly graphic user interface and a smart Voice-Activation feature make conversation a snap. VocalTec's sophisticated voice compression and transfer technology makes sure your voice gets across in a flash and using only a fraction of the bandwidth.

Internet Phone uses the IRC (Internet Relay Chat) as a global phonebook, always presenting you with an updated list of topics and on-line users. By connecting to an IRC server you can access this list and call anyone on it. You can create new topics or list your name under existing ones, waiting for people with shared interests to join in. When you get to know a person, you can put his or her name under a Quick-Dial button. To call the person, simply click on her Quick-Dial button and a call to her is made, even if you are both listed under different topics. Once you establish contact with a person, communication is carried out directly over the Internet, and not through the IRC.

You and others can use private topics to block out unwanted callers, when you are waiting for important business or intimate calls. Private topics also let the members of world-wide organizations or groups easily locate each other without interference.

Of course, the quality of voice communication depends on the quality of your Internet connection. We recommend that you use Internet Phone on systems with an Internet connection of 14400 baud or more. Installing a voice compression card, such as VocalTec's VC Card, can also increase the efficiency and quality of communication considerably.

Untitled

Domain Name: Q5.COM

Administrative Contact, Technical Contact, Zone Contact:

Weinberg, Ed (EW286) edw@DETEL.COM
203-333-3675

Billing Contact:

Weinberg, Ed (EW286) edw@DETEL.COM
203-333-3675

Record last updated on 19-Mar-98.

Record created on 22-Feb-96.

Database last updated on 19-Mar-98 04:11:09 EST.

Domain servers in listed order:

NS1.DETEL.COM	165.254.238.145
NS1.GRANITECANYON.COM	205.166.226.34
NS2.GRANITECANYON.COM	208.146.254.90
NS1.RESEARCH.TROY.NY.US	206.72.196.240

Enter a handle, name, mailbox, or other field, optionally preceded by a keyword, like "host diis". Type "?" for short, 2-page details, "HELP" for full documentation, or hit RETURN to exit.

---> Do ^E to show search progress, ^G to abort a search or output <--

Whois:



The Leader in Internet Discussion



[Search](#)



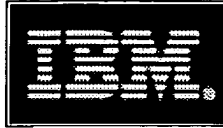
[Post Message](#)



[My Deja News](#)



[Help](#)



Writing code?

[Click here to find the resources you need to code faster and reach more customers.](#)

Article 5 of exactly 5

[<<](#) [>>](#) [^](#)
[Previous Article](#) [Next Article](#) [Current Results](#)

- [Help](#)
- [Post New](#)
- [Bookmark](#)
- [Author Profile](#)
- [Post Reply](#)
- [Text Only](#)
- [View Thread](#)
- [Email Reply](#)

Subject: Re: Getting IP address of PPP-connected Mac
From: jgull@umich.edu (Jason Gull)
Date: 1995/04/03
Message-ID: <jgull-0304951005350001@pm012-11.dialip.mich.net>
Newsgroups: [comp.sys.mac.comm](#)

[\[Subscribe to \[comp.sys.mac.comm\]\(#\)\]](#)

[\[More Headers\]](#)

Thanks for the advice. However, I'm already using MacTCPWatcher to find out *my own* IP address. It's trying to discover the IP addresses of other PPP users that is troubling me. So far the best "solution" seems to be a central location where I and my friends with whom I may wish to use Talk, NetPhone, etc. in the future can post our dynamic IPs each time we connect via PPP. Then other users can check that location and contact me.

I'm working on an AppleScript to do this. Any info, comments, advice would be appreciated. I'll post details here (and to my web page) if I ever get it going.

Jason Gull
jgull@umich.edu
<http://www.umich.edu/~jgull/>

In article <[3lmrnv\\$igu@sct1.sct.fr](mailto:3lmrnv$igu@sct1.sct.fr)>, Luc Saint-Elie
 <lstelie@world-net.sct.fr> wrote:

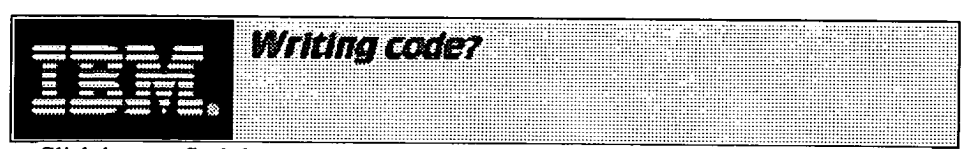
```

> jgull@umich.edu (Jason Gull) wrote:
> >Is there *any* other way I can find out the IP address of a Mac connected
> >via MacPPP without asking the person using the machine on the other end?
> >I've tried making Talk requests to my friend's various email addresses, to
> >no avail. It just seems like the server has to have some way of figuring
> >out the address of a MacPPP-connected machine. Right? So is there any
> >way I can tap into that knowledge on the server?
>
> There are two ways (may be much more) to know that:
>
> 1- Simple way, use the "stats.." item in your ConfigPPP panel. On the
bottom right
> side of the display you will find your IP address.
  
```

> 2- Best way : Use MacTCP Watcher (freeware from Peter Lewis), a really nice
> software allowing you to know EVREYTHING about your IP connection.
>
> Hope this helps

--
Jason Gull
jgull@umich.edu

Liberty is always dangerous. But it is
the safest thing we have. - H.E.Fosdick



[Click here to find the resources you need to code faster and reach more customers.](#)

<< >> ^
[Previous](#) [Next](#) [Current](#)
[Article](#) [Article](#) [Results](#)

- [Help](#)
- [Post New](#)
- [Bookmark](#)
- [Author Profile](#)
- [Post Reply](#)
- [Text Only](#)
- [View Thread](#)
- [Email Reply](#)

[IBM Technical Interchange '98](#) | [Directories](#) | [Classifieds](#) | [Yellow Pages](#)

[About Deja News](#) · [New Users](#) · [Ad Info](#) · [Our Advertisers](#) · [How are we doing?](#)
[Home](#) · [Search](#) · [Post](#) · [My Deja News](#) · [Help](#)

Copyright © 1995-98 [Deja News, Inc.](#) All rights reserved. [Conditions of use.](#)



The Leader in Internet Discussion



[Search](#)



[Post Message](#)

[my](#)

[My Deja News](#)



[Help](#)



[Click here for My Deja News.](#)

Article 8 of exactly 11

[<<](#) [>>](#) [^](#)
[Previous Article](#) [Next Article](#) [Current Results](#)

- [Help](#)
- [Post New](#)
- [Author Profile](#)
- [Post Reply](#)
- [View Thread](#)
- [Email Reply](#)
- [Bookmark](#)
- [Text Only](#)

Subject: Re: Internet Phone for Mac?
From: jgull@umich.edu (Jason Gull)
Date: 1995/04/17
Message-ID: <jgull-1704950116450001@pm049-28.dialip.mich.net>
Newsgroups: [comp.sys.mac.comm](#)

[\[Subscribe to comp.sys.mac.comm\]](#)

[\[More Headers\]](#)

It's called **NetPhone**, and from all accounts, it's a lot better than Internet Phone (which uses an IRC **server**). **NetPhone** supports GSM (the compression scheme used by a lot of European and other cellular phones), which means it works fine over a 14.4 line, though GSM really requires a 25mhz 040 minimum.

The only problem for dial-up SLIP/PPP users is that to call, a caller needs to know the **IP address** of the receiver's machine, which changes all the time with most SLIP/PPP accounts. I've heard Internet Phone is trying to solve this using a dedicated IRC **server**. I've been trying to solve it with a script to write my current dial-up **address** to my web page, but it doesn't really work yet.

NetPhone is from emagic, and their web site is at <http://www.emagic.com>
 There you can download a demo version (outgoing calls limited to 90 seconds).

Jason Gull
jgull@umich.edu

In article <jazzbo-1604951234280001@onramp2-11.onr.com>, jazzbo@onr.com wrote:

> The latest issue of Wired had a blurb that said there was something akin
 > to the Internet Phone available for Mac users. What is it and where can I
 > get it??

>
 >

. -Dave

> P.S. Have a nice day.
 > --

> Have you ever gotten sick of hearing AT&T take credit for things that they didn't invent? You will.

>
>

-Dave Hamilton (jazzbo@onr.com)



[Click here for My Deja News.](#)

[<<](#) [>>](#) [^](#)
[Previous Article](#) [Next Article](#) [Current Results](#)

- [Help](#)
- [Author Profile](#)
- [View Thread](#)
- [Post New](#)
- [Post Reply](#)
- [Email Reply](#)
- [Bookmark](#)
- [Text Only](#)

[IBM Technical Interchange '98](#) | [Directories](#) | [Classifieds](#) | [Yellow Pages](#)

[About Deja News](#) · [New Users](#) · [Ad Info](#) · [Our Advertisers](#) · [How are we doing?](#)
[Home](#) · [Search](#) · [Post](#) · [My Deja News](#) · [Help](#)

Copyright © 1995-98 [Deja News, Inc.](#) All rights reserved. [Conditions of use.](#)



The Leader in Internet Discussion



[Search](#)



[Post Message](#)



[My Deja News](#)



[Help](#)

Introducing an easier way to read Usenet...

[Click here for My Deja News.](#)

Power Search Results

11 Matches for search:

netphone ip address server

- [Help](#)
- [Quick Search](#)
- [Interest Finder](#)
- [Browse Groups](#)

	<u>Date</u>	<u>Scr</u>	<u>Subject</u>	<u>Newsgroup</u>	<u>Author</u>
1.	95/08/20	047	server addressed PPP account	comp.protocols.tcp-ip	Emilio C. Petri
2.	95/08/20	047	server addressed PPP account	comp.protocols.tcp-ip	Emilio C. Petri
3.	95/08/20	047	server addressed PPP account	comp.protocols.ppp	Emilio C. Petri
4.	95/08/20	047	server addressed PPP account	comp.sys.mac.comm	Emilio C. Petri
5.	95/08/21	046	Re: server addressed PPP acc	comp.sys.mac.comm	Clark Martin
6.	95/08/24	045	Re: Static vs. Dynamic IP ad	comp.sys.mac.comm	Dennis Wall
7.	95/06/28	043	PPP, Dynamic Addressing, & N	comp.sys.mac.comm	Jason Gull
8.	95/04/17	041	Re: Internet Phone for Mac?	comp.sys.mac.comm	Jason Gull
9.	95/04/03	041	Re: Getting IP address of PP	comp.sys.mac.comm	Jason Gull
10.	95/03/30	041	Re: Getting IP address of PP	comp.sys.mac.comm	Hao-lin Harry T
11.	95/07/10	034	Re: Eudora and Mailshare	comp.sys.mac.comm	T. Byfield

Individual word match counts (exact)

- address: 132081
- ip: 36509
- netphone: 225
- server: 164858

Introducing an easier way to read Usenet...

[Click here for My Deja News.](#)

Search Again: Power Search

Search for:

netphone ip address server

Example: ufo AND (sighting OR abduction OR alien)

- [Help](#)
- [Quick Search](#)
- [Interest Finder](#)
- [Browse Groups](#)

Archive: ▼

Keywords matched: ▼

Number of matches: ▼

Results format: ▼

Sorted by: ▼

Group(s):

Example: alt.tv.x-files or *x-files*

Author(s):

Example: demos@dejanews.com

Subject(s):

Example: FAQ or (Frequently Asked Questions)

Date from: To:

Example format: Apr 1 1997

[IBM Technical Interchange '98](#) | [Directories](#) | [Classifieds](#) | [Yellow Pages](#)

[About Deja News](#) · [New Users](#) · [Ad Info](#) · [Our Advertisers](#) · [How are we doing?](#)
[Home](#) · [Search](#) · [Post](#) · [My Deja News](#) · [Help](#)

Copyright © 1995-98 [Deja News, Inc.](#) All rights reserved. [Conditions of use.](#)



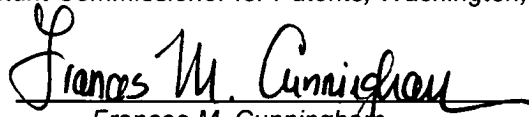
ATTORNEY'S DOCKET NO.:N0003/7002

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICANT: Shane D. Mattaway et al.
SERIAL NO.: 08/721,316
FILED: September 25, 1995
FOR: GRAPHIC USER INTERFACE FOR INTERNET TELEPHONY APPLICATION
EXAMINER: --
ART UNIT: --

CERTIFICATE OF MAILING UNDER 37-C.F.R. §1.8(a)

The undersigned hereby certifies that this document is being placed in the United States mail with first-class postage attached, addressed to Assistant Commissioner for Patents, Washington, DC 20231 on the 20th day of December, 1996.


Frances M. Cunningham

Assistant Commissioner for Patents
Washington, DC 20231

Sir:

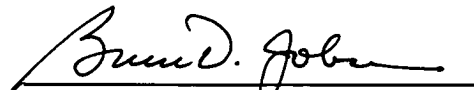
Transmitted herewith for filing is/are the following document(s):

- Information Disclosure Statement
- PTO Form 1449

If the enclosed papers are considered incomplete, the Mail Room and/or the Application Branch is respectfully requested to contact the undersigned collect at (617) 367-4600, Boston, Massachusetts.

No fee is being submitted. If the fee is insufficient, the balance may be charged to the account of the undersigned, Deposit Account No. 02-3038. A duplicate of this sheet is enclosed.

Respectfully submitted,



Bruce D. Jobse, Esq.
Reg. No.: 33,518
BOOKSTEIN & KUDIRKA, P.C.
One Beacon Street
Boston, Massachusetts 02108
Tel.: (617) 367-4600

ATTORNEY DOCKET NO.: N0003/7002
DATE: December 20, 1996
NDDH:\BD\N0003\7002\IDSTRANS.WPD



0300

ATTORNEY'S DOCKET NO.: N0003/7002

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Shane D. Mattaway et al.
Serial No.: 08/721,316
Filed: September 25, 1996
For: GRAPHIC USER INTERFACE FOR INTERNET TELEPHONY
APPLICATION

Examiner:
Art Unit:

CERTIFICATE OF MAILING UNDER 37 C.F.R. §1.8(a)

The undersigned hereby certifies that this document is being placed in the United States mail with first-class postage attached, addressed to Assistant Commissioner for Patents, Washington, DC 20231 on the 20th day of December, 1996.


Frances M. Cunningham

Assistant Commissioner for Patents
Washington, DC 20231

**STATEMENT FILED PURSUANT TO THE DUTY OF
DISCLOSURE UNDER 37 C.F.R. §§1.56, 1.97 AND 1.98**

Sir:

Pursuant to the duty of disclosure under 37 C.F.R. §§1.56, 1.97 and 1.98, the applicant requests consideration of this information disclosure statement.

Compliance with 37 C.F.R. §1.97

This information disclosure statement has been filed before the mailing date of a first office action on the merits in the above-identified application. No fee or certification is required.

Information Cited

The applicant hereby makes of record in the above-identified application the information listed on the attached form PTO-1449 (modified). The order of presentation of the references should not be construed as an indication of the relative importance of the references.

Remarks

A copy of each of the above-identified information is enclosed unless otherwise indicated on the attached form PTO-1449 (modified). It is respectfully requested that:

- The examiner consider completely the cited information, along with any other information, in reaching a determination concerning the patentability of the present claims;
- The enclosed form PTO-1449 be signed by the examiner to evidence that the cited information has been fully considered by the Patent and Trademark Office during the examination of this application;
- The citations for the information be printed on any patent which issues from this application.

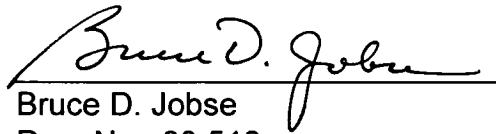
By submitting this information disclosure statement, the applicant makes no representation that a search has been performed, of the extent of any search performed, or that more relevant information does not exist.

By submitting this information disclosure statement, the applicant makes no representation that the information cited in the statement is, or is considered to be, material to patentability as defined in 37 C.F.R. §1.56(b).

By submitting this information disclosure statement, the applicant makes no representation that the information cited in the statement is, or is considered to be, in fact, prior art as defined by 35 U.S.C. §102.

It is understood by applicant that the foregoing information will be considered and, to the extent deemed appropriate by the examiner, will be reflected in the examiner's communication.

Respectfully submitted,



Bruce D. Jobse
Reg. No. 33,518
BOOKSTEIN & KUDIRKA, P.C.
One Beacon Street
Boston, Massachusetts 02108
Tel: (617) 367-4600
Attorneys for Applicant

Docket No.: N0003/7002
Date: December 20, 1996

H:\BDJ\N0003\7002\IDS.WPD



UNITED STATES DEPARTMENT OF COMMERCE

Patent and Trademark Office

Address: COMMISSIONER OF PATENTS AND TRADEMARKS

Washington, D.C. 20231

002 130-105 BCD #3

APPLICATION NUMBER	FILING/RECEIPT DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NO./TITLE
08721316	09725/96	MATTAWAY	S

✓ fmc
✓ fmc
✓ corr
nt

0222/1202

BRUCE D JOBSE
ROCKSTEIN & KUDIRKA
ONE BEACON STREET
BOSTON, MA 02108

2/2/97

0000

DATE MAILED:

12/02/96

NOTICE TO FILE MISSING PARTS OF APPLICATION

Filing Date Granted

An Application Number and Filing Date have been assigned to this application. However, the items indicated below are missing. The required items and fees identified below must be timely submitted ALONG WITH THE PAYMENT OF A SURCHARGE for items 1 and 3-6 only of \$ 130.00 for a large entity small entity in compliance with 37 CFR 1.27. The surcharge is set forth in 37 CFR 1.16(e). Applicant is given TWO MONTHS FROM THE DATE OF THIS NOTICE within which to file all required items and pay any fees required above to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

If all required items on this form are filed within the period set above, the total amount owed by applicant as a

large entity small entity (verified statement filed), is \$ 130.00

1. The statutory basic filing fee is:

- missing.
- insufficient.

Applicant must submit \$ _____ to complete the basic filing fee and/or file a verified small entity statement claiming such status (37 CFR 1.27).

2. Additional claim fees of \$ _____, including any multiple dependent claim fees, are required.

Applicant must either submit the additional claim fees or cancel additional claims for which fees are due.

3. The oath or declaration:

- is missing.
- does not cover the newly submitted items.
- does not identify the application to which it applies.
- does not include the city and state or foreign country of applicant's residence.

An oath or declaration in compliance with 37 CFR 1.63, including residence information and identifying the application by the above Application Number and Filing Date is required.

4. The signature(s) to the oath or declaration is/are:

- missing.
- by a person other than inventor or person qualified under 37 CFR 1.42, 1.43, or 1.47.

A properly signed oath or declaration in compliance with 37 CFR 1.63, identifying the application by the above Application Number and Filing Date, is required.

5. The signature of the following joint inventor(s) is missing from the oath or declaration:

An oath or declaration listing the names of all inventors and signed by the omitted inventor(s), identifying this application by the above Application Number and Filing Date, is required.

6. A \$ _____ processing fee is required since your check was returned without payment (37 CFR 1.21(m)).

7. Your filing receipt was mailed in error because your check was returned without payment.

8. The application does not comply with the Sequence Rules.

See attached "Notice to Comply with Sequence Rules 37 CFR 1.821-1.825."

9. OTHER:

Direct the response and any questions about this notice to "Attention: Box-Missing Parts."

A copy of this notice MUST be returned with the response.

Customer Service Center
Initial Patent Examination Division (703) 308-1202



N0003/7002
PATENT

Assistant Commissioner for Patents
Box Missing Parts
Washington, DC 20231

TRANSMITTAL LETTER

Sir:

Transmitted herewith for filing in the Patent Application of

Applicant: Shane D. Mattaway, et al.

Serial No.: 08/721,316

Filed: September 25, 1996

For: GRAPHIC USER INTERFACE FOR INTERNET TELEPHONY APPLICATION

are the following papers:

- X Response to Notice to File Missing Parts of Application
- X Signed Declaration
- X Copy of Notice to File Missing Parts of Application *Filing Date Granted*

A check in the amount of \$130.00 is enclosed to cover the surcharge. The Commissioner is hereby authorized to charge any other fees under 37 C.F.R. §§1.16 and 1.17 that may be required, or credit any overpayment, to our Deposit Account No. 02-3038.

Respectfully submitted,

Bruce D. Jobse, Esq.
Reg. No. 33,518
BOOKSTEIN & KUDIRKA, P.C.
One Beacon Street
Boston, MA 02108
(617) 367-4600

BDJ:rc

H:\BDJ\N0003\7002\MISPRTRR.WPD



PATENT
N0003/7002

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re The Application of:)	Examiner: Not yet assigned
Shane D. Mattaway, et al.)	Art Unit: Not yet assigned
Serial No. 08/721,316)	
Filed: September 25, 1996)	Bookstein & Kudirka, P.C.
For: GRAPHIC USER INTERFACE)	One Beacon Street
FOR INTERNET TELEPHONY)	Boston, MA 02108
APPLICATION)	

CERTIFICATE OF MAILING

I hereby certify that the following Response to Notice to File Missing Parts is being deposited with the United States Postal Service as first class mail pursuant to 37 C.F.R. 1.8 in an envelope addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231, Attention: Box Missing Parts, on December 17, 1996.

Rivki Cohen

Assistant Commissioner for Patents
Washington, D.C. 20231
Attention: Box Missing Parts

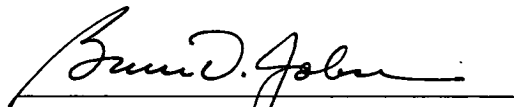
Sir:

RESPONSE TO NOTICE TO FILE MISSING PARTS OF APPLICATION

In response to the Notice to File Missing Parts of Application-Filing Date Granted mailed December 2, 1996, enclosed herewith is a Declaration executed by the named inventors for the above-referenced application. A check for \$130.00 to cover the surcharge for this Response is enclosed.

The Commissioner is hereby authorized to charge any other fees under 37 C.F.R. §§1.16 and 1.17 that may be required, or credit any overpayment, to our Deposit Account No. 02-3038.

Respectfully submitted,



Bruce D. Jobse, Esq.
Registration No. 33,518
BOOKSTEIN & KUDIRKA, P.C.
One Beacon Street
Boston, MA 02108
(617) 367-4600

BDJ:rc

Date: December 17, 1996

ATTORNEY'S DOCKET NO. N0003/7002

x020297

H:\BDJ\N0003\7002\RESPMISS.WPD

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are stated below next to my name:

I believe I am an original, first and joint inventor the subject matter which is claimed and for which a patent is sought on the invention entitled **GRAPHIC USER INTERFACE FOR INTERNET TELEPHONY APPLICATION** the specification of which was filed on September 25, 1996 under Attorney's Docket Number N0003/7002, now U.S. Patent Application Serial No. 08/721,316.

I hereby state that I have reviewed and understand the contents of the above identified patent application, including the claims as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with 37 C.F.R. 1.56.

I hereby claim the benefit of foreign priority under 35 U.S.C. 119 of any foreign application(s) for patent or inventor's certificate having a filing date before that of the application the priority of which is claimed:

Prior Foreign Application(s):	Priority Claimed		
			Yes <input type="checkbox"/> No <input type="checkbox"/>
(Number)	(Country)	(Filing Date)	

I hereby claim the benefit of United States priority under 35 U.S.C. 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in a listed prior United States application in the manner provided by the first paragraph of 35 U.S.C. 112, I acknowledge the duty to disclose information material to the patentability of this application as defined in 37 C.F.R. 1.56 which occurred between the filing date of the prior application and the national or PCT international filing date of this application.

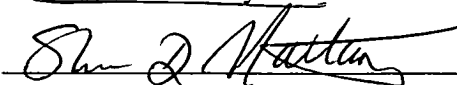
08/533,115	9/25/95	pending
(Application Serial #)	(Filing Date)	(Status)
60/024,251	8/21/96	pending
(Application Serial #)	(Filing Date)	(Status)
60/025,415	9/4/96	pending
(Application Serial #)	(Filing Date)	(Status)

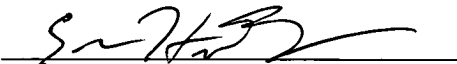
I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. 1001 and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

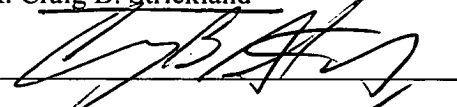
POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorneys and/or agents to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

7
Bruce D. Jobse Reg. No. 33,518 Paul E. Kudirka Reg. No. 26,931
Arthur Z. Bookstein Reg. No. 22,958 John F. Perullo Reg. No. 36,265
Philip L. Conrad Reg. No. 34,567 Steven G. Saunders Reg. No. 36,265
Paul J. Cook Reg. No. 20,280

Send correspondence to Bruce D. Jobse, BOOKSTEIN & KUDIRKA, P.C., One Beacon Street, Boston, Massachusetts, 02108.

1-00
FULL NAME OF INVENTOR: Shane D. Mattaway
INVENTOR'S SIGNATURE:  DATE: 12/8/96
RESIDENCE: 826 Periwinkle, Boca Raton, FL 33486
CITIZENSHIP: U.S.A.
POST OFFICE ADDRESS: 826 Periwinkle, Boca Raton, FL 33486

2-00
FULL NAME OF INVENTOR: Glenn W. Hutton
INVENTOR'S SIGNATURE:  DATE: 12-2-96
RESIDENCE: 9725 Hammocks Boulevard, #206, Miami, FL 33196
CITIZENSHIP: Canada
POST OFFICE ADDRESS: 9725 Hammocks Boulevard, #206, Miami, FL 33196

3-00
FULL NAME OF INVENTOR: Craig B. Strickland
INVENTOR'S SIGNATURE:  DATE: 12/10/96
RESIDENCE: 5713 NW 65th Terrace, Tamarac, FL 33321
CITIZENSHIP: ~~U.S.A.~~ Canada (CBS)
POST OFFICE ADDRESS: 5713 NW 65th Terrace, Tamarac, FL 33321

H:\BDJ\N0003\7002\DECL.WPD



UNITED STATES DEPARTMENT OF COMMERCE
 Patent and Trademark Office
 Address: COMMISSIONER OF PATENTS AND TRADEMARKS
 Washington, D.C. 20231

APPLICATION NUMBER	FILING/RECEIPT DATE	FIRST NAMED APPLICANT	ATTORNEY DOCKET NO./TITLE
087721,316	09/25/96	MATTAWAY	S

0222/1202

BRUCE D JOBSE
 BOOKSTEIN & KUDIRKA
 ONE BEACON STREET
 BOSTON MA 02108

0000

DATE MAILED:

12/02/96

NOTICE TO FILE MISSING PARTS OF APPLICATION
Filing Date Granted

An Application Number and Filing Date have been assigned to this application. However, the items indicated below are missing. The required items and fees identified below must be timely submitted ALONG WITH THE PAYMENT OF A SURCHARGE for items 1 and 3-6 only of \$ 130.00 for a large entity small entity in compliance with 37 CFR 1.27. The surcharge is set forth in 37 CFR 1.16(e). Applicant is given TWO MONTHS FROM THE DATE OF THIS NOTICE within which to file all required items and pay any fees required above to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

If all required items on this form are filed within the period set above, the total amount owed by applicant as a large entity small entity (verified statement filed), is \$ 130.00.

- 1. The statutory basic filing fee is:
 - missing.
 - insufficient.
 Applicant must submit \$ _____ to complete the basic filing fee and/or file a verified small entity statement claiming such status (37 CFR 1.27).
- 2. Additional claim fees of \$ _____, including any multiple dependent claim fees, are required. Applicant must either submit the additional claim fees or cancel additional claims for which fees are due.
- 3. The oath or declaration:
 - is missing.
 - does not cover the newly submitted items.
 - does not identify the application to which it applies.
 - does not include the city and state or foreign country of applicant's residence.
 An oath or declaration in compliance with 37 CFR 1.63, including residence information and identifying the application by the above Application Number and Filing Date is required.
- 4. The signature(s) to the oath or declaration is/are:
 - missing.
 - by a person other than inventor or person qualified under 37 CFR 1.42, 1.43, or 1.47.
 A properly signed oath or declaration in compliance with 37 CFR 1.63, identifying the application by the above Application Number and Filing Date, is required.
- 5. The signature of the following joint inventor(s) is missing from the oath or declaration:

An oath or declaration listing the names of all inventors and signed by the omitted inventor(s), identifying this application by the above Application Number and Filing Date, is required.
- 6. A \$ _____ processing fee is required since your check was returned without payment (37 CFR 1.21(m)).
- 7. Your filing receipt was mailed in error because your check was returned without payment.
- 8. The application does not comply with the Sequence Rules.
See attached "Notice to Comply with Sequence Rules 37 CFR 1.821-1.825."
- 9. OTHER:

Direct the response and any questions about this notice to "Attention: Box Missing Parts."

A copy of this notice MUST be returned with the response. *C. Horne*

Customer Service Center
 Initial Patent Examination Division (703) 308-1202



PATENT
September 25, 1996
N0003/7002

Assistant Commissioner for Patents
Box Patent Application
Washington, DC 20231

NEW APPLICATION TRANSMITTAL LETTER

Sir:

Transmitted herewith for filing is the Patent Application of

Inventor(s): Shane D. Mattaway, Glenn W. Hutton, and Craig B. Strickland Et al.

For: GRAPHIC USER INTERFACE FOR INTERNET TELEPHONY APPLICATION

This application is a continuation-in-part of U.S. Patent Application Serial No. 08/533,115, entitled, Point-to-Point Internet Protocol, by Glenn W. Hutton, filed September 25, 1995.

Enclosed are the following papers required to obtain a filing date under 37 C.F.R. §1.53(b):

27 Sheets of Informal Drawings
80 Pages of Specification, Including Claims and Abstract
6 Claims

The following papers, if indicated by an X, are also enclosed:

A Declaration and Power of Attorney
 An Assignment of the invention
 An Information-Disclosure Statement, Form PTO-1449 and a copy of each cited reference
 A Small-Entity Declaration
 A Certificate of Express Mailing, Express Mail Label No. EM316008331US

FEE CALCULATION:

Total Claims: 6 - 20 = 0 X \$22.00 = \$00.00

Independent Claims: 1 - 3 = 0 X \$78.00 = \$00.00

Basic Fee: \$750.00

Total of Above Calculations:

\$750.00

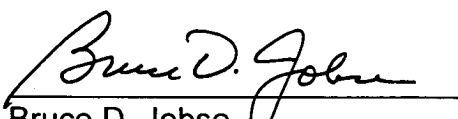
X A check in the amount of \$750.00 is enclosed to cover the Filing Fee.

_____ A check in the amount of \$40.00 is enclosed to cover the Recording Fee for the Assignment. A duplicate copy of this transmittal letter is enclosed.

The Commissioner is hereby authorized to charge any fees under 37 C.F.R. §1.16 and 1.17 that may be required by this paper or any paper filed in connection with this Patent Application, or credit any overpayment, to our Deposit Account No. 02-3038.

Please address all communications and telephone calls to the undersigned.

Respectfully submitted,

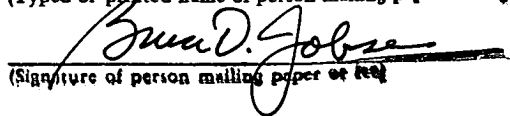

Bruce D. Jobse
Reg. No. 33,518
BOOKSTEIN & KUDIRKA, P.C.
One Beacon Street
Boston, MA 02108
(617) 367-4600

Express Mail mailing label number EM 316008331US

Date of Deposit 9/25/96

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to: The Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Bruce D. Jobse
(Typed or printed name of person mailing paper or fee)


(Signature of person mailing paper or fee)

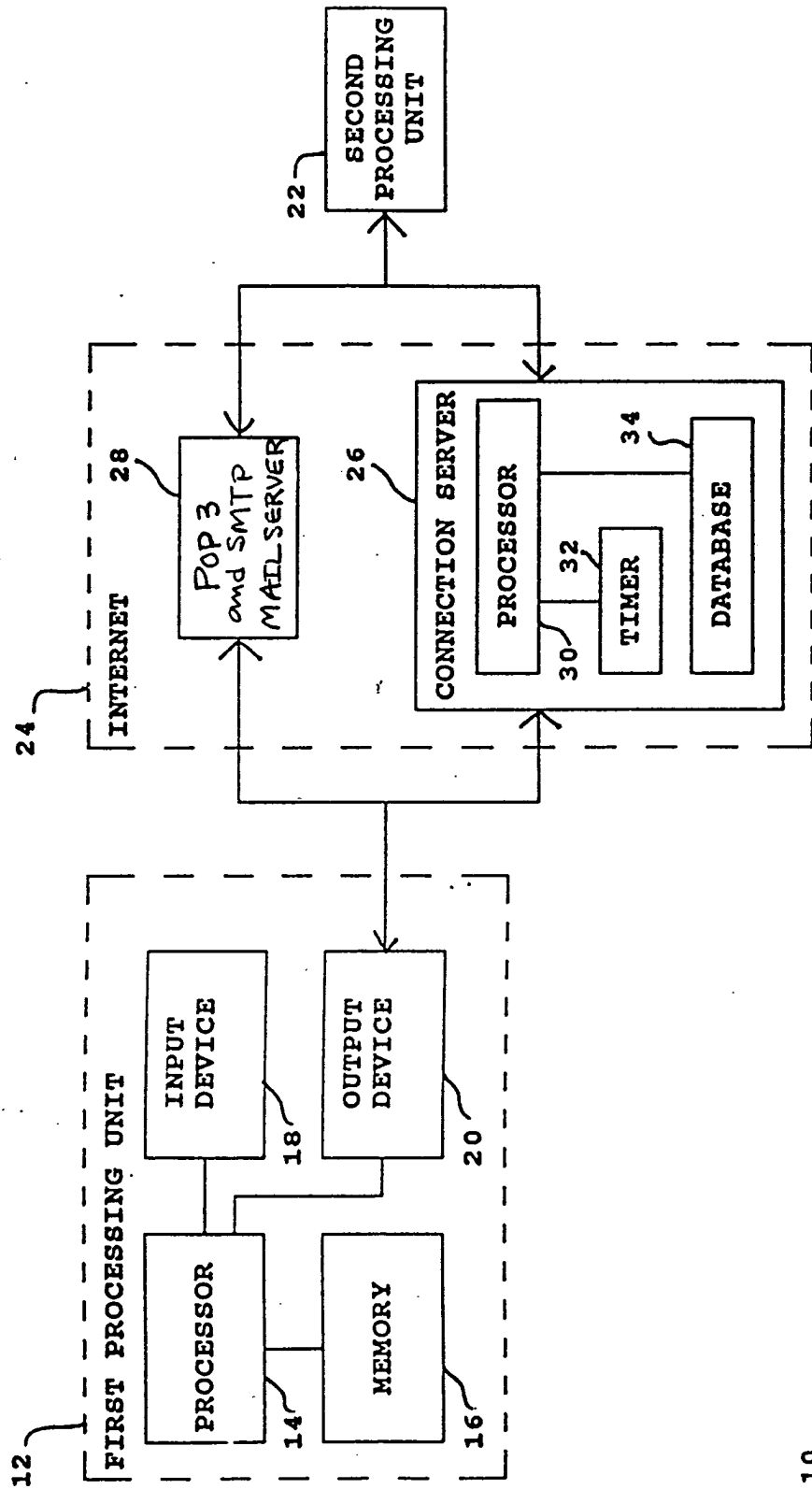


FIG. 1

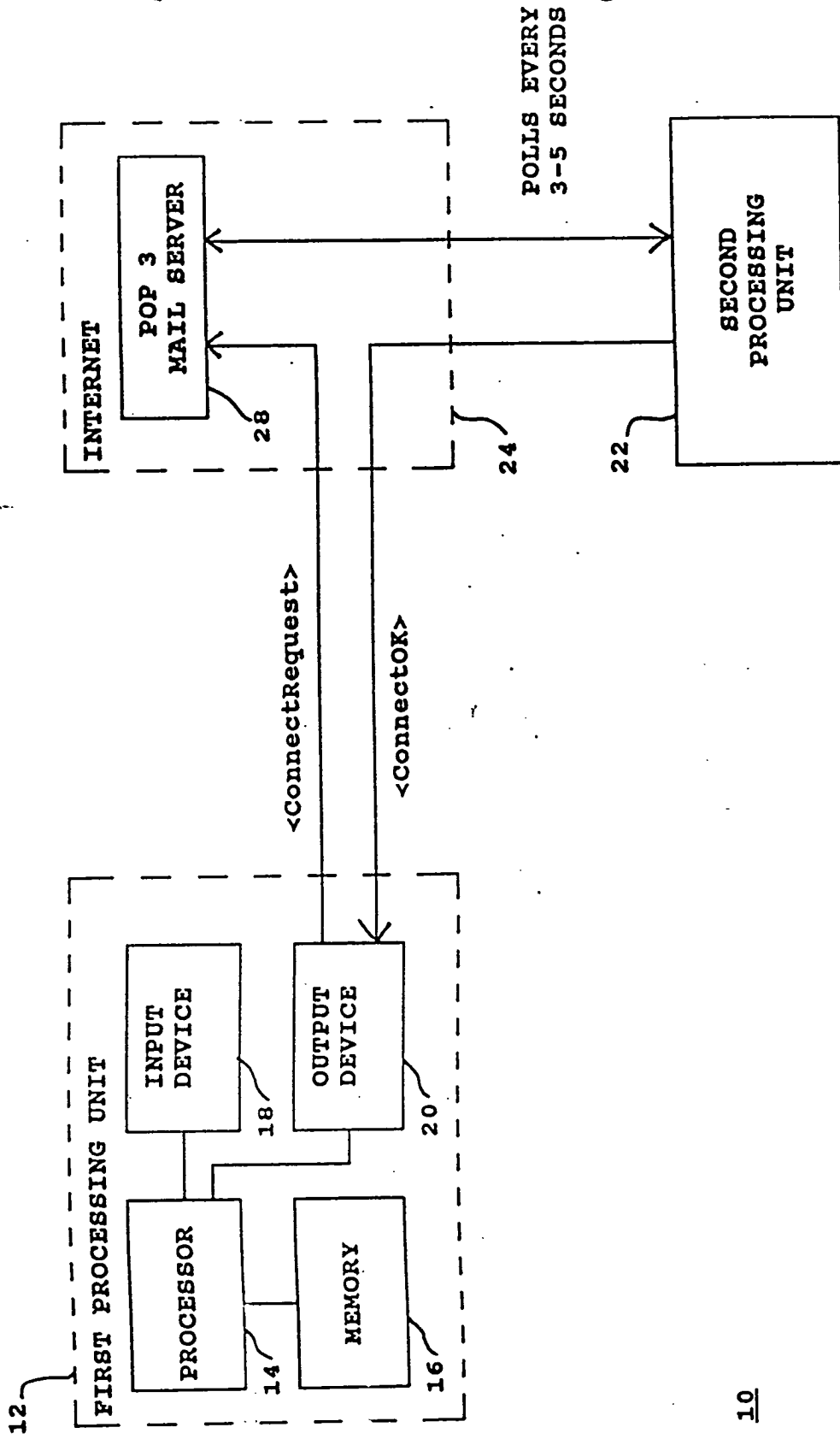


FIG. 2

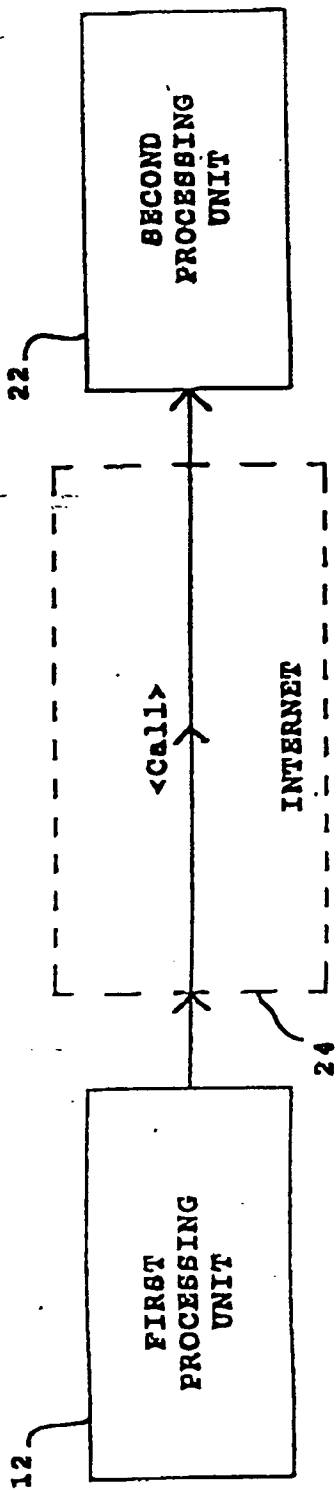


FIG. 3

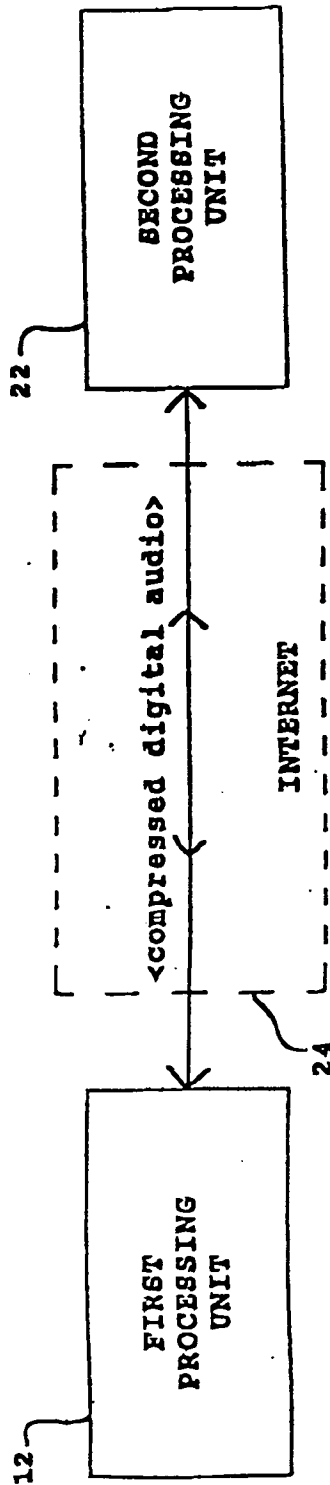


FIG. 4

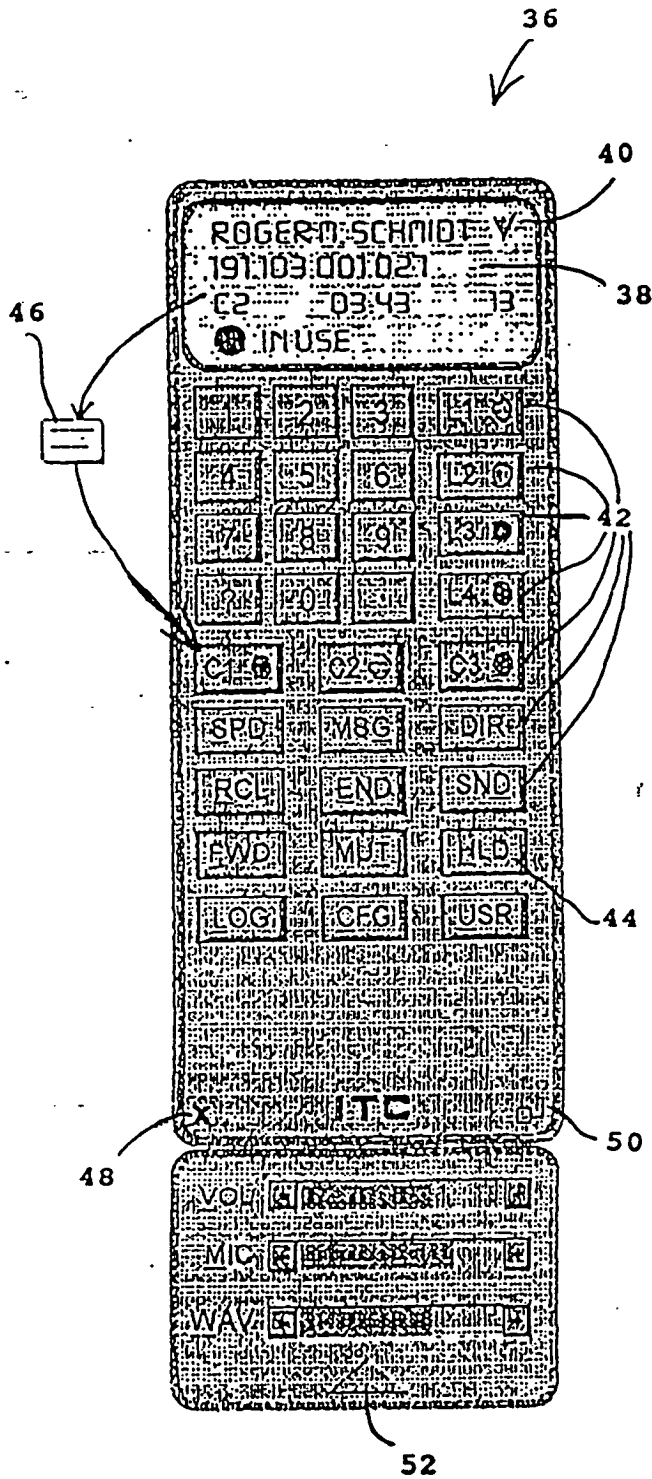


FIG. 5

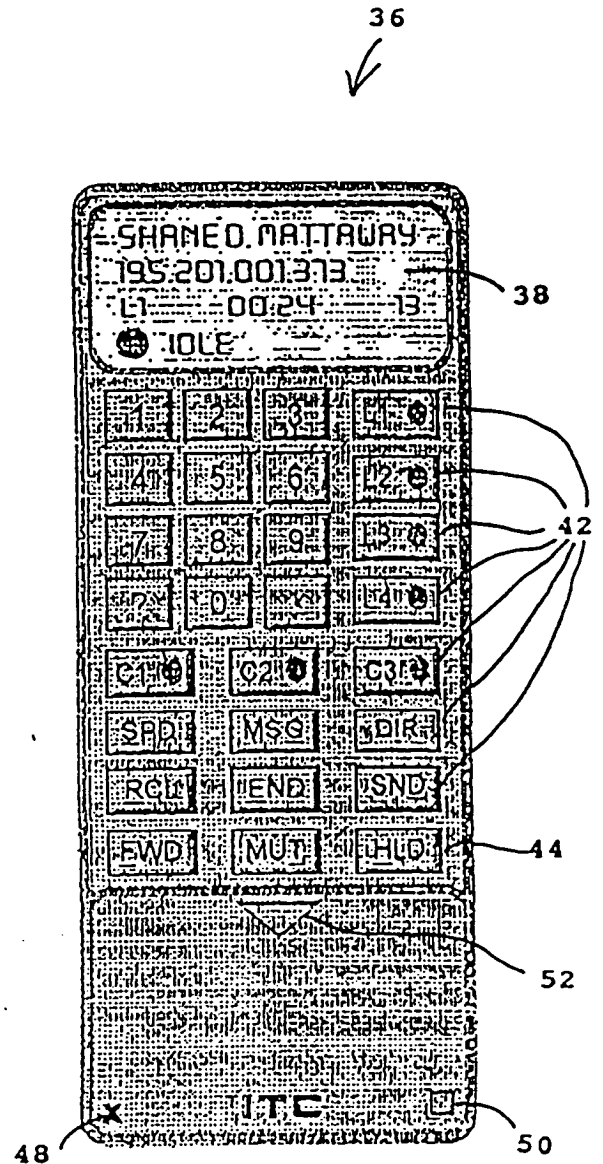


FIG. 6

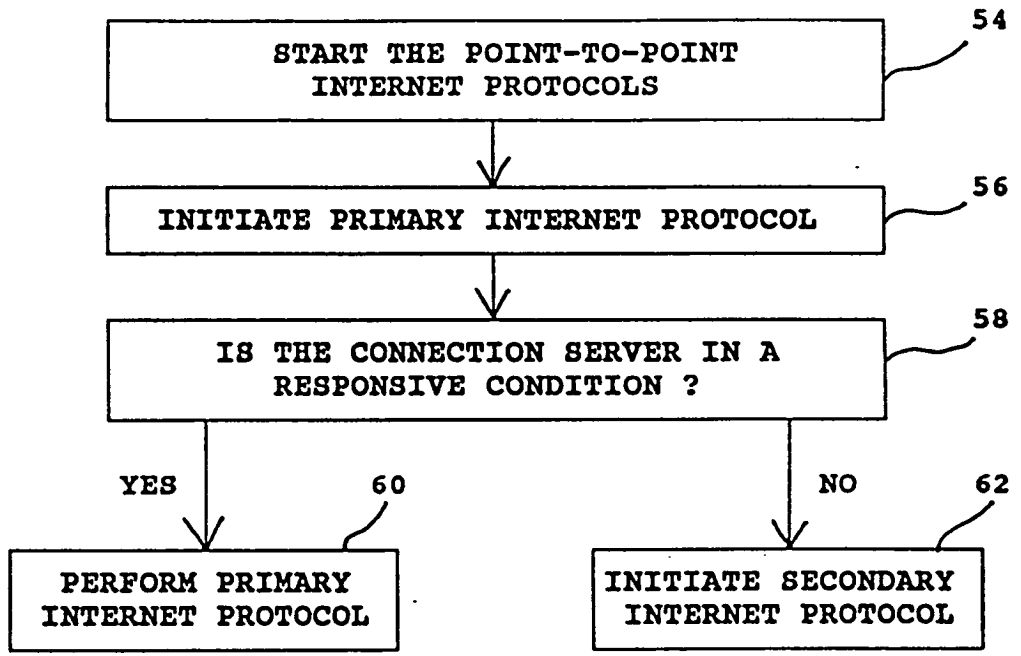


FIG. 7

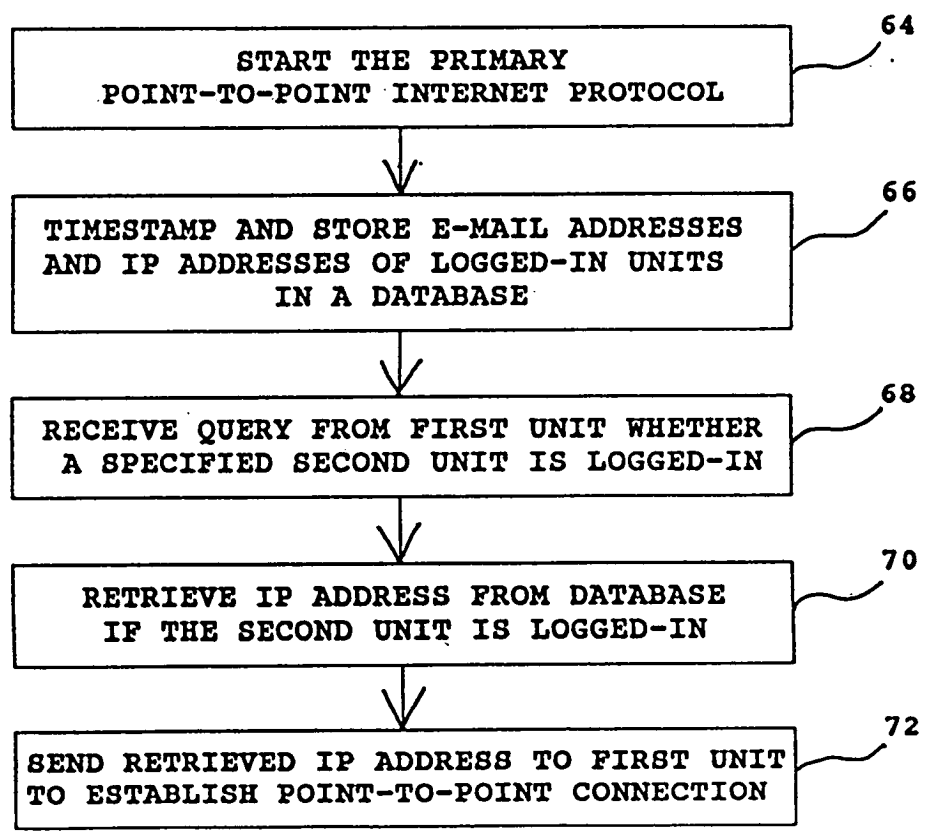


FIG. 8

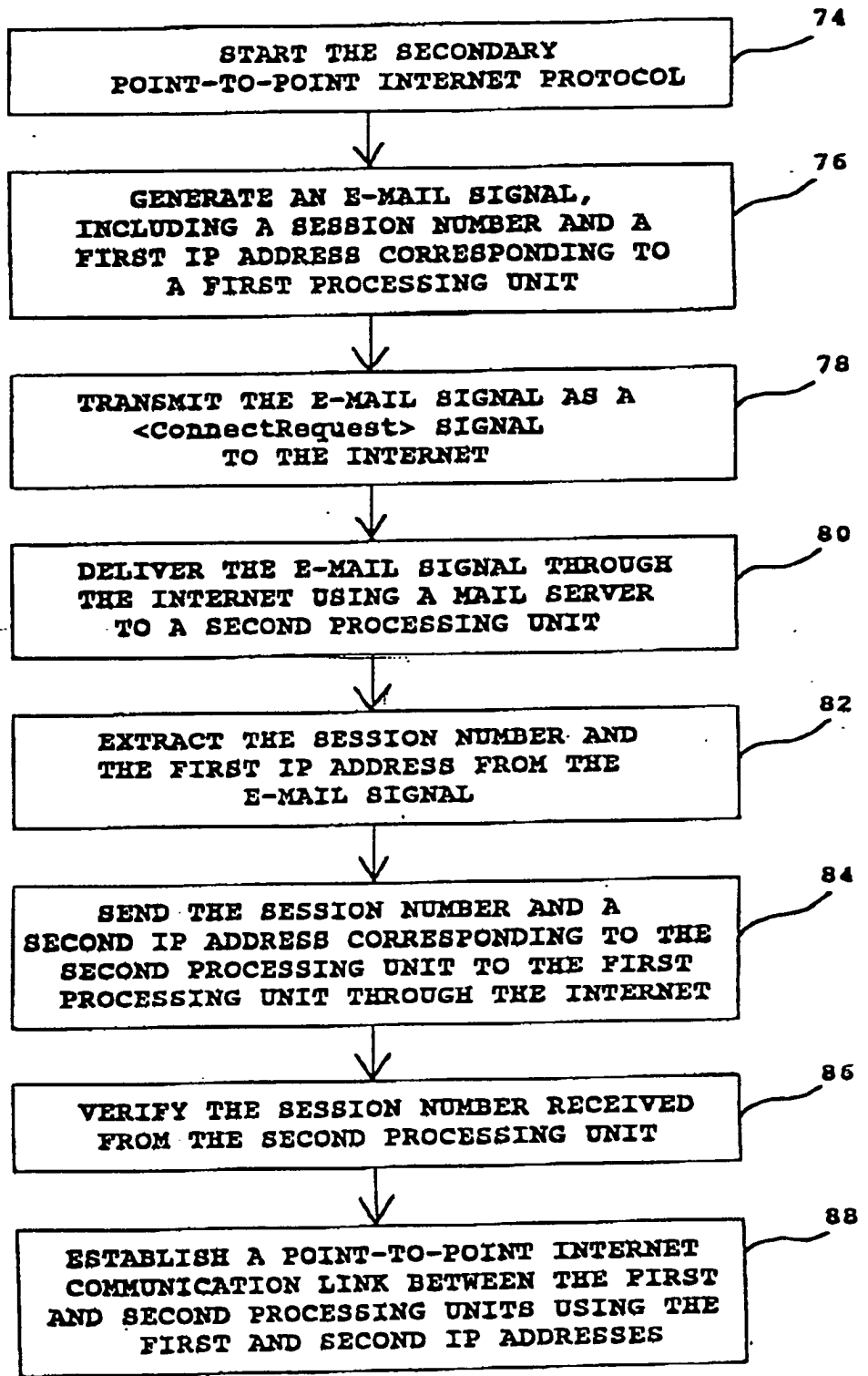


FIG. 9

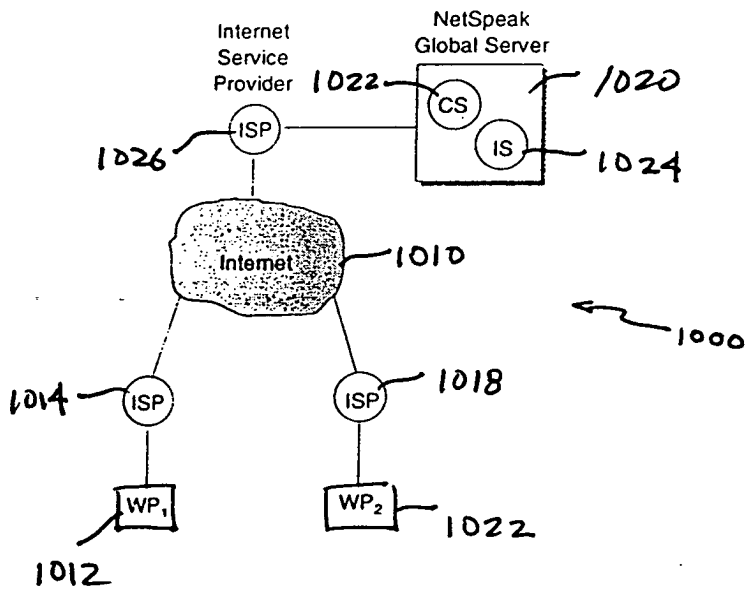


Fig. 10

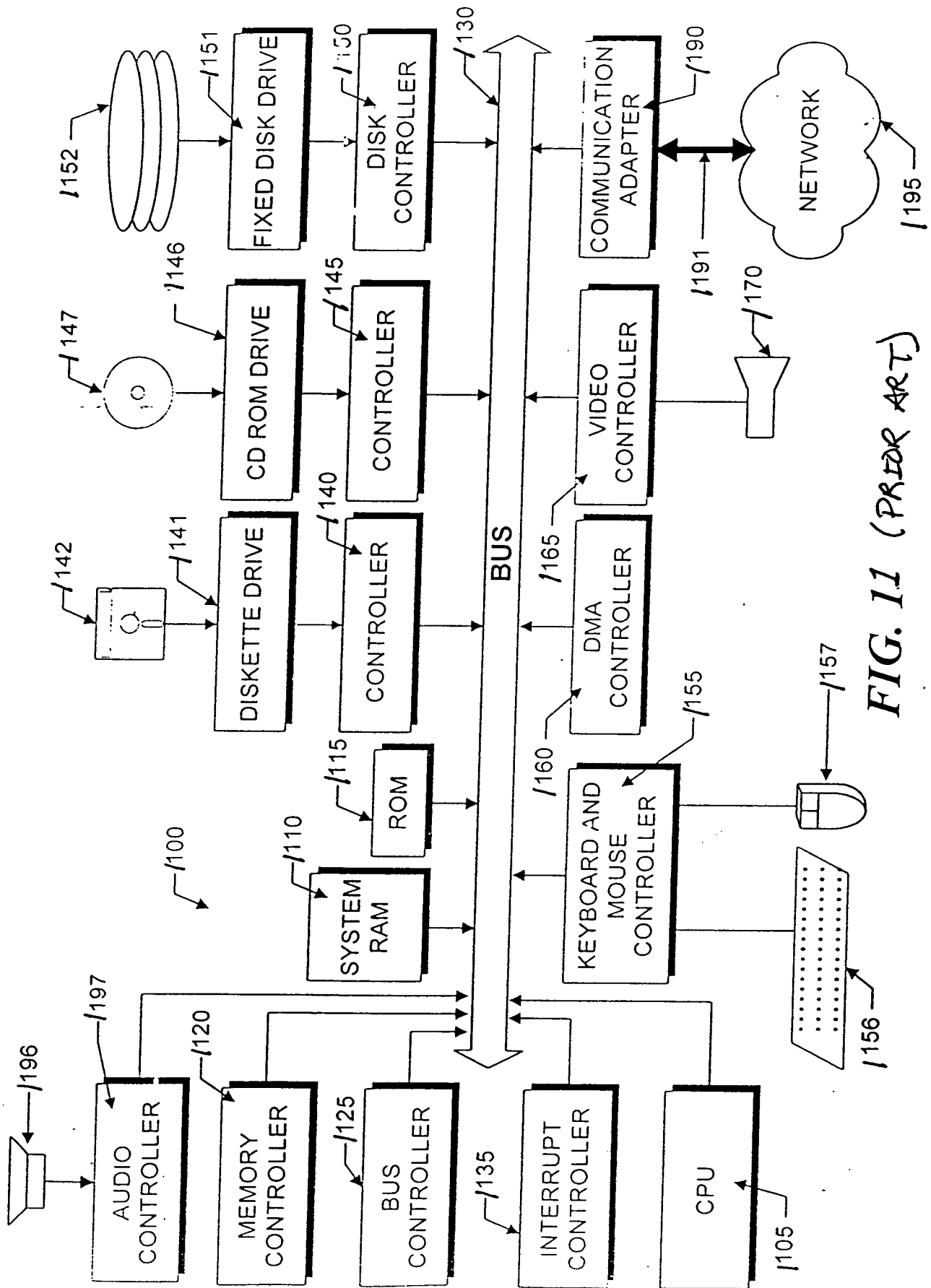


FIG. 11 (PRIOR ART)

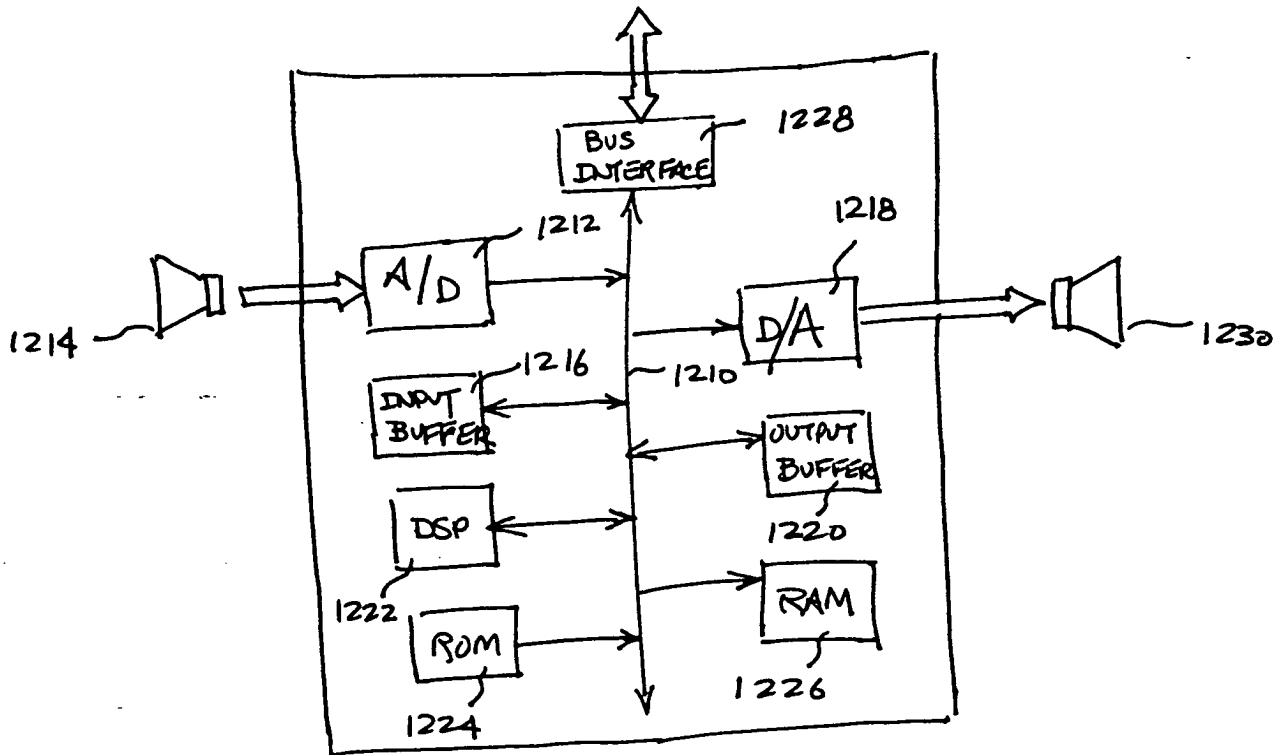


Fig. 12
(PRIOR ART)

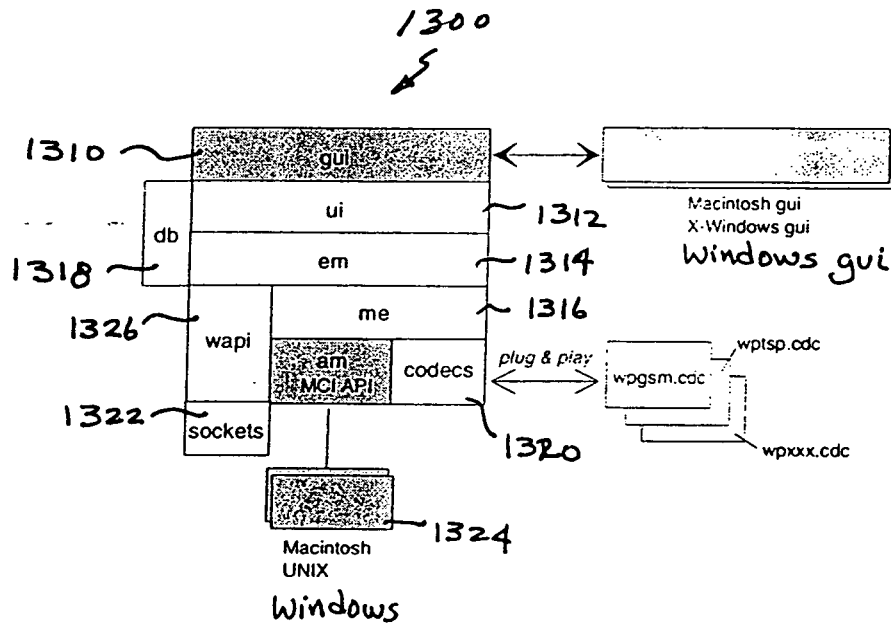


Fig. 13A

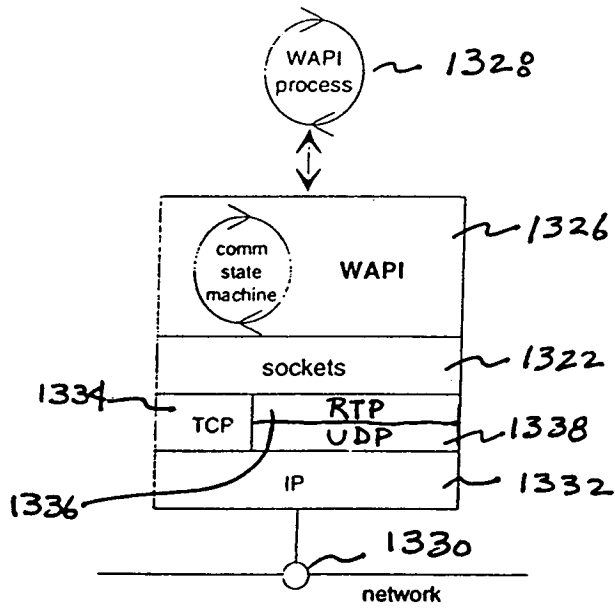


Fig. 13B

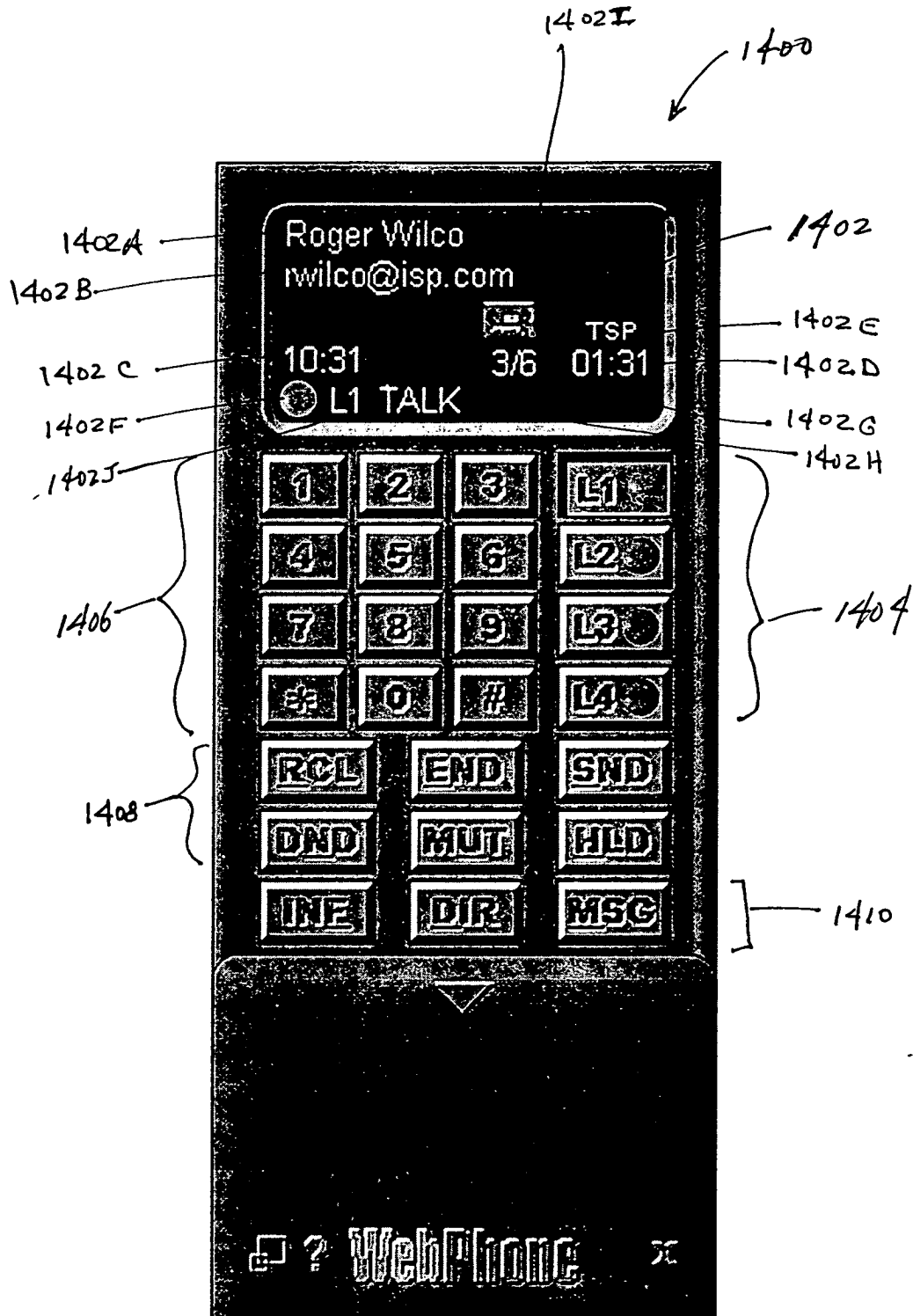


Figure 14

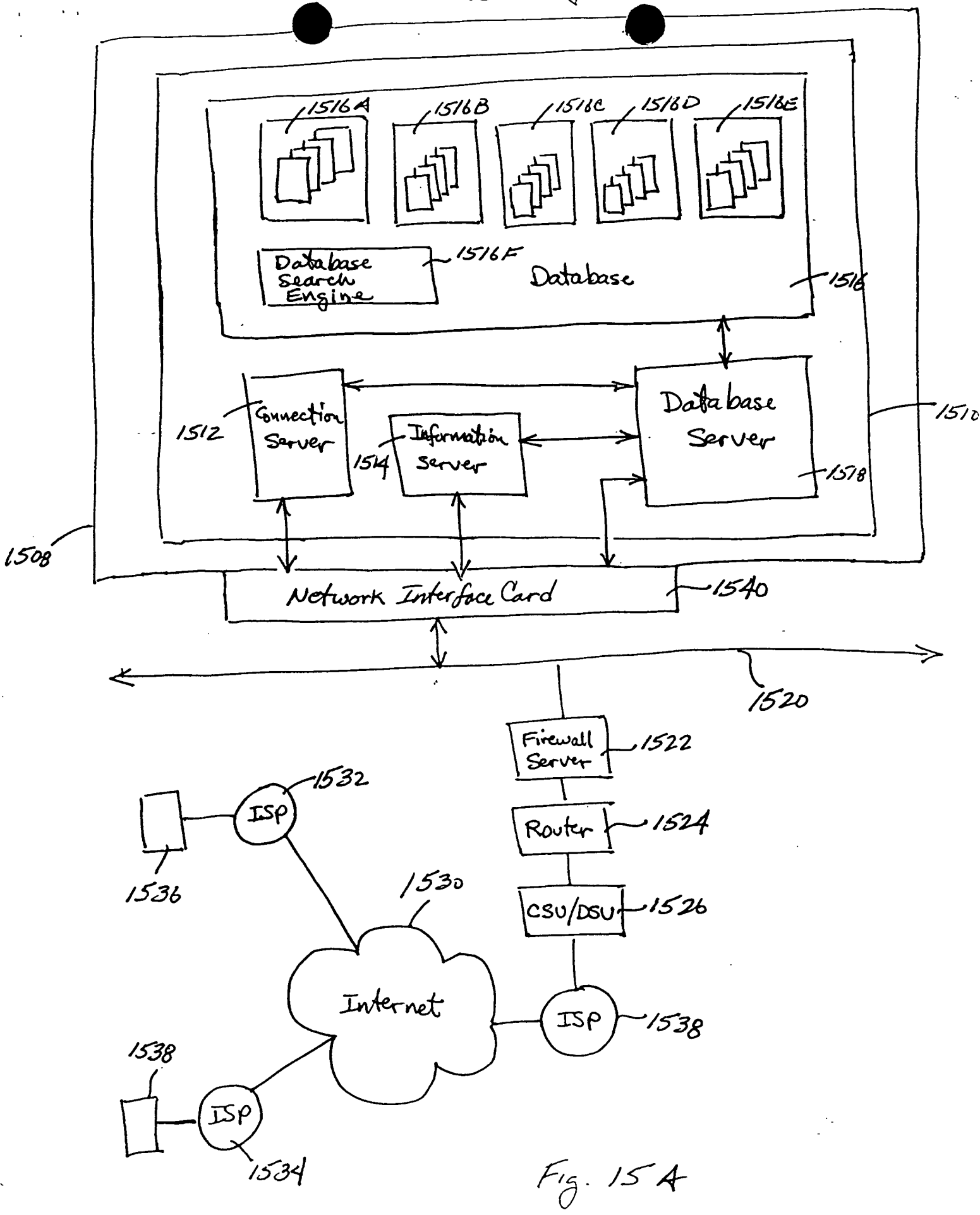


Fig. 15 A

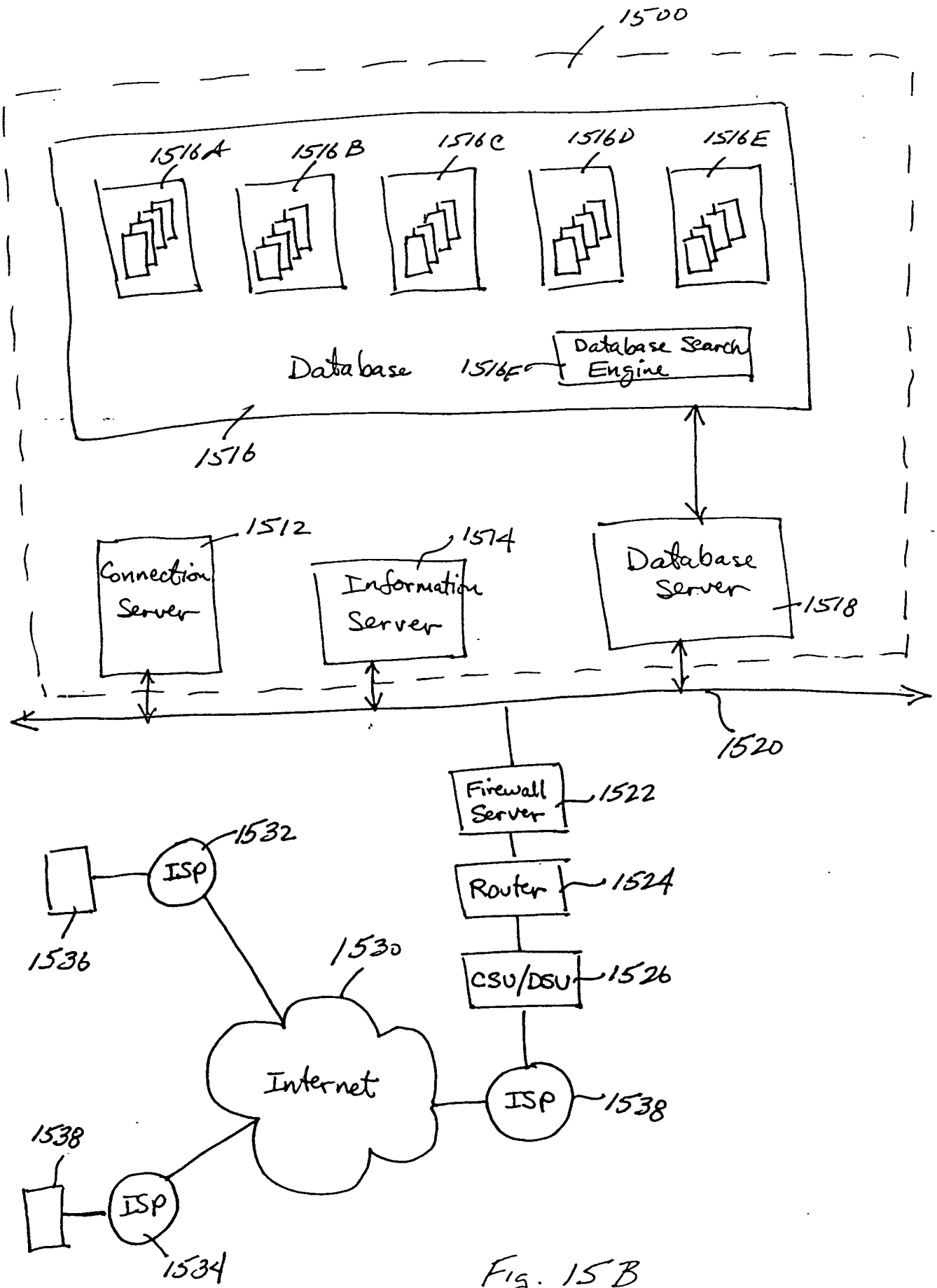


Fig. 15B

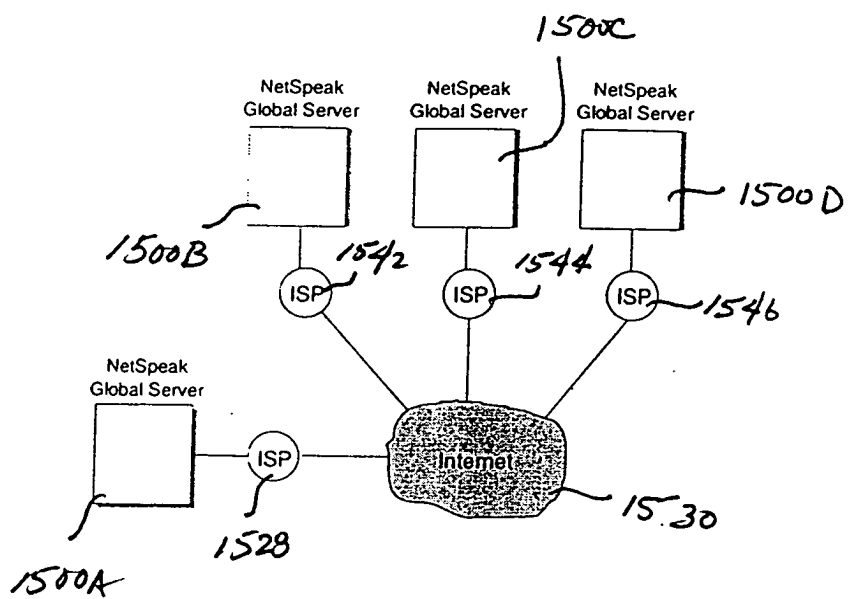


Fig. 15C

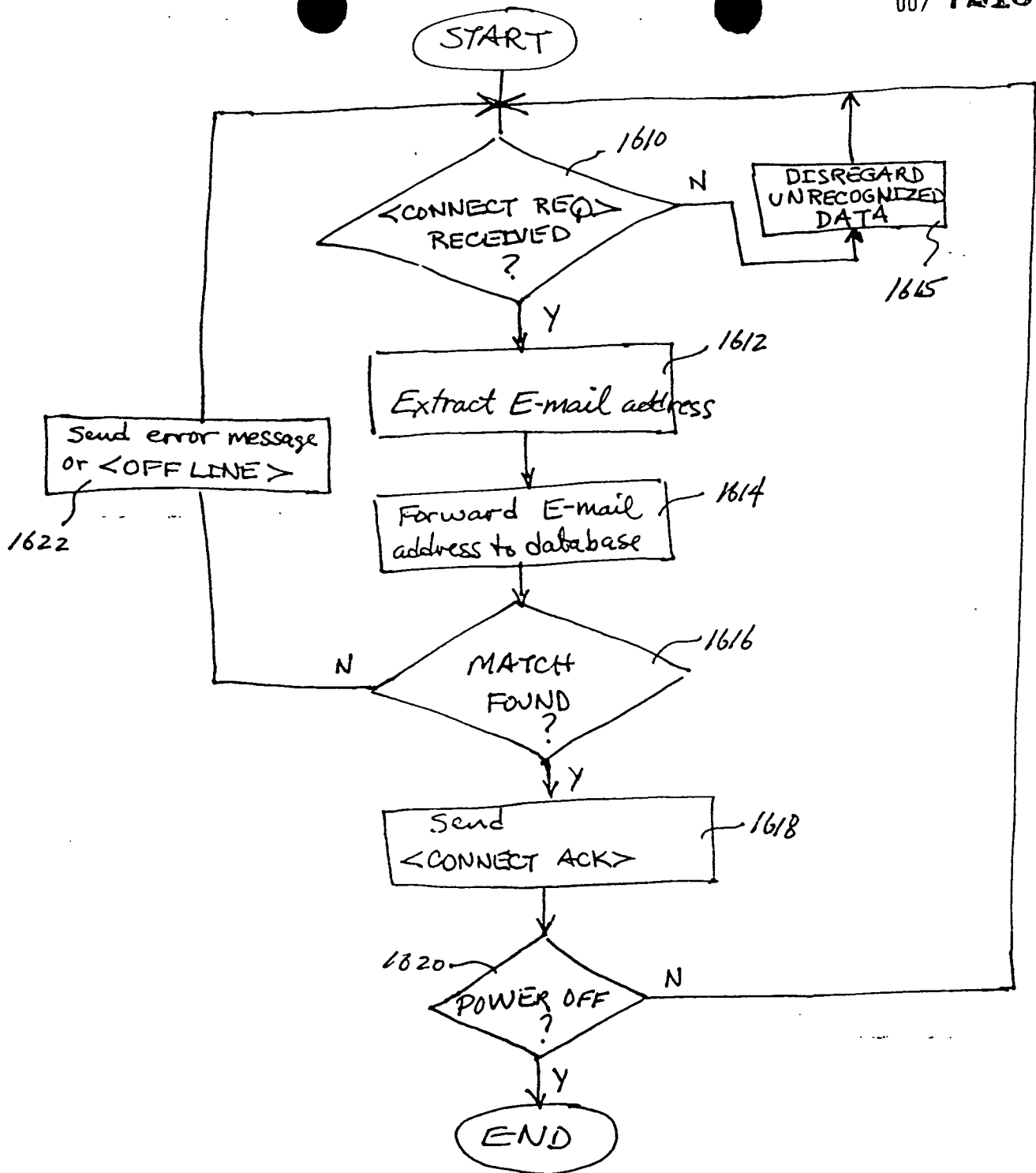


Fig. 16A

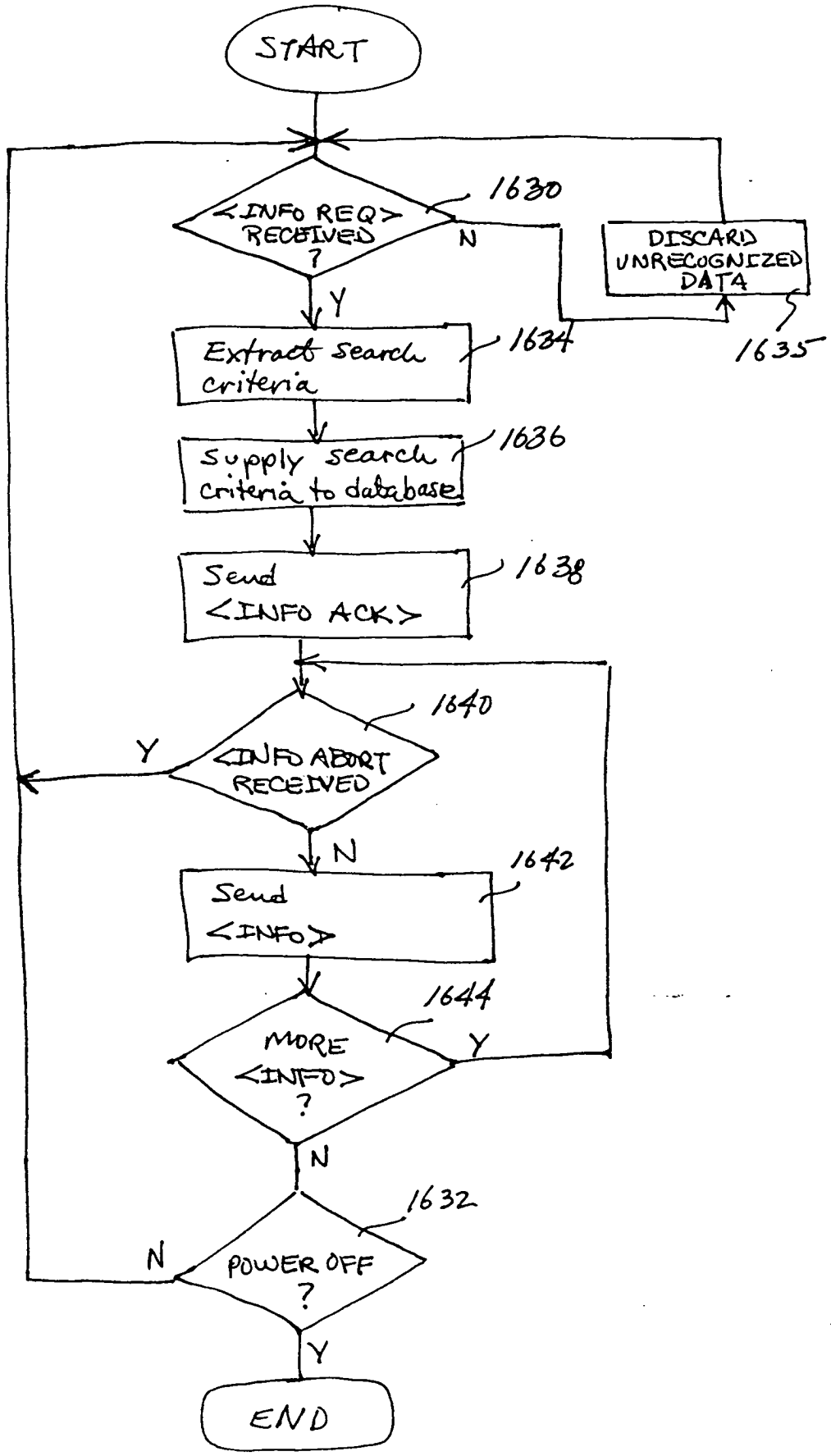


Fig. 16B

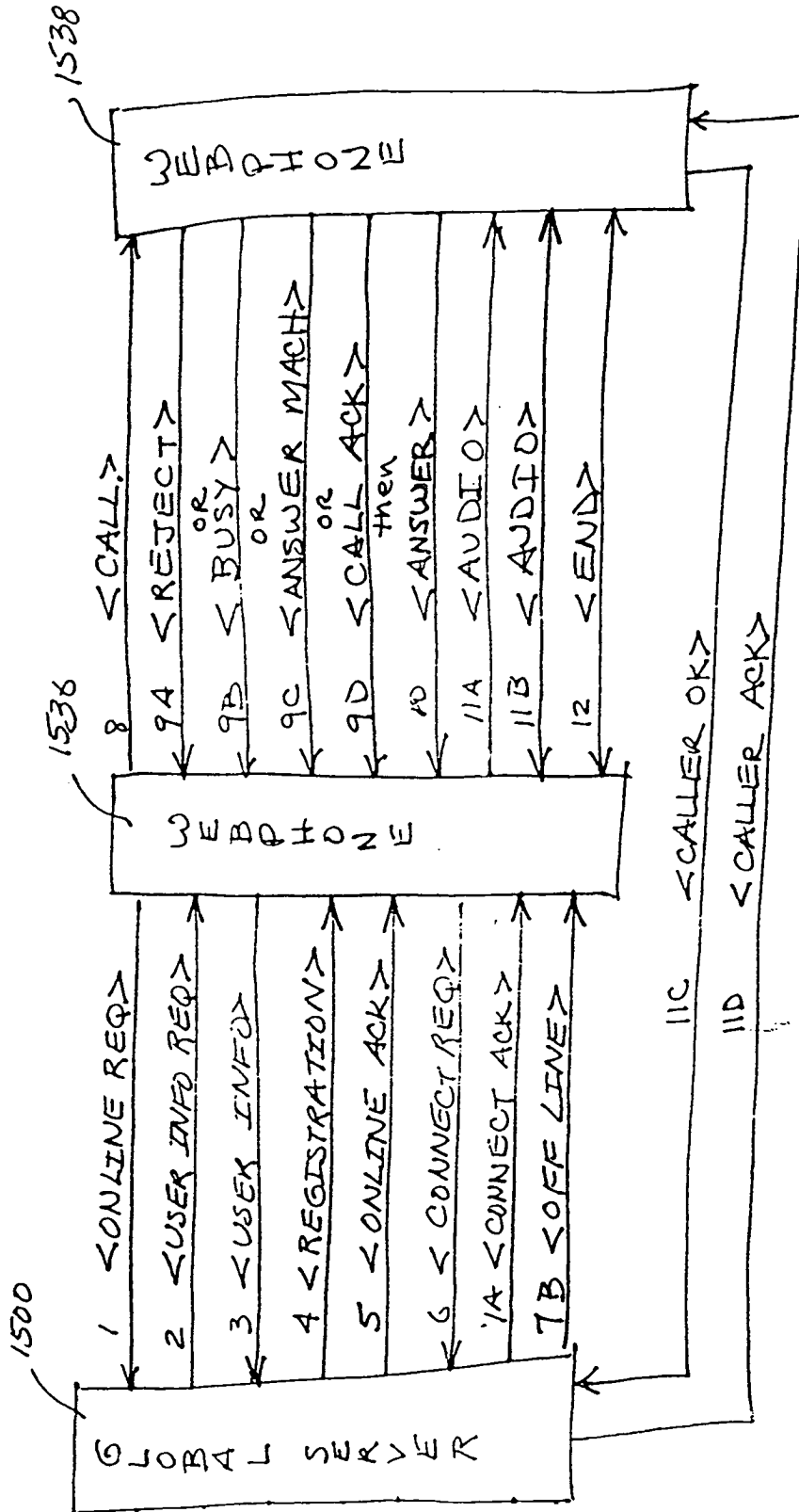


Fig. 17A

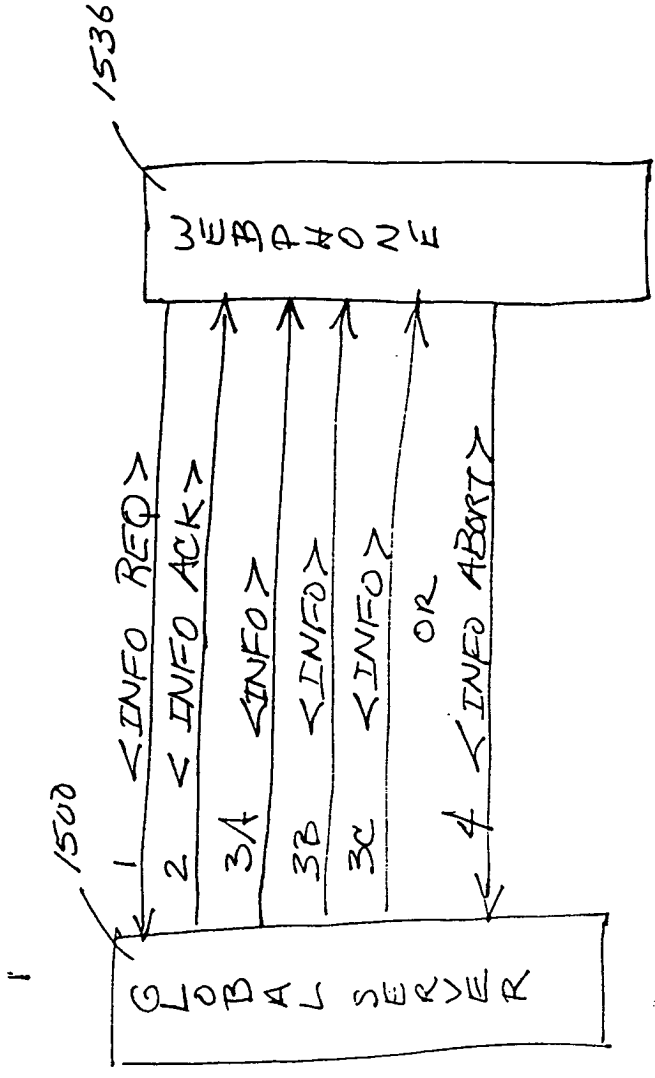


Fig. 17B

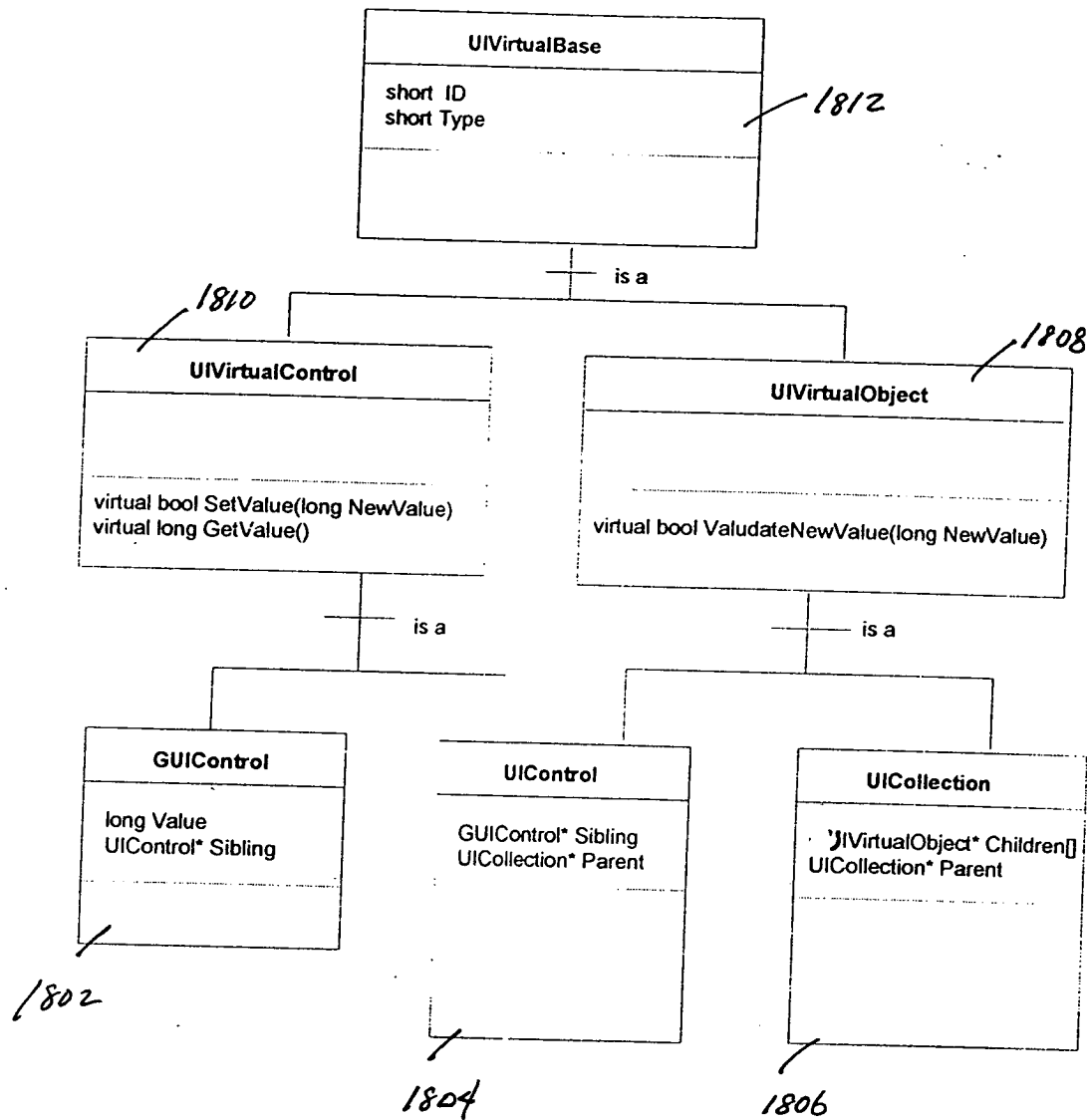


Fig. 18A

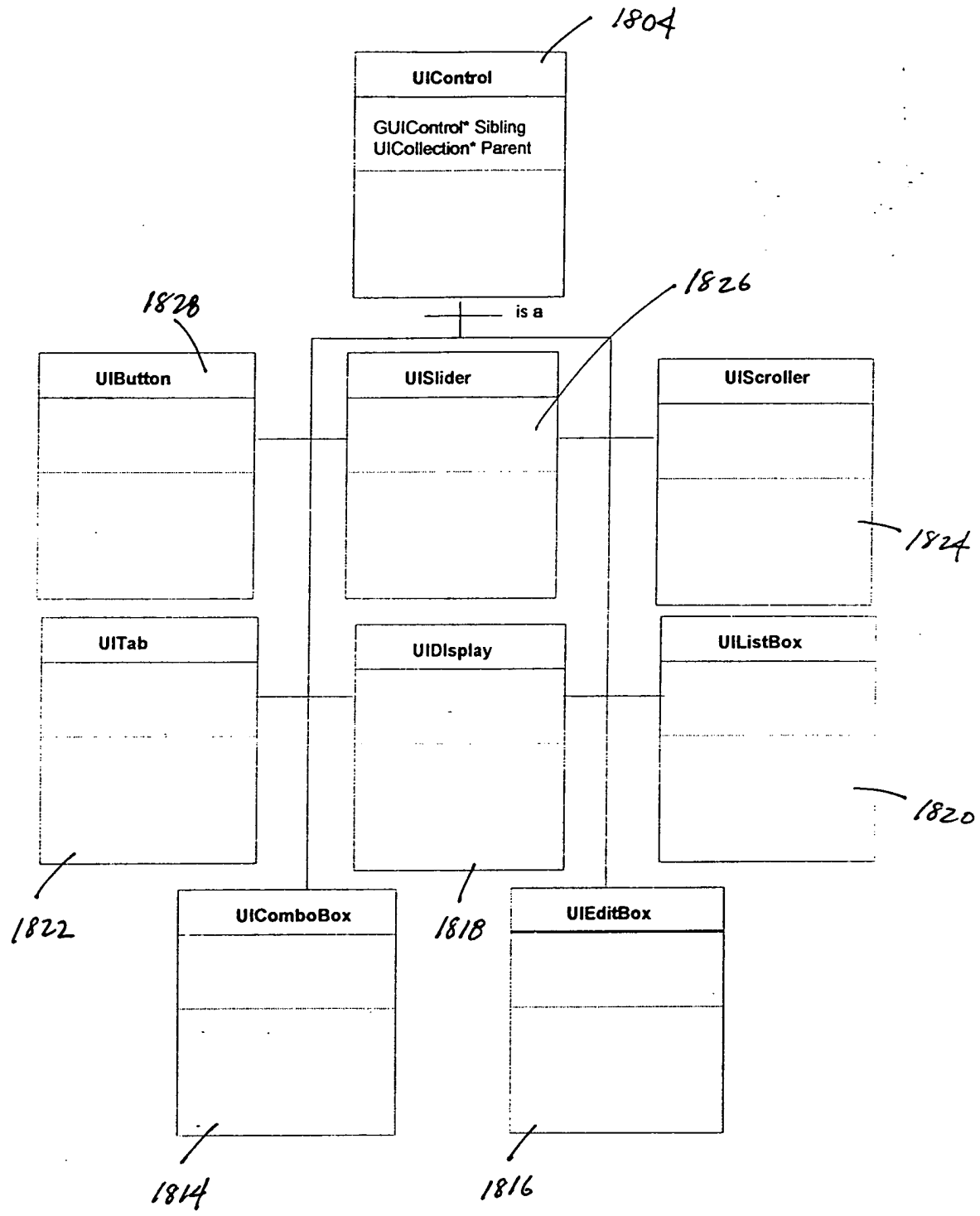


Fig. 18B

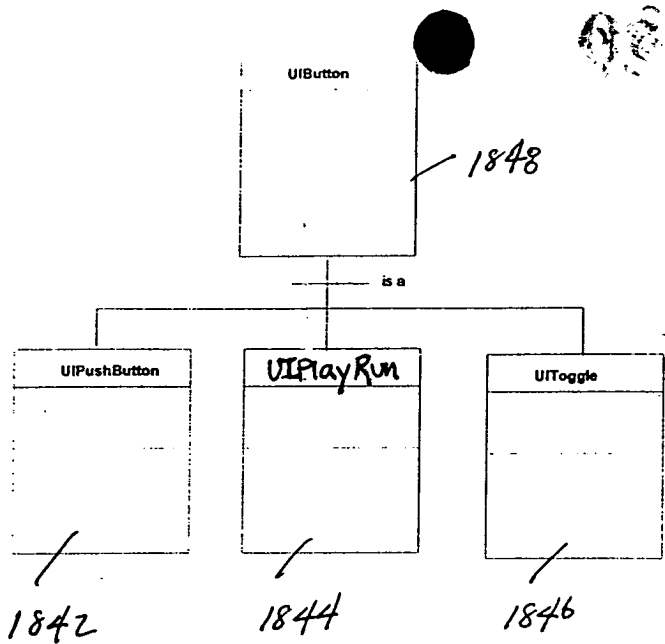


Fig. 18C

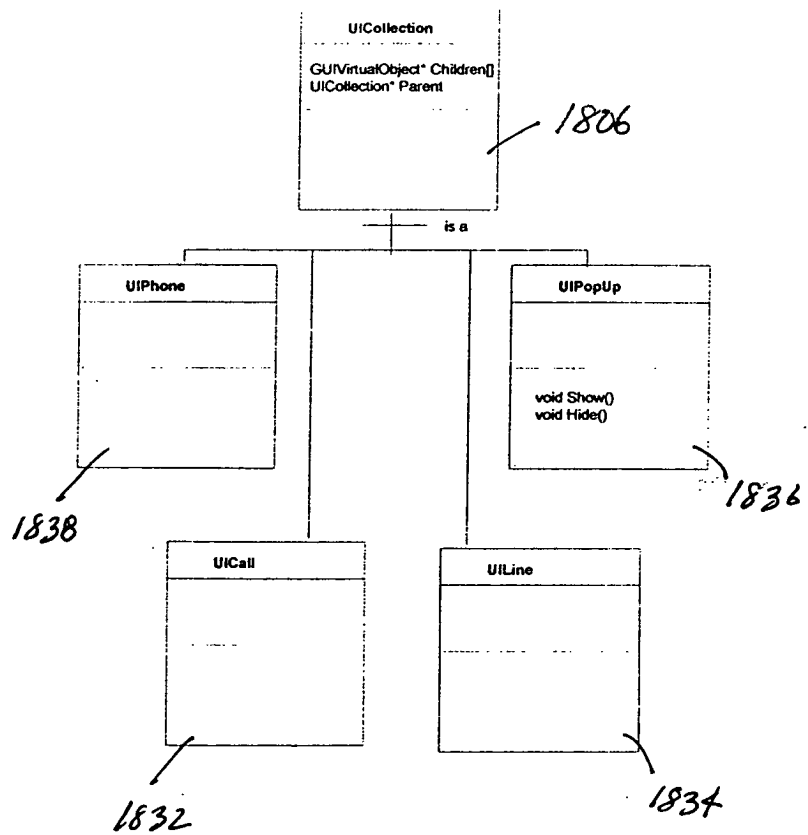


Fig. 18D

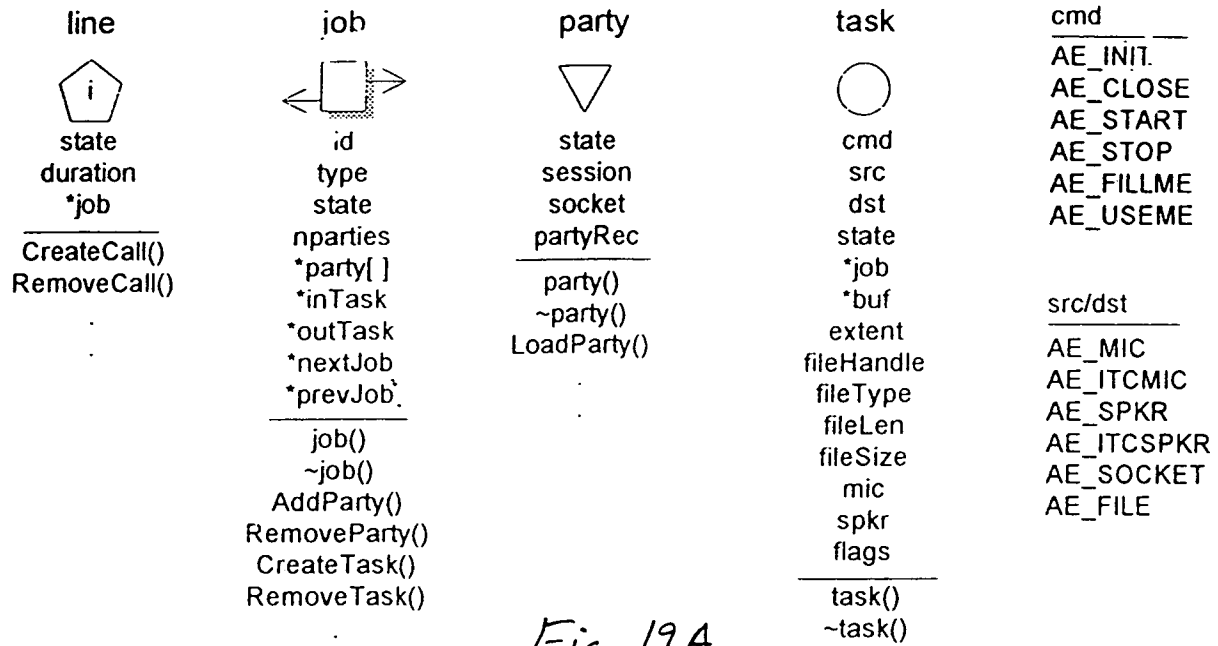


Fig. 19A

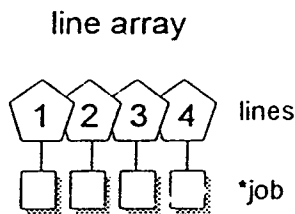


Fig. 19B

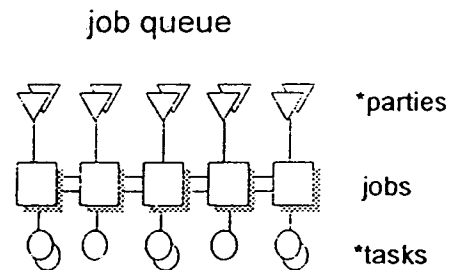


Fig. 19C

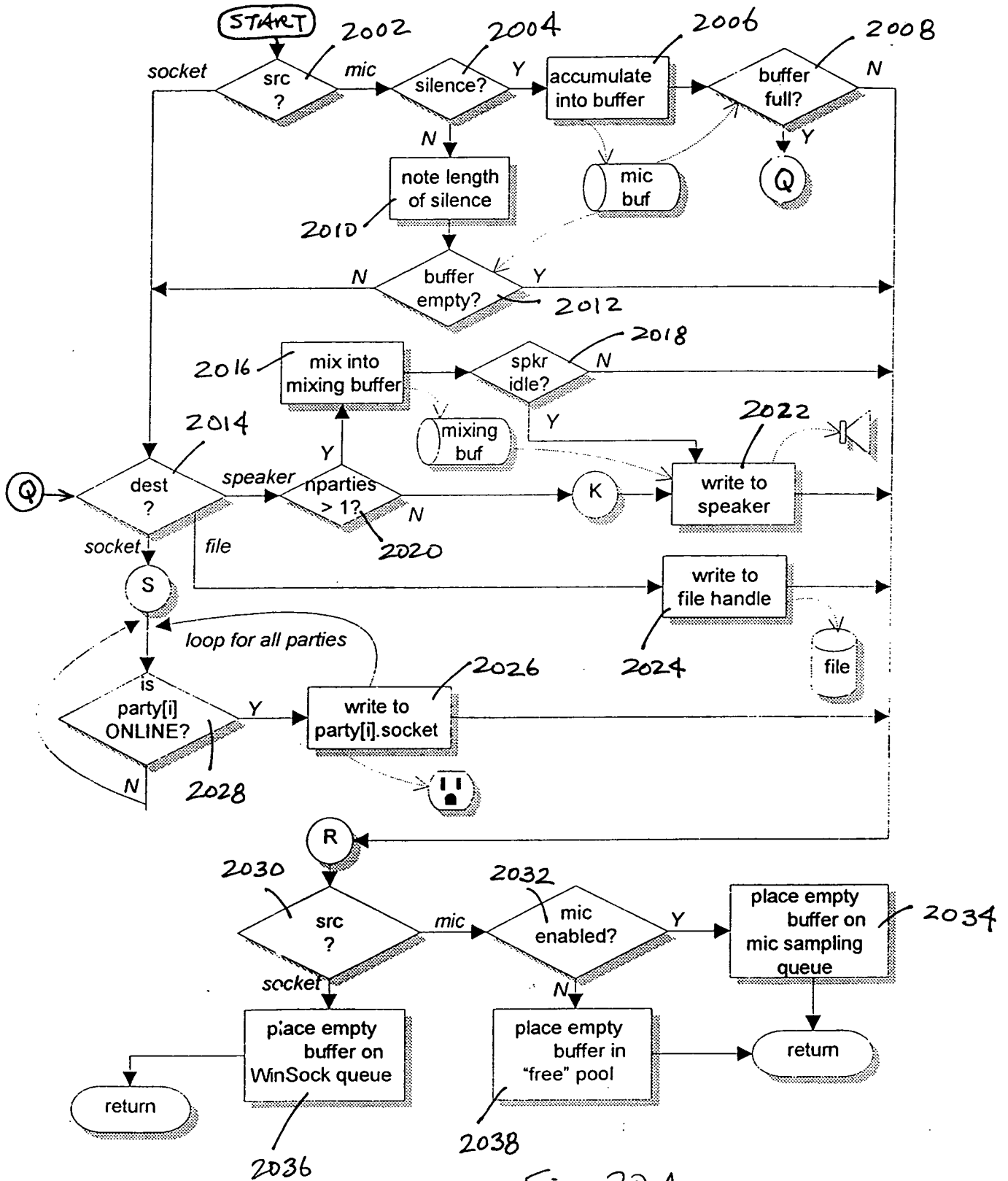


Fig. 20A

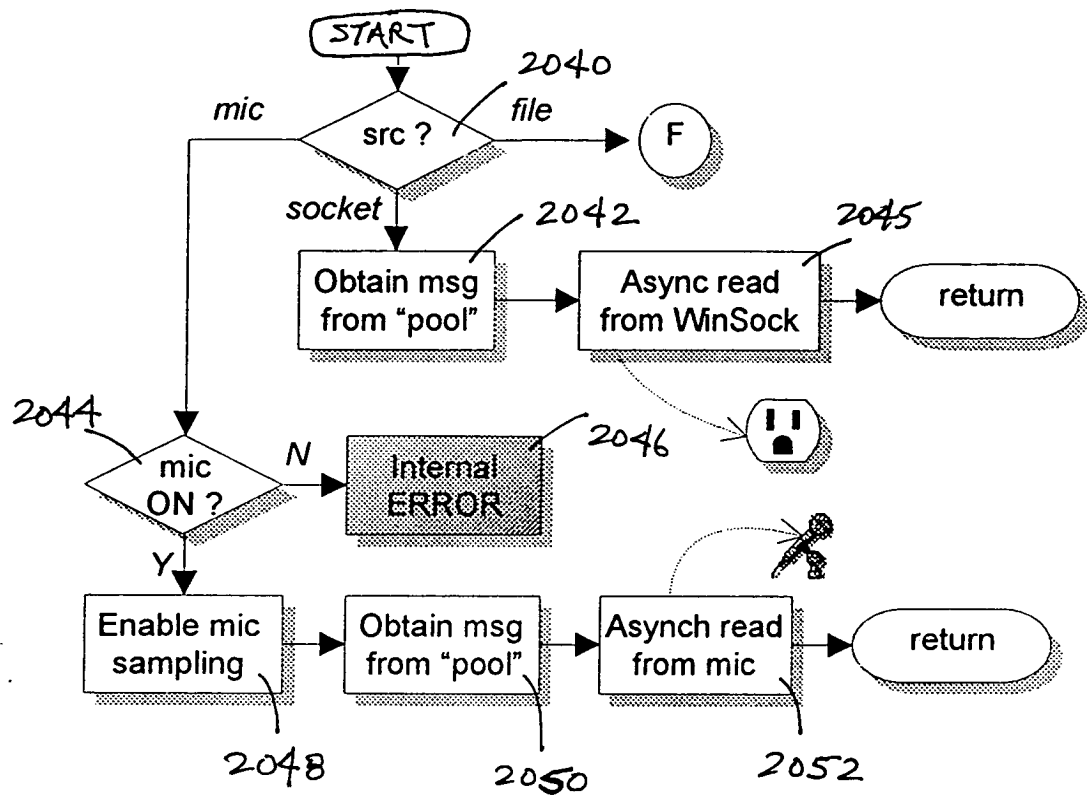


Fig. 20B

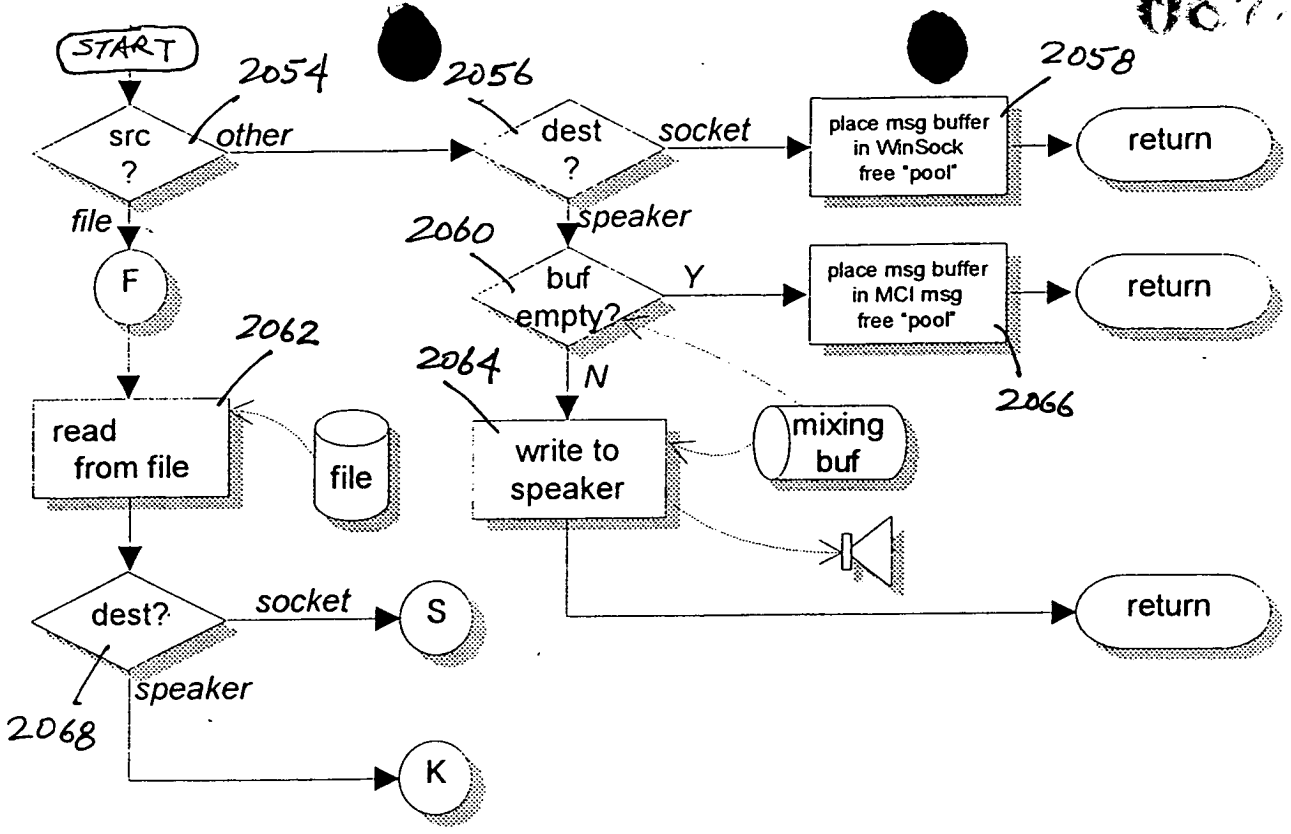


Fig. 20C

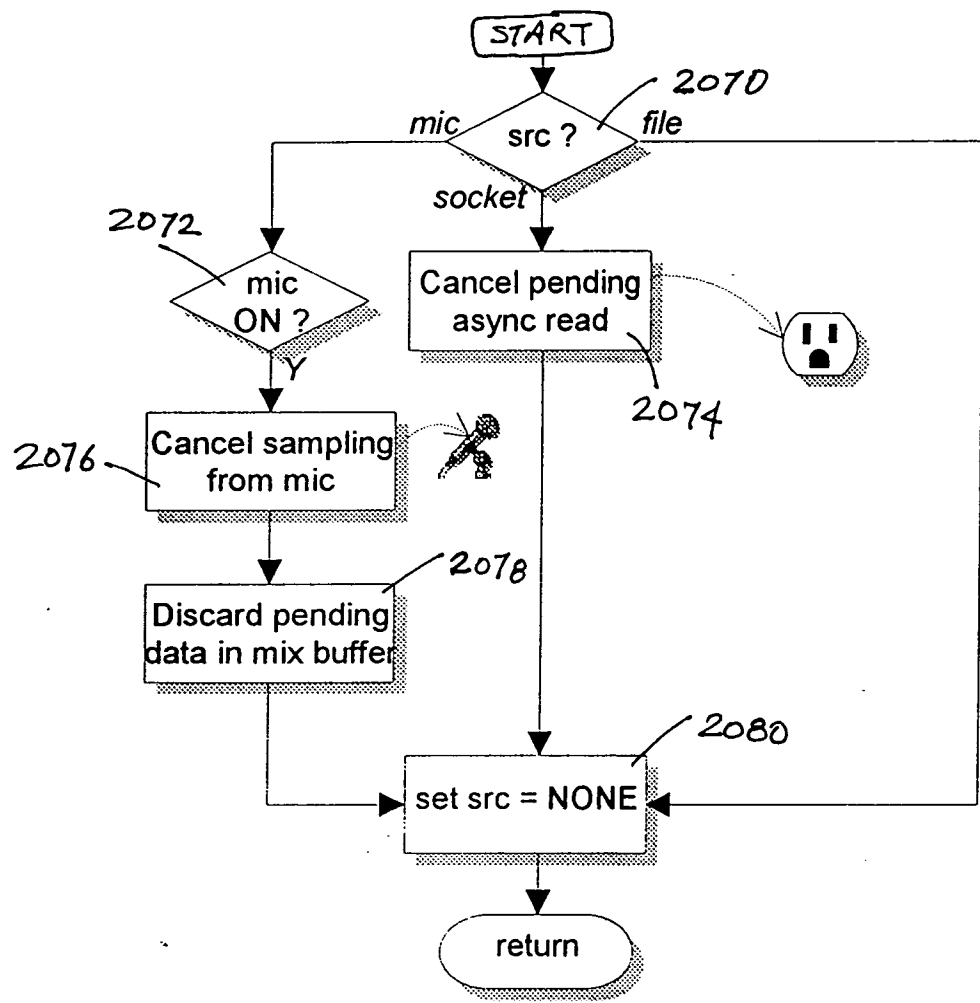
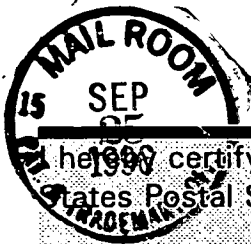


Fig. 20D



I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail in an envelope addressed to:

ASSISTANT COMMISSIONER FOR PATENTS AND
TRADEMARKS, WASHINGTON, DC 20231

and bearing Label Number EM316008331US


Bruce D. Jobse

PATENT

Inventors: Shane D. Mattaway, Glenn W. Hutton, and Craig B. Strickland

GRAPHIC USER INTERFACE FOR INTERNET TELEPHONY APPLICATION



RELATED APPLICATIONS

This application is a continuation-in-part of United States patent application serial number 08/533,115 ^(Pending) entitled Point-to-Point Internet Protocol, by Glenn W. Hutton, filed September 25, 1995, commonly assigned, the subject matter of which is incorporated herein by reference. *the applicant also claims the benefit of provisional application 60/025,415, 60/024,251, and Provisional 60/024,251.*

20
9/21/99

To the extent that any matter contained herein is not already disclosed in the above-identified parent application, this application claims priority to United States provisional patent application, ~~XXXXX,XXX~~ ^{60/025,415 08/523,110} entitled Internet Telephony Apparatus and Method by Mattaway et al., filed September 4, 1996, and United States provisional patent application serial number, ~~XXXXX,XXX~~ ^{60/024,251} entitled System and Methods for Point-To-Point Communications Over a Computer Network, by Mattaway et al., filed August 21, 1996.

a
a

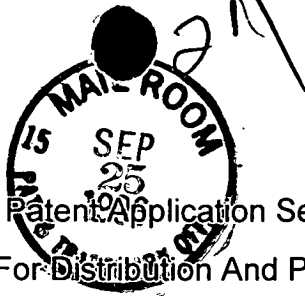
In addition, this application is one of a number of related applications filed on an even date herewith and commonly assigned, the subject matters of which are incorporated herein by reference, including the following:

U.S. Patent Application Serial No. ~~XXXXX,XXX~~ ^{08/719,894} entitled Directory Server For Providing Dynamically Assigned Network Protocol Addresses, by Mattaway et al.;

U.S. Patent Application Serial No. ~~XXXXX,XXX~~ ^{08/719,554} entitled Point-to-point Computer Network Communication Utility Utilizing Dynamically Assigned Network Protocol Addresses, by Mattaway et al.;

U.S. Patent Application Serial No. ~~XXXXX,XXX~~ ^{08/719,640} entitled Method And Apparatus For Dynamically Defining Data Communication Utilities, by Mattaway et al.;

a



75-101

PATENT N0003/7002

A

08/721316

08/719891

U.S. Patent Application Serial No. ~~XXXXXX,XXX~~, entitled Method And Apparatus For Distribution And Presentation Of Multimedia Data Over A Computer Network, by Mattaway et al.;

08/719898

U.S. Patent Application Serial No. ~~XXXXXX,XXX~~, entitled Method And Apparatus For Providing Caller Identification Based Out-going Messages In A Computer Telephony Environment, by Mattaway et al.;

08/718,911

U.S. Patent Application Serial No. ~~XXXXXX,XXX~~, entitled Method And Apparatus For Providing Caller Identification Based Call Blocking In A Computer Telephony Environment, by Mattaway et al.; and

08/719,639

U.S. Patent Application Serial No. ~~XXXXXX,XXX~~, entitled Method And Apparatus For Providing Caller Identification Responses In A Computer Telephony Environment, by Mattaway et al.

FIELD OF THE INVENTION

The present invention relates, in general, to data processing systems, and more specifically, to a method and apparatus for facilitating audio communications over computer networks.

BACKGROUND OF THE INVENTION

The increased popularity of on-line services such as AMERICA ONLINE™, COMPUSERVE®, and other services such as Internet gateways have spurred applications to provide multimedia, including video and voice clips, to online users. An example of an online voice clip application is VOICE E-MAIL FOR WINCIM and VOICE E-MAIL FOR AMERICA ONLINE™, available from Bonzi Software, as described in "Simple Utilities Send Voice E-Mail Online", MULTIMEDIA WORLD, VOL. 2, NO. 9,

3

August 1995, p. 52. Using such Voice E-Mail software, a user may create an audio message to be sent to a predetermined E-mail address specified by the user.

Generally, devices interfacing to the Internet and other online services may communicate with each other upon establishing respective device addresses. One type of device address is the Internet Protocol (IP) address, which acts as a pointer to the device associated with the IP address. A typical device may have a Serial Line Internet Protocol or Point-to-Point Protocol (SLIP/PPP) account with a permanent IP address for receiving E-mail, voicemail, and the like over the Internet. E-mail and voicemail is generally intended to convey text, audio, etc., with any routing information such as an IP address and routing headers generally being considered an artifact of the communication, or even gibberish to the recipient.

Devices such as a host computer or server of a company may include multiple modems for connection of users to the Internet, with a temporary IP address allocated to each user. For example, the host computer may have a general IP address "XXX.XXX.XXX," and each user may be allocated a successive IP address of XXX.XXX.XXX.10, XXX.XXX.XXX.11, XXX.XXX.XXX.12, etc. Such temporary IP addresses may be reassigned or recycled to the users, for example, as each user is successively connected to an outside party. For example, a host computer of a company may support a maximum of 254 IP addresses which are pooled and shared between devices connected to the host computer.

Permanent IP addresses of users and devices accessing the Internet readily support point-to-point communications of voice and video signals over the Internet. For example, real-time video teleconferencing has been implemented using dedicated IP addresses and mechanisms known as reflectors. Due to the dynamic nature of temporary IP addresses of some devices accessing the Internet, point-to-point communications in real-time of voice and video have been generally difficult to attain.

The ability to locate users having temporary or dynamically assigned Internet Protocol address has been difficult without the user manually initiating the communication. Accordingly, spontaneous, real-time communications with such users over computer networks have been impractical. Further, it is desirable to have a communication utility which contains familiar features and functions to current communication utility such as telephones and cellular telephones. It is even further desirable to utilize the current graphic user interface technology associated with computer software in a manner to achieve a more flexible interface to a such a communication utility, without the limitations associated with hardware.

Accordingly, a need exists for a way to determine whether computer users are actively connected to a computer network.

A further need exists for a way to obtain the dynamically assigned Internet Protocol address of a user having on-line status with respect to a computer network, particularly the Internet.

An even further need exists for a method and apparatus by which to establish real-time, point-to-point communications over a computer network using a communication utility having an interface which combines the familiar aspects of current hardware communication utilities but which allows for the flexibility associated with graphic user interfaces.

SUMMARY OF THE INVENTION

The above deficiencies in the prior art and previously described needs are fulfilled by the present invention which provides a virtual communications utility displayable on computer system interfaces which enables real-time, point-to-point communications over computer networks. According to one embodiment of the present invention, a computer program product for use with a computer system having a display

and an audio transducer comprises a computer usable medium having computer readable code means embodied therein comprising program code means for generating a user interface, program code means responsive to user input commands for establishing a point-to-point communication link with another computer over a network and program code means responsive to audio data from the audio transducer for transmitting the audio data over the communication link.

According to another embodiment of the present invention, a computer program product for use with a computer system comprises a computer usable medium having computer readable program code means embodied thereon comprising code means for transmitting from a client process to a server a query as to whether a second client process is connected to the computer network, program code means for receiving the network protocol address of the second process from the server, and program code means responsive to the network protocol address of the second client process for establishing a point-to-point communication link between the first client process and the second client process.

BRIEF DESCRIPTION OF THE DRAWINGS

The features of the invention will become more readily apparent and may be better understood by referring to the following detailed description of an illustrative embodiment of the present invention, taken in conjunction with the accompanying drawings, in which:

Fig. 1 illustrates, in block diagram format, a system for the disclosed point-to-point Internet protocol;

Fig. 2 illustrates, in block diagram format, the system using a secondary point-to-point Internet protocol;

Fig. 3 illustrates, in block diagram format, the system of FIGS. 1-2 with the point-to-point Internet protocol established;

Fig. 4 is another block diagram of the system of FIGS 1-2 with audio communications being conducted;

Fig. 5 illustrates a display screen for a processing unit;

Fig. 6 illustrates another display screen for a processing unit;

Fig. 7 illustrates a flowchart of the initiation of the point-to-point Internet protocols;

Fig. 8 illustrates a flowchart of the performance of the primary point-to-point Internet protocols;

Fig. 9 illustrates a flowchart of the performance of the secondary point-to-point Internet protocol;

Fig. 10 illustrates schematically a computer network over which the present invention may be utilized;

Fig. 11 is a block diagram of a computer system suitable for use with the present invention;

Fig. 12 is a block diagram of an audio processing card suitable for use with the computer system of FIG. 10;

Fig. 13 A-B are schematic block diagrams of the elements comprising the inventive computer network telephony mechanism of the present invention;

Fig. 14 is a screen capture illustrating an exemplary user interface of the present invention;

Fig. 15 is a schematic diagram illustrating the architecture of the connection server apparatus suitable for use with the present invention;

Fig. 16A is a flowchart illustrating the process steps performed by the connection server in accordance with the present invention;

Fig. 16B is a flowchart illustrating the process steps performed in accordance with the information server of the present invention;

Figs. 17A-B are schematic block diagrams illustrating of the packet transfer sequence in accordance with the communication protocol of the present invention;

Fig. 18A-D are conceptual block diagrams illustrating user interface and graphic user interface objects utilized by the communication utility of the present invention;

Fig. 19A-C are conceptual block diagrams illustrating the event manager and media engine objects utilized by the communication utility of the present invention;
and

Figs. 20A-D illustrate process steps performed by the media engine function of the communication utility in accordance with the present invention.

DETAILED DESCRIPTION

Referring now in specific detail to the drawings, with like reference numerals identifying similar or identical elements, as shown in FIG. 1, the present disclosure describes a point-to-point network protocol and system 10 for using such a protocol.

In an exemplary embodiment, the system 10 includes a first processing unit 12 for sending at least a voice signal from a first user to a second user. The first processing unit 12 includes a processor 14, a memory 16, an input device 18, and an output device 20. The output device 20 includes at least one modem capable of, for example, 14.4 Kilobit-per-second communications and operatively connected via wired and/or wireless communication connections to the Internet or other computer networks such as an Intranet, i.e., a private computer network. One skilled in the art would understand that the input device 18 may be implemented at least in part by the modem of the output device 20 to allow input signals from the communication connections to be received. The second processing unit 22 may have a processor, memory, and input

and output devices, including at least one modem and associated communication connections, as described above for the first processing unit 12. In an exemplary embodiment, each of the processing units 12, 22 may execute the WEBPHONE® Internet telephony application available from NetSpeak Corporation, Boca Raton, FL, which is capable of performing the disclosed point-to-point Internet protocol and system 10, as described herein.

The first processing unit 12 and the second processing unit 22 are operatively connected to the Internet 24 by communication devices and software known in the art, such as an Internet Service Provider (ISP) or an Internet gateway. The processing units 12, 22 may be operatively interconnected through the Internet 24 to a connection server 26, and may also be operatively connected to a mail server 28 associated with the Internet 24.

The connection server 26 includes a processor 30, a timer 32 for generating time stamps, and a memory such as a database 34 for storing, for example, E-mail and Internet Protocol (IP) addresses of logged-in units. In an exemplary embodiment, the connection server 26 may be a SPARC 5 server or a SPARC 20 server, available from SUN MICROSYSTEMS, INC., Mountain View, CA, having a central processing unit (CPU) as processor 30, an operating system (OS) such as UNIX, for providing timing operations such as maintaining the timer 32, a hard drive or fixed drive, as well as dynamic random access memory (DRAM) for storing the database 34, and a keyboard and display and/or other input and output devices (not shown in FIG. 1). The database 34 may be an SQL database available from ORACLE or INFORMIX.

In an exemplary embodiment, the mail server 28 may be implemented with a Post Office Protocol (POP) Version 3 mail server and the Simple Mail Transfer Protocol (SMTP), including a processor, memory, and stored programs operating in a UNIX

environment, or, alternatively, another OS, to process E-mail capabilities between processing units and devices over the Internet 24.

In the illustrative embodiment, the POP protocol is utilized to retrieve E-mail messages from mail server 28 while the SMTP protocol is used to submit E-mail message to Internet 24.

The first processing unit 12 may operate the disclosed point-to-point Internet protocol by a computer program described hereinbelow in conjunction with FIG. 6, which may be implemented from compiled and /or interpreted source code in the C++ programming language and which may be downloaded to the first processing unit 12 from an external computer. The operating computer program may be stored in the memory 16, which may include about 8 MB RAM and/or a hard or fixed drive having about 8 MB of available memory. Alternatively, the source code may be implemented in the first processing unit 12 as firmware, as an erasable read only memory (EPROM), etc. It is understood that one skilled in the art would be able to use programming languages other than C++ to implement the disclosed point-to-point network protocol and system 10.

The processor 14 receives input commands and data from a first user associated with the first processing unit 12 though the input device 18, which may be an input port connected by a wired, optical, or a wireless connection for electromagnetic transmissions, or alternatively may be transferable storage media, such as floppy disks, magnetic tapes, compact disks, or other storage media including the input data from the first user.

The input device 18 may include a user interface (not shown) having, for example, at least one button actuated by the user to input commands to select from a plurality of operating modes to operate the first processing unit 12. In alternative embodiments, the input device 18 may include a keyboard, a mouse, a touch screen,

and/or a data reading device such as a disk drive for receiving the input data from input data files stored in storage media such as a floppy disk or, for example, an 8 mm storage tape. The input device 18 may alternatively include connections to other computer systems to receive the input commands and data therefrom.

The first processing unit 12 may include a visual interface for use in conjunction with the input device 18 and output device 20 similar to those screens illustrated in FIGS. 5-6, discussed below. It is also understood that alternative devices may be used to receive commands and data from the user, such as keyboards, mouse devices, and graphical user interfaces (GUI) such as WINDOWS™ 3.1 available from MICROSOFT Corporation, Redmond, WA., and other operating systems and GUIs, such as OS/2 and OS/2 WARP, available from IBM CORPORATION, Boca Raton, FL. Processing unit 12 may also include microphones and/or telephone handsets for receiving audio voice data and commands, speech or voice recognition devices, dual tone multi-frequency (DTMF) based devices, and/or software known in the art to accept voice data and commands and to operate the first processing unit 12.

In addition, either of the first processing unit 12 and the second processing unit 22 may be implemented in a personal digital assistant (PDA) providing modem and E-mail capabilities and Internet access, with the PDA providing the input/output screens for mouse interactions or for touchscreen activation as shown, for example, in FIGS. 5-6, as a combination of the input device 18 and output device 20.

For clarity of explanation, the illustrative embodiment of the disclosed point-to-point Internet protocol and system 10 is presented as having individual functional blocks, which may include functional blocks labeled as "processor" and "processing unit". The functions represented by these blocks may be provided through the use of either shared or dedicated hardware, including, but not limited to, hardware capable of executing software. For example, the functions of each of the processors and

processing units presented herein may be provided by a shared processor or by a plurality of individual processors. Moreover, the use of the functional blocks with accompanying labels herein is not to be construed to refer exclusively to hardware capable of executing software. Illustrative embodiments may include digital signal processor (DSP) hardware, such as the AT&T DSP16 or DSP32C, read-only memory (ROM) for storing software performing the operations discussed below, and random access memory (RAM) for storing DSP results. Very large scale integration (VLSI) hardware embodiments, as well as custom VLSI circuitry in combination with a general purpose DSP circuit, may also be provided. Any and all of these embodiments may be deemed to fall within the meaning of the labels for the functional blocks as used herein.

The processing units 12, 22 are capable of placing calls and connecting to other processing units connected to the Internet 24, for example, via dialup SLIP/PPP lines. In an exemplary embodiment, each processing unit assigns an unsigned long session number, for example, a 32-bit long sequence in a *.ini file for each call. Each call may be assigned a successive session number in sequence, which may be used by the respective processing unit to associate the call with one of the SLIP/PPP lines, to associate a <ConnectOK> response signal with a <Connect Request> signal, and to allow for multiplexing and demultiplexing of inbound and outbound conversations on conference lines, as explained hereinafter.

For callee (or called) processing units with fixed IP addresses, the caller (or calling) processing unit may open a "socket", i.e. a file handle or address indicating where data is to be sent, and transmit a <Call> command to establish communication with the callee utilizing, for example, datagram services such as Internet Standard network layering as well as transport layering, which may include a Transport Control Protocol (TCP) or a User Datagram Protocol (UDP) on top of the IP. Typically, a processing unit having a fixed IP address may maintain at least one open socket and a

called processing unit waits for a <Call> command to assign the open socket to the incoming signal. If all lines are in use, the callee processing unit sends a BUSY signal or message to the caller processing unit. As shown in FIG. 1, the disclosed point-to-point Internet protocol and system 10 operate when a callee processing unit does not have a fixed or predetermined IP address. In the exemplary embodiment and without loss of generality, the first processing unit 12 is the caller processing unit and the second processing unit 22 is the callee processing unit. When either of processing units 12, 22 logs on to the Internet via a dial-up connection, the respective unit is provided a dynamically allocated IP address by an Internet service provider.

Upon the first user initiating the point-to-point Internet protocol when the first user is logged on to the Internet 24, the first processing unit 12 automatically transmits its associated E-mail address and its dynamically allocated IP address to the connection server 26. The connection server 26 then stores these addresses in the database 34 and time stamps the stored addresses using timer 32. The first user operating the first processing unit 12 is thus established in the database 34 as an active on-line party available for communication using the disclosed point-to-point Internet protocol. Similarly, a second user operating the second processing unit 22, upon connection to the Internet 24 through an Internet service provider, is processed by the connection server 26 to be established in the database 34 as an active on-line party.

The connection server 26 may use the time stamps to update the status of each processing unit; for example, after 2 hours, so that the on-line status information stored in the database 34 is relatively current. Other predetermined time periods, such as a default value of 24 hours, may be configured by a systems operator.

The first user with the first processing unit 12 initiates a call using, for example, a Send command and/or a command to speedial an NTH stored number, which may be labeled [SND] and [SPD] [N], respectively, by the input device 18 and/or the output

device 20, such as shown in FIGS. 5-6. In response to either the Send or speedial commands, the first processing unit 12 retrieves from memory 16 a stored E-mail address of the callee corresponding to the NTH stored number. Alternatively, the first user may directly enter the E-mail address of the callee.

The first processing unit 12 then sends a query, including the E-mail address of the callee, to the connection server 26. The connection server 26 then searches the database 34 to determine whether the callee is logged-in by finding any stored information corresponding to the callee's E-mail address indicating that the callee is active and on-line. If the callee is active and on-line, the connection server 26 then performs the primary point-to-point Internet protocol; i.e. the IP address of the callee is retrieved from the database 34 and sent to the first processing unit 12. The first processing unit 12 may then directly establish the point-to-point Internet communications with the callee using the IP address of the callee.

If the callee is not on-line when the connection server 26 determines the callee's status, the connection server 26 sends an OFF-LINE signal or message to the first processing unit 12. The first processing unit 12 may also display a message such as "Called Party Off-Line" to the first user.

When a user logs off or goes off-line from the Internet 24, the connection server 26 updates the status of the user in the database 34; for example, by removing the user's information, or by flagging the user as being off-line. The connection server 26 may be instructed to update the user's information in the database 34 by an off-line message, such as a data packet, sent automatically from the processing unit of the user prior to being disconnected from the connection server 26. Accordingly, an off-line user is effectively disabled from making and/or receiving point-to-point Internet communications.

As shown in FIGS. 2-4, the disclosed secondary point-to-point Internet protocol may be used as an alternative to the primary point-to-point Internet protocol described above, for example, if the connection server 26 is non-responsive, unreachable, inoperative, and/or unable to perform the primary point-to-point Internet protocol, as a non-responsive condition. Alternatively, the disclosed secondary point-to-point Internet protocol may be used independent of the primary point-to-point Internet protocol. In the disclosed secondary point-to-point Internet protocol, the first processing unit 12 sends a <ConnectReq> message via E-mail over the Internet 24 to the mail server 28. The E-mail including the <ConnectReq> message may have, for example, the subject

[*wp#XXXXXXXXX#nnn.nnn.nnn.#emailAddr]

where nnn.nnn.nnn.nnn. is the current (i.e. temporary or permanent) IP address of the first user, and XXXXXXXX is a session number, which may be unique and associated with the request of the first user to initiate point-to-point communication with the second user.

The following E-mail messages are transmitted to a remote users post office protocol server via simple mail transport protocol using MIME by the event manager, as explained hereinafter.

<ConnectRequest>

<CampRequest>

<VoiceMail>

<FileTransfer>

<E-mail>

The following E-mail messages are received from a local WebPhone users POP server via the POP protocol using MIME by the event manager, as explained hereinafter.

<Connect Request>
<Camp Request>
<Voice Mail>
<File Transfer>
<E-mail>
<Registration>

As described above, the first processing unit 12 may send the <ConnectReq> message in response to an unsuccessful attempt to perform the primary point-to-point Internet protocol. Alternatively, the first processing unit 12 may send the <ConnectReq> message in response to the first user initiating a SEND command or the like.

After the <ConnectRequest> message via E-mail is sent, the first processing unit 12 opens a socket and waits to detect a response from the second processing unit 22. A timeout timer, such as timer 32, may be set by the first processing unit 12, in a manner known in the art, to wait for a predetermined duration to receive a <ConnectOK> signal. The processor 14 of the first processing unit 12 may cause the output device 20 to output a Ring signal to the user, such as an audible ringing sound, about every 3 seconds. For example, the processor 14 may output a *.wav file, which may be labeled RING.WAV, which is processed by the output device 20 to output an audible ringing sound.

Second processing unit 22 polls mail server 28 at an interval, for example, once a minute, to check for incoming E-mail. Generally, second processing unit 22 checks the messages stored on mail server 28 at regular intervals to wait for and detect incoming E-mail indicating a <CONNECT REQ> message from first processing unit 12.

Typically, for sending E-mail to user's having associated processing units operatively connected to a host computer or server operating an Internet gateway, E-mail for a specific user may be sent over Internet 24 and directed to the permanent IP address of the mail server providing the target user's mail services. The E-mail is transported by a standard protocol, for example, SMTP, and stored into memory (not shown in Fig. 1) associated with mail server 28.

The E-mail may subsequently be retrieved by processing unit 22 on behalf of the user with another standard protocol, for example POP 3. The actual IP address utilized by the user's processing unit is immaterial to the retrieval of E-mail, as the mail server 28 can, for example, be polled or queried from any point on the network.

Upon receiving the incoming E-mail signal from the first processing unit 12, the second processing unit 22 may assign or may be assigned a temporary IP address. Therefore, the delivery of the E-mail through the Internet 24 provides the second processing unit 22 with a session number as well as IP addresses of both the first processing unit 12 and the second processing unit 22.

Point-to-point communication may then be established by the processing unit 22 processing the E-mail signal to extract the <ConnectRequest> message, including the IP address of the first processing unit 12 and the session number. The second processing unit 22 may then open a socket and generate a <ConnectOK> response signal, which includes the temporary IP address of the second processing unit 22 as well as the session number of the first processing unit.

The second processing unit 22 sends the <ConnectOK> signal directly over the Internet 24 to the IP address of the first processing unit 12 without processing by the mail server 28, and a timeout timer of the second processing unit 22 may be set to wait and detect a <Call> signal expected from the first processing unit 12.

Real-time point-to-point communication of audio signals over the Internet 24, as well as video and voicemail, may thus be established and supported without requiring permanent IP addresses to be assigned to either of the users or processing units 12, 22. For the duration of the realtime point-to-point link, the relative permanence of the current IP addresses of the processing units 12, 22 is sufficient, whether the current IP addresses were permanent (i.e. predetermined or preassigned) or temporary (i.e. assigned upon initiation of the point-to-point communication).

In the exemplary embodiment, a first user operating the first processing unit 12 is not required to be notified by the first processing unit 12 that an E-mail is being generated and sent to establish the point-to-point link with the second user at the second processing unit 22. Similarly, the second user is not required to be notified by the second processing unit 22 that an E-mail has been received and/or a temporary IP address is associated with the second processing unit 22. The processing units 12, 22 may perform the disclosed point-to-point Internet protocol automatically upon initiation of the point-to-point communication command by the first user without displaying the E-mail interactions to either user. Accordingly, the disclosed point-to-point Internet protocol may be transparent to the users. Alternatively, either of the first and second users may receive, for example, a brief message of "CONNECTION IN PROGRESS" or the like on a display of the respective output device of the processing units 12, 22.

After the initiation of either the primary or the secondary point-to-point Internet protocols described above in conjunction with FIGS. 1-2, the point-to-point communication link over the Internet 24 may be established as shown in FIGS. 3-4 in a

manner known in the art. For example, referring to FIG. 3, upon receiving the <ConnectOK> signal from the second processing unit 22, the first processing unit 12 extracts the IP address of the second processing unit 22 and the session number, and the session number sent from the second processing unit 22 is then checked with the session number originally sent from the first processing unit 12 in the <ConnectReq> message as E-mail. If the session numbers sent and received by the processing unit 12 match, then the first processing unit 12 sends a <Call> signal directly over the Internet 24 to the second processing unit 22; i.e. using the IP address of the second processing unit 22 provided to the first processing unit 12 in the <ConnectOK> signal.

Upon receiving the <Call> signal, the second processing unit 22 may then begin a ring sequence, for example, by indicating or annunciating to the second user that an incoming call is being received. For example, the word "CALL" may be displayed on the output device of the second processing unit 22. The second user may then activate the second processing unit 22 to receive the incoming call.

Referring to FIG. 4, after the second processing unit 22 receives the incoming call, realtime audio and/or video conversations may be conducted in a manner known in the art between the first and second users through the Internet 24, for example, by compressed digital audio signals. Each of the processing units 12, 22 also display to each respective user the words "IN USE" to indicate that the point-to-point communication link is established and audio or video signals are being transmitted.

In addition, either user may terminate the point-to-point communication link by, for example, activating a termination command, such as by activating an [END] button or icon on a respective processing unit, causing the respective processing unit to send an <End> signal which causes both processing units to terminate the respective sockets, as well as to perform other cleanup commands and functions known in the art.

FIGS. 5-6 illustrate examples of display screens 36 which may be output by a respective output device of each processing unit 12, 22 of FIGS. 1-4 for providing the disclosed point-to-point Internet protocol and system 10. Such display screens may be displayed on a display of a personal computer (PC) or a PDA in a manner known in the art.

As shown in FIG. 5, a first display screen 36 includes a status area 38 for indicating, for example, a called user by name and/or by IP address or telephone number; a current function such as C2; a current time; a current operating status such as "IN USE", and other control icons such as a down arrow icon 40 for scrolling down a list of parties on a current conference line. The operating status may include such annunciators as "IN USE," "IDLE," "BUSY," "NO ANSWER," "OFFLINE," "CALL," "DIALING," "MESSAGES," and "SPEEDDIAL."

Other areas of the display screen 36 may include activation areas or icons for actuating commands or entering data. For example, the display screen 36 may include a set of icons 42 arranged in columns and rows including digits 0-9 and commands such as END, SND, HLD, etc. For example, the END and SND commands may be initiated as described above, and the HLD icon 44 may be actuated to place a current line on hold. Such icons may also be configured to substantially simulate a telephone handset or a cellular telephone interface to facilitate ease of use, as well as to simulate function keys of a keyboard. For example, icons labeled L1-L4 may be mapped to function keys F1-F4 on standard PC keyboards, and icons C1-C3 may be mapped to perform as combinations of function keys, such as CTRL-F1, CTRL-F2, and CTRL-F3, respectively. In addition, the icons labeled L1-L4 and C1-C3 may include circular regions which may simulate lamps or light emitting diodes (LEDs) which indicate that the function or element represented by the respective icon is active or being performed.

Icons L1-L4 may represent each of 4 lines available to the caller, and icons C1-C3 may represent conference calls using at least one line to connect, for example, two or more parties in a conference call. The icons L1-L4 and C1-C3 may indicate the activity of each respective line or conference line. For example, as illustrated in FIG. 5, icons L1-L2 may have lightly shaded or colored circles, such as a green circle, indicating that each of lines 1 and 2 are in use, while icons L3-L4 may have darkly shaded or color circles, such as a red or black circle, indicating that each of lines 3 and 4 are not in use. Similarly, the lightly shaded circle of the icon labeled C2 indicates that the function corresponding to C2 is active, as additionally indicated in the status area 38, while darkly shaded circles of icons labeled C1 and C3 indicate that such corresponding functions are not active.

The icons 42 are used in conjunction with the status area 38. For example, using a mouse for input, a line that is in use, as indicated by the lightly colored circle of the icon, may be activated to indicate a party's name by clicking a right mouse button for 5 seconds until another mouse click is actuated or the [ESC] key or icon is actuated. Thus, the user may switch between multiple calls in progress on respective lines.

Using the icons as well as an input device such as a mouse, a user may enter the name or alias or IP address, if known, of a party to be called by either manually entering the name, by using the speedial feature, or by double clicking on an entry in a directory stored in the memory, such as the memory 16 of the first processing unit 12, where the directory entries may be scrolled using the status area 38 and the down arrow icon 40.

Once a called party is listed in the status area 38 as being active on a line, the user may transfer the called party to another line or a conference line by clicking and dragging the status area 38, which is represented by a reduced icon 46. Dragging the reduced icon 46 to any one of line icons L1-L4 transfers the called party in use to the

selected line, and dragging the reduced icon 46 to any one of conference line icons C1-C3 adds the called party to the selected conference call.

Other features may be supported, such as icons 48-52, where icon 48 corresponds to, for example, an ALT-X command to exit the communication facility of a processing unit, and icon 50 corresponds to, for example, an ALT-M command to minimize or maximize the display screen 36 by the output device of the processing unit. Icon 52 corresponds to an OPEN command, which may, for example, correspond to pressing the O key on a keyboard, to expand or contract the display screen 36 to represent the opening and closing of a cellular telephone. An "opened" configuration is shown in FIG. 5, and a "closed" configuration is shown in FIG. 6. In the "opened" configuration, additional features such as output volume (VOL) controls, input microphone (MIC) controls, waveform (WAV) sound controls, etc.

The use of display screens such as those shown in FIGS. 5-6 provided flexibility in implementing various features available to the user. It is to be understood that additional features such as those known in the art may be supported by the processing units 12, 22.

Alternatively, it is to be understood that one skilled in the art may implement the processing units 12, 22 to have the features of the display screens in FIGS. 5-6 in hardware; i.e. a wired telephone or wireless cellular telephone may include various keys, LEDs, liquid crystal displays (LCDs), and touchscreen actuators corresponding to the icons and features shown in FIGS. 5-6. In addition, a PC may have the keys of a keyboard and mouse mapped to the icons and features shown in FIGS. 5-6.

Referring to FIG. 7, the disclosed point-to-point Internet protocol and system 10 is illustrated. First processing unit 12 initiates the point-to-point Internet protocol in step 56 by sending a query from the first processing unit 12 to the connection server 26. If connection server 26 is operative to perform the point-to-point Internet protocol, in step

58, first processing unit 12 receives an on-line status signal from the connection server 26, such signal may include the IP address of the callee or a "Callee Off-Line" message. Next, first processing unit 12 performs the primary point-to-point Internet protocol in step 60, which may include receiving, at the first processing unit 12, the IP address of the callee if the callee is active and on-line. Alternatively, processing unit 60 may initiate and perform the secondary point-to-point Internet protocol in step 62, if connection server 26 is not operable.

Referring to FIG. 8, in conjunction with FIGS. 1 and 3-4, the disclosed point-to-point Internet protocol and system 10 are illustrated. Connection server 26 starts the primary point-to-point Internet protocol, in step 64, and timestamps and stores E-mail and IP addresses of logged-in users and processing units in the database 34 in step 66. Connection server 26 receives a query from a first processing unit 12 in step 68 to determine whether a second user or second processing unit 22 is logged-in to the Internet 24, with the second user being specified, for example, by an E-mail address. Connection server 26 retrieves the IP address of the specified user from the database 34 in step 70, if the specified user is logged-in to the Internet, and sends the retrieved IP address to the first processing unit 12 in step 72 to enable first processing unit 12 to establish point-to-point communications with the specified second user.

The disclosed secondary point-to-point Internet protocol operates as shown in FIG. 9. First processing unit 12 generates an E-mail signal, including a session number and a first IP address corresponding to a first processing unit in step 76. First processing unit 12 transmits the E-mail signal as a <ConnectRequest> signal to the Internet 24 in step 78. The E-mail signal is delivered through the Internet 24 using a mail server 28 to the second processing unit 22 in step 80. Second processing unit 22 extracts the session number and the first IP address from the E-mail signal in step 82 and transmits or sends the session number and a second IP address corresponding to

the second processing unit 22, back to the first processing unit 12 through the Internet 24, in step 84. First processing unit 12 verifies the session number received from the second processing unit 22 in step 86, and establishes a point-to-point Internet communication link between the first processing unit 12 and second processing unit 22 using the first and second IP addresses in step 88.

The primary and secondary point-to-point Internet protocols previously described enable users to establish real-time direct communication links over the Internet or other computer networks without the need for any interaction with connection server 26, the connection server providing only directory and information related services.

Fig. 10 illustrates an exemplary computer network 1000 over which the invention may operate. A first processing unit 1012 is coupled to a computer network, illustrated here as the Internet 1010, through an Internet service provider 1014. Similarly, a second processing unit 1022 is coupled to Internet 1010 through Internet service provider 1018. The inventive directory server 1020 is similarly coupled to Internet 1010 through Internet service provider 1026. Directory server 1020 further comprises a connection server 1022 and information server 1024, as will be explained hereinafter. The first processing unit 1012, second processing unit 1022 and directory server 1020 are operatively coupled to each other via the Internet 1010. It will be obvious to those reasonably skilled in the art that network 1000 is not restricted to implementation over the Internet 1010 but may comprise other network configurations such as a local area network (LAN), a wide area network (WAN), a global area network or any number of private networks currently referred to as an Intranet. Such networks may be implemented with any number of hardware and software components, transmission media and network protocols.

Exemplary Computer Architecture

Fig.11 illustrates the system architecture for a computer system 1100 such as an IBM PS/2®, suitable for implementing first and second processing units 1012 and 1022, respectively, of Fig. 10, as well as global server 1020. The exemplary computer system of Fig.11 is for descriptive purposes only. Although the description may refer to terms commonly used in describing particular computer systems, such as in IBM PS/2 computer, the description and concepts equally apply to other computer systems ranging from personal digital assistants (PDAs) to workstations to mainframe systems.

Computer system 1100 includes a central processing unit (CPU) 1105, which may be implemented with a conventional microprocessor. System 1100 further includes a random access memory (RAM) 1110 for temporary storage of information, and a read only memory (ROM) 1115 for permanent storage of information. A memory controller 1120 is provided for controlling RAM 1110. A bus 1130 interconnects the components of computer system 1100. A bus controller 1125 is provided for controlling bus 1130. An interrupt controller 1135 is used for receiving and processing various interrupt signals from the system components.

Mass storage may be provided by diskette 1142, CD ROM 1147, or hard drive 1152. Data and software may be exchanged with computer system 1100 via removable media such as diskette 1142 and CD ROM 1147. Diskette 1142 is insertable into diskette drive 1141 which is, in turn, connected to bus 1130 by a controller 1140. Similarly, CD ROM 1147 is insertable into CD ROM drive 1146 which is, in turn, connected to bus 1130 by controller 1145. Hard disk 1152 is part of a fixed disk drive 1151 which is connected to bus 1130 by controller 1150.

User input to computer system 100 may be provided by a number of devices. For example, a keyboard 1156 and mouse 1157 are connected to bus 1130 by controller 1155. An audio transducer 1196, which may act as both a microphone and a speaker, is connected to bus 1130 by audio controller 1197, as illustrated. It will be

obvious to those reasonably skilled in the art that other input devices, such as a pen and/or tablet may be connected to bus 1130 with an appropriate controller and software, as required. DMA controller 1160 is provided for performing direct memory access to RAM 1110. A visual display is generated by video controller 1165 which controls video display 1170. Computer system 1100 also includes a communications adaptor 1190 which allows the system to be interconnected to a network such as a local area network (LAN), a wide area network (WAN), or the Internet, schematically illustrated by transmission medium 1191 and network 1195.

In the illustrative embodiment, computer system 1100 may include an Intel microprocessor such as the 80486DX-33 MHz, or faster, a 14.4 Kb communication modem or faster, and a sound card, as further described with reference to Fig. 12.

Operation of computer system 1100 is generally controlled and coordinated by operating system software, such as the OS/2® operating system, available from International Business Machines Corporation, Boca Raton, FL, or Windows® DOS-based operating system available from Microsoft Corp., Redmond, WA. The operating system controls allocation of system resources and performs tasks such as process scheduling, memory management, networking, and I/O services, among other things.

Fig. 12 illustrates schematically an audio sound card 1200 which may be used to implement audio controller 1197 of Fig. 11. Specifically, sound card 1200 may comprise, in the exemplary embodiment, an analog-to-digital (A/D) converter 1212, an input buffer 1216, a digital signal processor (DSP) 1222, ROM 1224, RAM 1226, an output buffer 1220, and an analog-to-digital (D/A) converter 1218, all of which may be interconnected over a bus 1210. Bus 1210 is in turn coupled to a bus interface 1228 which, in turn, is coupled to bus controller 1125 of computer system 1100 of Fig. 11.

As illustrated in Fig. 12, A/D converter 1212 is coupled to audio transducer 1214 which is typically a microphone. Conversely, D/A converter 1218 is coupled to audio

transducer 1230, typically a speaker. It will be obvious to those reasonably skilled in the art that audio transducers 1214 and 1230, may be combined into a single element which serves as both a transmitter and receiver of audio signal.

In operation, A/D converter 1212 samples the audio signals supplied to it by transducer 1214 and stores the digital samples in buffer 1216. The digital sampling occurs under control of a program typically stored in ROM 1224, or, alternatively, under the control of digital signal processor 1222. The digital samples stored in input buffer 1216 are forwarded periodically, typically when the buffer reaches near capacity, over bus 1210 to bus 1130 of Fig. 11, for further processing by computer system 1100. The device driver for audio sound card 1200 generates system interrupts which will cause the digital samples stored in input buffer 1216 to be retrieved for processing. In the exemplary embodiment, the digital samples are uncompressed as supplied to computer system 1100. However, compression of the digital samples may occur using DSP 1222 executing an appropriate compression algorithm, if desired.

Digital audio samples from computer system 1100 are also be converted to analog signals by sound card 1200. The digital samples are supplied to bus 1210 and temporarily stored into output buffer 1220. The digital samples are then converted by D/A converter 1218 into an analog signals which are then supplied to audio transducer 1230, i.e., a speaker, or to further amplification and processing devices.

Sound card 1200 contemplated for use with the present invention may be implemented with any number of Windows compliant sound cards, such as the Sound Blaster sound card, commercially available from Creative Technologies Ltd., Singapore. Such Window compliant sound cards have a Windows compliant software interface allowing a standardized mechanism for software programs to operate the sound card device, such as ^{Winsock} Winsock 1.1.

WebPhone Application

In the exemplary embodiment of the present invention, each of first processing unit 1012 and second processing unit 1022 of Fig. 10 are executing a software application capable of enabling point-to-point communication over network 1000, such as an Internet telephone application. One such application suitable for use with the present invention is the WebPhone Version 1.0 or higher, software, hereafter referred as the "WebPhone," commercially available from NetSpeak Corporation, Boca Raton, FL. A description of the architecture and operation of the WebPhone is provided herein with reference to Figs. 5-6, 13A-B and 14. An extensive detailed description of the architecture, application program interface, graphic user interface, and operation of the WebPhone can be found in copending U.S. patent application serial number ^{08/719,554} ~~XXXXXX~~,
XXX entitled "Point-to-Point Computer Network Communication Utility Utilizing Dynamically Assigned Internet Protocol Addresses" by Mattaway et al. filed on an even date herewith and commonly assigned, the complete subject matter of which is incorporated herein by reference.

Referring to Figs. 13A-B, schematic block diagrams of the WebPhone architecture are illustrated. The WebPhone is an end-user software application which enables users to send real-time audio data to other WebPhone users over the Internet or any public or private TCP/IP based computer networks. The WebPhone application and architecture may be designed to run on any number of operating systems or computer architectures. In the illustrative embodiment, the WebPhone application is implemented as a Windows compatible application executable on an IBM PC architecture or a clone thereof.

Referring to Fig. 13A, the WebPhone 1300 comprises a set of object modules, written in a programming language such as C++, which work together in a concerted fashion to provide real-time, multitasking, network-based media transmission and

reception. WebPhone 1300 comprises a graphic user interface (GUI) 1310, a user interface (UI) 1312, an event manager 1314, a media engine 1316, a database dynamic link library 1318, one or more audio compression/decompression (codecs) 1320, an audio manager 1324, a WebPhone application program interface (API) 1326, and a network interface 1322.

WebPhone GUI 1310 comprises the visual objects seen on a computer display by the user, as illustrated by the screen capture of Fig. 14 discussed hereinafter. WebPhone GUI 1310 serves only to display the artwork associated with the underlying objects of WebPhone UI 1312. WebPhone GUI 1310 may be implemented in a modular fashion distinct from the WebPhone UI for rapid portability. In this manner, other graphic user interface environments such as those compatible with the Macintosh, X-Windows or OS/2 operating systems, may be substituted via the Plug and Play protocol, as would be understood by those reasonably skilled in the arts.

The WebPhone UI 1312 objects maintain the state of the WebPhone GUI and provide feedback to the WebPhone GUI objects from events originating from either the user or the event manager 1314. When WebPhone changes a state that requires user notification, WebPhone UI objects notify associated WebPhone GUI objects to display the appropriate art work to the user. WebPhone UI objects also interface with the database dynamic link library 1318 to maintain the WebPhone database information, e.g. configuration information, phone directory information, etc.

The WebPhone event manager 1314 processes all the events originating from the user, via WebPhone UI 1312, the media engine 1316, and WebPhone API 1326. Event manager 1314 may be implemented as a table-driven state machine that processes the above-identified events and performs the functions necessary to bring the WebPhone from one state to another. For example, event manager 1314 interacts with media engine 1316 to create, control and remove concurrently executing jobs

managed by media engine 1316. Event manager 1314 also interfaces with the WebPhone API 1326 to provide communications with other WebPhones and connection servers, as described in more detail hereinafter. WebPhone database 1318 is a dynamic link library of tree-based subroutines that provide fast database access to the WebPhone configuration information, personal phone directory, etc.

WebPhone media engine 1316 manages the allocation of associated resources to provide a multitasking environment and controls the flow of real-time data streams, e.g., conversations, outgoing messages, etc., and non-real-time data streams, e.g., voice mail, graphic images, files, etc., to and from a user network connection. The objects representing tasks are created by event manager 1314, thereby freeing media engine 1316 to manage resource routing. Specifically, the media engine routes data streams from sources such as a microphone, file or network socket, to destinations such as speaker, destination file or other network socket. To perform such routing functions the media engine interfaces with the WebPhone API 1326 to control communication with other processes, and further communicates with audio manager 1324 to communicate with the system input/output apparatus, such as sound card 1200 of Fig. 12. Media engine 1314 may be designed to employ heuristic methods to sense and efficiently utilize available bandwidth to achieve timely and accurate delivery of all data streams, both real-time and non-real-time.

Media engine 1316 further interacts with WebPhone codec 1320 to achieve compression and decompression of audio data streams. Codec 1320 provides coding of digital samples from the sound card 1200 of Fig. 12 into a compressed format more suitable for transmission over a computer network. Codec 1320 further provides decoding of a compressed signal prior to its submission to sound card 1200 for subsequent conversion to an audible analog signal. In the exemplary embodiment, WebPhone codec 1320 is implemented in a modular fashion so that codecs may be

replaced and updated with newer, more efficient compression/decompression algorithms via the Plug and Play protocol. A codec suitable for use with the present invention is the True Speech codec, version 8.5, commercially available from the DSP Group, Inc., Santa Clara, California. The True Speech codec is an enhanced linear predicative coding algorithm, specifically designed to efficiently encode and decode human speech data. The True Speech codec samples the digital sample stream from sound card 1200, and, using a look-up table-based algorithm, tries to predict the value of the next data sample in the digital data stream based on the history of prior data sample values. The compressed data stream comprises a combination of identifiers of the predicted sample values, as well as error values used to correct the predictive values. Accordingly, the amount of digital data actually transmitted to represent the audio signal is significantly reduced in comparison to transmission of the actual data samples generated by sound card 1200. The True Speech codec provides temporal, frequency domain compression of the digital data representing the audio signal.

Audio manager 1324 handles communication with the audio sound card 1200 and presents a common interface to media engine 1314. Audio manager 1324 interfaces with sound card 1200 through one or more application program interfaces. In the illustrative embodiment, audio manager 1324 utilizes low-level Microsoft Windows wave input/output routines to interface with MCI compliant sound cards. As with codecs 1320, audio manager 1324 may be implemented to adhere to the Plug and Play protocol so other compliant audio sound cards or circuits, such as those for the Apple Macintosh, commercially available from Apple Computer Company, Cupertino, California, or a Unix compatible sound card or circuit may interact with the audio manager 1324.

The WebPhone API 1326 enables the WebPhone to communicate with other WebPhones, connection and directory assistance servers, Internet gateway servers,

credit processing servers, database access servers and other client processes implementing the WebPhone API. As illustrated in Fig. 13B, the WebPhone API utilizes sockets, i.e., a file handle or address indicating where data is to be sent, allowing WebPhone API enabled processes to reside on the same computer, on a local area network, on a wide area network, or over the Internet. A process 1328 communicates with the WebPhone API 1326 through a plurality of sockets 1322. The sockets 1322 are accessible by network 1330 through a number of protocols including Internet Protocol (IP) 1332, Transmission Control Protocol (TCP) 1334, Real-Time Protocol (RTP) 1336 and User Datagram Protocol (UDP) 1338. The WebPhone API provides remote command control of WebPhones and servers via the TCP. WebPhone API 1326 transfers real-time and streamed audio via the UDP protocol and real-time audio and video data via the UDP and RTP protocols. The WebPhone API utilizes TCP to transfer data of different types, i.e., file, image, graphics, etc. as well as to transfer streamline video and other multimedia data types, such as Java developed by Sun Microsystems, Mountain View, CA. In addition, the WebPhone API provides user definable commands and data types.

Fig. 14 illustrates the graphic display produced upon invoking the WebPhone application. Display 1400 is an alternative embodiment to that illustrated in Figs. 5-6 with similar graphic elements, icons and display areas functioning as previously described with reference to Figs. 5-6.

WebPhone Global Server

Having described the architecture of the WebPhone software which enables the first and second processing units to establish point-to-point communication over a network, a discussion of the global connection/information server is appropriate.

Referring to Fig. 15A, a network diagram, similar to that shown in Fig. 10, is illustrated, including a schematic diagram of the global server 1500 and the various devices operatively coupling server 1500 to the Internet 1530. A first processing unit executing the WebPhone application, hereafter referred to as WebPhone 1536, is coupled to Internet 1530 through an Internet service provider 1532. Similarly, a second processing unit executing the WebPhone application, referred to as WebPhone 1538, is coupled to the Internet 1530 by an Internet service provider 1534. Global server 1500 is coupled to Internet 1530 by an Internet service provider 1528, a CSU/DSU 1526, a router 1524, and a fire wall server 1522. In the illustrative embodiment, fire wall server 1522 and global server 1500 are connected through a local area network 1520. Network 1520 may be implemented with an Ethernet or other suitable transport for TCP/IP communications. However, as will be obvious to those recently skilled in the arts, server 1500 may be connected directly to fire wall server 1522.

In the illustrative embodiment, firewall server 1522 is a single firewall mechanism which protects unauthorized access from network 1530 into global server 1500. Firewall server 1522 may be implemented on a work station, such as a SPARC 5 or SPARC 20 server from Sun Microsystems, executing a commercially available firewall software application such as Raptor, available from Raptor Systems. Essentially, the firewall server prevents unauthorized access into global server 1500 and thereby prevents destruction of any of the information contained therein by checking the source of requests for information to global server 1500.

Router 1524 translates logical addresses among networked topologies and may be implemented with any number of commercial router devices such as the CISCO model 2501 router executing CISCO 11.0 software, both commercially available from CISCO Systems, Inc., San Jose, CA.

CSU/DSU 1526 (Channel Send Unit/Data Send Unit) functions as a sophisticated modem, converting network data to high speed serial data for transfer over a T1 or T3 line. Such high speed data is connected to another CSU/DSU, typically at the telephone company over the T1 or T3 line. An apparatus suitable for use in implementing CSU/DSU 1526 in the present invention is the AT&T Paradigm by AT&T Laboratories, Murray Hill, NJ.

Fig. 15A further illustrates a logical schematic of global server 1500. The server comprises a hardware platform 1508 on which an operating system 1510 executes. In the illustrative embodiment, hardware platform 1508 may comprise any number of commercially available high end work stations such as a DEC Alpha 4100 System, commercially available from Digital Equipment Corporation, Maynard, MA, or a SPARC 5 or a SPARC 20, both commercially available from Sun Micro Systems, Mountain View, CA. Operating system 1510, in the illustrative embodiment, may comprise the Unix, commercially available from Novell, Windows NT, commercially available from Microsoft Corporation, or Solaris, commercially available from Sun Microsystems, Inc. Executing on operating system 1510 are a number of processes including connection server 1512, information server 1514, database server 1518 and database 1516.

Connection Server

Connection server 1512 provides a directory information service to WebPhone client processes currently on-line with respect to the computer network. Connection server 1512 behaves like a virtual machine within global server 1500 and interacts with database 1516 through database server 1518 and with network interface card 1540 through the WebPhone API. The basic function of connection server 1512 is to provide a one-to-one mapping between an identifier of a WebPhone client process, such as a

E-mail address, and the current IP address, dynamic or fixed, associated with that WebPhone client process.

As described in further detail hereinafter, when a WebPhone client transmits a <CONNECT REQ> packet to global server 1500, an E-mail address such as "Shane@netspeak.com" is provided to connection server 1512. Connection server 1512 then compares the E-mail address with the values of the records contained in on-line table 1516B and, if a match occurs with one of the records contained therein, transmits the value of the Internet Protocol address associated with that record to the requesting WebPhone client, i.e., a one-to-one matching between E-mail addresses and Internet Protocol addresses.

Referring to Fig. 16A, a flow chart illustrating the basic process steps used by connection server 1512 to implement a one-to-one mapping of E-mail addresses to Internet Protocol addresses in accordance with the present invention is illustrated. The coding of the process steps of the flowchart of Fig. 16A into instructions suitable to control global server 1500 will be understandable by those having ordinary skill in the art of programming. Connection server 1512 remains in an idle state until a <CONNECT REQ> packet is transmitted from a WebPhone client to global server 1500, as illustrated by decisional block 1610 of Fig. 16A. Upon receipt of the packet, connection server 1512 extracts the E-mail address from the packet and supplies the E-mail address to database server 1518 which then communicates using the ODBC standard with database 1516 to perform a search of On-line Table 1516B, as illustrated by process blocks 1612 and 1614. Database 1516 performs a search of on-line Table 1516B and supplies the current Internet Protocol address of the WebPhone client associated with the E-mail address to connection server 1512, via database server 1518. If a corresponding Internet Protocol address is found for the E-mail address contained in the query, connection server 1512 supplies the Internet protocol address

to the requesting WebPhone client by transmitting a <CONNECT ACK> packet, as illustrated by decisional block 1616 and process block 1618. If, however, there is no Internet Protocol address associated with the queried E-mail address or the WebPhone client is off line, connection server 1512 will send an <OFFLINE> packet to the WebPhone client, as illustrated by process block 1622. Connection server 1512 will return to an idle state to await the receipt of another <CONNECT REQ> packet, as illustrated by Fig. 16A. A description of the above described packets as well as a diagram illustrating the packet transfer sequence between a WebPhone client and global server 1500 can be found with reference to Tables 7-8 and Fig. 17A, respectively.

Information Server

Information server 1514 provides an interface between requests from WebPhone client processes and database 1516. Information server 1514 includes code written to extract the search criteria from an <INFO REQ> packet and supply the search criteria to the database search engine of database 1516 using the ODBC standard. In particular, referring to Fig. 16B, a flow chart illustrating the basic process steps used by information server 1514 in performing information/directory service functions in accordance with the present invention is illustrated. The coding of the process steps of the flow chart into instructions suitable for execution by global server 1500 will be understood by those having ordinary skill in the art of programming. Information server 1514 remains idle until an <INFO REQ> packet is received from a WebPhone client process, as illustrated by decisional step 1630. Next, information server 1514 extracts the data elements defined within the <INFO REQ> packet and supplies them to database server 1518 which, in turn, forward them to database 1516, as represented by the process step 1634 and 1636. The search engine contained within database 1516 performs the search and supplies to information server 1514 all client records meeting

the search criteria specified in the <INFO REQ> packet, or a message indicating that no records were found. Next, information server 1514 transmits a <INFO ACK> packet to the WebPhone client process indicating the number of records satisfying the search criteria, as indicated by process step 1638. The WebPhone client may wish to receive all records satisfying the search criteria, or, if the number is excessively large, may desire to further refine the search by transmitting a <INFO ABORT> packet to information server 1514 and defining new search parameters to be sent with a subsequent <INFO REQ> packet. If a <INFO ABORT> packet is received by information server 1514, the process will return to an idle state, as illustrated by decisional block 1640. If no <INFO ABORT> packet was received, information server 1514 will transmit one or more <INFO> packets to the requesting WebPhone client until all records have been received by the WebPhone client, as illustrated by process step 1642. Information server 1514 will return to an idle state awaiting another <INFO REQ> packet, as illustrated in Fig. 16B. A description of the packets comprising the WebPhone protocol is illustrated in Tables 7-8 and a diagram illustrating the packet transfer sequence defined in Fig. 17A-B.

Network interface card 1540 interfaces with connection server 1512, information 1514, and database server 1518 using the WebPhone API definition, as described herein, and the Windows Sockets 1.1 Protocol, or, in a Unix-based operating system, Berkeley Sockets Network API. Network interface card 1514 may comprise, in illustrative embodiment, an Ethernet card capable of transmitting data at rates of 100 Mbps or greater, such cards being commercially available through a number of different vendors.

The connection from CSU/DSU 1526 to ISP 1528 may comprise a T1 connection, i.e., a long-distance, digital, point-to-point communication circuit capable of transmitting a signal at 1.544 Mbps with 24 channels at 64 Kbps. Alternatively, a T3

connection may be used, i.e., a connection is similar to a T1 connection except it is capable of transmitting at 44.746 Mbps per second with up to 28 T1 channels. Other connections may be suitable, depending on specific requirements and availability.

Database

Database 1516 of global server 1500 may be implemented with any of a number of commercially available structured query language (SQL) database engines, such as Oracle 7.x, Informix, or Microsoft SQL server 6.x. The SQL database resides on a RAID 1 and RAID 5 mirrored disk array. As will be explained hereinafter, database 1516 interacts with control server 1512 and information server 1514 through database server 1518. In the illustrative embodiment, database 1516 comprises a Client table 1516A, an On-line table 1516B, a WebBoard table 1516C, a WebBoard configuration table 1516D and a WebBoard Source table 1516E.

Client table 1516A comprises a plurality of records, each of which may have the fields and corresponding data elements as described in Table 1. Each WebPhone user, hereinafter "client," has a separate record in table 1516A containing the information defining the client's profile of personal information. In Table 1, the "activated," "paid," and "published" fields are boolean yes/no fields. The "id" field comprises a unique ID sequence identifying a particular WebPhone client. The "activation date," "address change date," and "access date" fields are time references measured in seconds since 00:00 Coordinated Universal Time (UTC), January 1, 1970. The "IPAddr" field represents the Internet protocol address of the WebPhone client and, if unknown, has a default value of 0.0.0.0. The database record containing a WebPhone client's profile, is defined upon first logging-on to global server 1500 and may be updated each time a WebPhone user's profile changes, as explained hereinafter.

The On-line table 1516B provides a dynamic list of those clients from 1516A who are currently On-line, as well as their current Internet protocol address. On-line Table 1516B comprises a plurality of records each of which may have the fields and data types illustrated in Table 2. The record entries of On-line table 1516B are used by connection server 1512 and information server 1514, as explained hereinafter, to provide a directory of those WebPhone client processes currently having on-line status with respect to the computer network.

The WebBoard™ is a virtual multimedia billboard which is transmitted as a series of multimedia data files to WebPhone client processes while the WebPhone application is activated. An extensive description of the WebBoard utility and its operation can be found in copending U.S. patent application serial number ^{08/719,891} ~~XXXXXX,XXX~~ entitled Method and Apparatus for Distribution of Multimedia Data Over a Computer Network by Mattaway et al., commonly assigned, the subject matter of which is incorporated herein by reference.

A number of tables are associated with the WebBoard functionality including WebBoard table 1516C, a WebBoard configuration table 1516D, and a WebBoard source table 1516E. WebBoard table 1516C includes a plurality of records each describing a specific WebBoard and having the field and data types illustrated in Table 3. The "id" field of Table 3 provides a unique identification number for the WebBoard file. The "imageType" field defines the video format of the image such as JPEG, TIF, GIF, etc. The "audio" field defines the nature of the audio file, e.g. a .wav file or a MIDI file, while the "audioType" field defines the codec, if any, used to compress/decompress the audio file. The "hits" field defines the number of times the WebBoard has been selected by WebPhone clients, while the "hits profile" field defines the file name of the file identifying those WebPhone clients generating hits to the subject WebBoard.

The WebBoard configuration table 1516D may have at least one record having the fields and data types illustrated in Table 4. The count field represents the number of WebBoard records currently in the table 1516C.

The WebBoard source table 1516E may comprise a plurality of records each having the fields and data types defined in Table 5. The "URL" field of Table 5 defines a data link in accordance with Uniform Resource Locator protocol to the home page or Web site of the source. In the illustrative embodiment, any entity, including vendors, advertisers, individuals or groups wishing to post information or having a Web site or home page may have a WebBoard displayable through the present invention.

Database Server

Database server 1518 serves as the interface between database 1516 and connection server 1512 and information server 1514. Specifically, connection server 1512 and information server 1514 communicate with database engine 1518 through application program interfaces embedded in the code implementation of both the connection server and the information server. Database server 1518 communicates with database 1516, in the illustrative embodiment, using the open database connectivity (ODBC) standard, developed by Microsoft Corporation, Redmond, WA. Database server 1518 functions to supply structured database queries to database 1516 and to supply the results therefrom to connection server 1514 and information server 1512. In the illustrative embodiment, database server 1518 may be implemented as a "virtual machine" executing on global server 1500, or, alternatively, may be implemented on a separate computer system such as a DEC Alpha 4100 Workstation executing DEC Unix operating system, both available from Digital Equipment Corporation, Maynard, MA. Database server 1518 communicates with

network interface card 1518 using the WebPhone Application Program Interface described herein.

Global Server Network

In the illustrative embodiment, global server 1500 is implemented as a single server apparatus on which a plurality of “virtual machines” execute simultaneously. However, it will be obvious to those reasonably skilled in the art that a plurality of separate servers, one dedicated to each of connection server 1512, information server 1514, and database server 1518 may be interconnected to database 1516 and to each other using a local area network, to form a composite “virtual” global server, as illustrated by Fig. 15B, the construction of the system illustrated in Fig. 15B being within the knowledge of those reasonably skilled in the art in light of the descriptions contained herein.

It is further contemplated within the present invention that more than one global server 1500 may be utilized, as illustrated by Fig. 15C. In this implementation, multiple global servers 1500A-D are maintained for fault tolerant load sharing, each one performing the above-described connection server, information server and database server processes. Each of global servers 1500A-D are connected to the Internet via a separate T1 or T3 connection to different Internet service providers, and are synchronized with each other via database server replication. In such an embodiment, multiple global servers may be located in close proximity or in geographically disparate locations. In such an embodiment, the WebPhone application is provided with the network address information of each global server 1500A-D. In the event that any one of the global servers initially contacted is nonresponsive the WebPhone application will attempt connection to one or more of the remaining global servers to obtain directory and information services.

Further, in an implementation with multiple global servers, if the initially contacted global server is unable to accommodate a WebPhone client request, or, is not geographically convenient, the global server can provide the network address of another global server capable of servicing the WebPhone client's request or which is logically more convenient. This process may occur during the initial log-in of the WebPhone client process, as described with references to messages 1-5 of Fig. 17A.

As previously described, if none of the global servers are available, the WebPhone application can rely on the secondary Internet Protocol technique in which a WebPhone client process sends its current dynamically assigned Internet Protocol address to a prospective WebPhone callee through an E-mail message, as described herein.

WebPhone Protocol

Prior to describing the interaction of the connection server 1512 and information server 1514 with WebPhone client processes, a description of the WebPhone protocol by which the WebPhone client processes and the global server 1500 communicate is appropriate. Tables 6-7 below illustrate the packet definitions of the packets comprising the WebPhone protocol (WPP) including the packet type, the direction and the data elements comprising each packet. In Tables 6-7 the symbol "→" indicates a packet transmitted by a WebPhone client process, while the "←" symbol indicates a packet transmitted by the global server. Tables 8-9 define the data elements described in Tables 6-7. In Tables 6-9, the terms "ULONG" and "UNSIGNED LONG" designate an unsigned long integer value, i.e., 32-bit integer value. Similarly, the terms "USHORT" and "UNSIGNED SHORT" designate an unsigned short integer value, i.e., 16-bit integer value. The term "CHAR" designates a single character, typically assuming a binary value of either 1 or 0. The term "VARCHAR(X)", where X is an integer, value

symbolizes a variable length character string, with the number of characters indicated with the integer value. The term "UNSIGNED CHAR" designates an 8-bit character code, i.e., no sign bit. Finally, the term "variable" indicates a variable length data field.

Figure 17A illustrates a schematic block diagram of a packet transfer sequence between a pair of WebPhone client processes and the global server, in accordance with the present invention. Each WebPhone application, also referred to as a WebPhone client process, connects to global server 1500 upon start up to inform global server 1500 that the WebPhone client process is on-line and available to make and/or receive calls. Specifically, as illustrated in Fig. 17A, WebPhone 1536 opens a socket to the global server 1500 and transmits an <ONLINE REQ> packet from WebPhone 1536 to Global server 1500, as illustrated by message 1 and Fig. 17A. The <ON LINE REQ> packet may have the format and data illustrated in Table 6, and additional Feature bits which define the functionality of the WebPhone application, as explained in greater detail hereinafter. In response, connection server 1512 and information server 1514 of global server 1500 use the information contained in the <ONLINE REQ> packet to update the status of database 1516. In the event that the WebPhone client process is logging on for the first time, global server 1500 returns to the WebPhone 1536 a <USER INFO REQ> packet, as illustrated by message 2 of Fig. 17A. The <USER INFO REQ> packet includes the elements as defined in Table 9. In response, WebPhone 1536 returns a <USER INFO> packet as illustrated by message 3 of Fig. 17A. The <USER INFO> packet contains the data elements defined in Table 8. Connection server 1512 and information server 1514 of global server 1500 utilize the data in the <USER INFO> packet to update database 1516. Specifically, information server 1514 utilizes such data to create a record in client table 1516A representing WebPhone 1536. Next, global server 1500 transmits to WebPhone 1536 a <REGISTRATION> packet, as illustrated by message 4 of Figs. 17A. The

<REGISTRATION> packet contains the data described in Table 7 plus Feature bits, as described hereinafter. The <REGISTRATION> packet returned to WebPhone 1536 enables certain functions within the WebPhone architecture based on predetermined criteria, for example, whether the user has paid for the product, or which version of the product the user possesses. Following the <REGISTRATION> packet, global server 1500 further transmits an <ONLINE ACK> packet, as illustrated by message 5 of Fig. 17A. Prior to transmission of the <ONLINE ACK> packet, connection server 1514 updates database 1516, specifically On-line table 1516B to indicate that WebPhone 1536 is on-line with respect to the computer network. Upon receiving the <ON-LINE ACK> packet, WebPhone 1536 closes the socket to global server 1500.

In the event WebPhone 1536 had previously registered with global server 1500, only messages 1 and 5 are required to establish WebPhone 1536 as being on-line. If WebPhone 1536 had new user information to supply to global server 1500, then packet sequence illustrated by messages 3 and 4 would occur.

Although the packet sequence illustrated by messages 1-5 is described with reference to WebPhone 1536, WebPhone 1538 interacts in a similar manner with global server 1500 to establish on-line status. No further interaction occurs between the respective WebPhone client processes and the global server unless the WebPhones require directory or search assistance about a prospective callee.

In one calling scenario, a WebPhone user knows the E-mail address of another WebPhone user to which he/she wishes to establish a point-to-point communication, however, the current dynamically assigned Internet protocol address of the callee is unknown to the caller. In this scenario, the user of WebPhone 1536 requests assistance from global server 1500 to obtain the current dynamically assigned Internet Protocol address of the prospective callee WebPhone. First, the user of WebPhone 1536 specifies the callee by entering all or part of the callee party's name or alias in the

party name field area of the graphic user interface. If the party is not in the WebPhone user's local directory, the IP address or E-mail address of the callee WebPhone may be entered into the number field area of the graphic user interface, followed by activation of the send button or icon on the graphic user interface. As a result, WebPhone 1536 opens a socket to global server 1500 and transmits a <CONNECT REQ> packet having the format described in Table 6. Connection server 1512 of global server 1500 utilizes the value of the E-mail address specified in the <CONNECT REQ> packet to perform a one-to-one mapping in the on-line table 1516B to determine the current Internet Protocol address of the indicated callee, as illustrated by the flowchart of Fig. 15A. Once this mapping is performed, the server 1500 transmits to WebPhone 1536 a <CONNECT ACK> packet, as indicated by message 7A of Fig. 17A. The <CONNECT ACK> packet has the format and content as illustrated in Table 6 and includes the IP address of the callee as well as information such as an error code to indicate that no WebPhone application is associated with that callee. Alternatively, if the selected callee is off line, global server 1500 transmits to WebPhone 1536 an <OFF LINE> packet to indicate that the desired party is not on-line, as illustrated by message 7B of Fig. 17A. Following the receipt of either a <CONNECT ACK> or an <OFF LINE> packet by WebPhone 1536, the socket to global server 1500 opened by WebPhone 1536 is closed.

If the current Internet Protocol address of the callee was returned from global server 1500, the packet transmission sequence illustrated between WebPhones 1536 and 1538 of Fig. 17A transpires. Whether a calling WebPhone knows the Internet Protocol address of the callee WebPhone, as in the case of a fixed Internet Protocol address, or obtains the Internet Protocol address from global server 1500, as previously described, the calling sequence to establish a call occurs as follows. WebPhone 1536 opens a socket to WebPhone 1538. Next, WebPhone 1536 transmits to WebPhone

1538 a <CALL> packet as illustrated by message 8 of Fig. 16A. The <CALL> packet has the format illustrated in Table 6 and may, optionally, include information identifying the compression/decompression (codec) used by the caller WebPhone. In response to the <CALL> packet, WebPhone 1538 may return with a number of different packets, as illustrated by messages 9A-D. First, callee WebPhone 1538 may respond to caller WebPhone 1538 with a <REJECT> packet, as illustrated by message 9A, indicating that the callee WebPhone does not wish to be disturbed, e.g. total call blocking, or, that the callee WebPhone does not wish to talk to caller WebPhone, e.g. party specific or group specific call blocking. In the event of party or group specific call blocking, the user information contained within the <CALL> packet of message 9A is compared by the caller WebPhone application to a predefined list of WebPhone user information profiles which the callee does not wish to converse, such list having been predefined by the callee in the WebPhone user's personal directory, as explained hereinafter. Upon receiving the <REJECT> packet the caller WebPhone announces the result to the user and the socket to the callee WebPhone is closed.

Alternatively, callee WebPhone 1538 may return a <BUSY> packet, as illustrated by message 9B of Fig. 17A. The <BUSY> packet indicates that the callee WebPhone is currently utilizing all available lines within its WebPhone application.

A further possible response from callee WebPhone 1538 is to issue an <ANSWER MACH> packet, as illustrated by message 9C of Fig. 17A. The <ANSWER MACH> packet includes data indicating whether the machine is capable of receiving voice mail type messages, as described in greater detail in copending U.S. patent application serial number ^{08/719,898} ~~XXXXX,XXX~~ entitled "Method and Apparatus for Providing Caller Identification Based Out-Going Messages in a Computer Telephony Environment," by Mattaway et al., commonly assigned and incorporated herein by reference.

The preferred response by callee WebPhone 1538 is to transmit a call acknowledge <CALL ACK> packet, as illustrated by message 9D of Fig. 17A. The <CALL ACK> packet has the data content illustrated in Table 6. Both the <CALL> and <CALL ACK> packets contain the information of the WebPhone users sending the packet. This information is useful by the recipient of the packet for a number of purposes. For example, the user information is displayed on the enunciator area of the WebPhone graphic display to identify the party placing the call. Second, the user may select such information and, using the drag and drop functionality of the WebPhone graphic user interface, add the user information to the callee WebPhone user's personal directory resident within his/her specific WebPhone application. In such a manner, both parties are completely identified to each other prior to commencing audio communications. The transmission of complete caller identification information with the <CALL> and <CALL ACK> symbols packets enables such functions as individual or group specific call blocking, party specific outgoing messages, visual caller identification, and party specific priority ringing and sound effects, as explained herein.

Following transmission of <CALL ACK> packet by callee WebPhone 1538, the callee WebPhone further transmits an <ANSWER> packet to caller WebPhone 1536, as illustrated by message 10 of Fig. 17A. Like the <BUSY> packet, the <ANSWER> packet is essentially empty, containing nothing more than a session ID number which is unique to the call. The socket previously opened by caller WebPhone 1536 over which the forgoing packets were transmitted remains open for the transmission of control information between caller WebPhone 1536 and callee WebPhone 1538. Such control information may comprise an <END> packet signaling the end of a call, a <HOLD> packet indicating that one of the parties to a call has placed the call "on hold" or other packets related to advance functionality of the WebPhone architecture. In addition, caller WebPhone 1536 opens a second socket to callee WebPhone 1538 over which

the respective WebPhones may exchange <AUDIO> packets, as illustrated by messages 11A-B of Fig. 17A. The <AUDIO> packets have the data content illustrated in Table 6. The WebPhone application enables the parties to converse in real-time, telephone quality, encrypted audio communication over the Internet and other TCP/IP based networks. If both WebPhone client processes are utilized with full duplex sound cards, such as that illustrated in Fig. 12, the WebPhone users may transmit and receive audio packets simultaneously, similar to normal telephone conversation. However, if the WebPhone client processes are used with half duplex sound cards, a WebPhone user may only transmit or receive audio data simultaneously, similar to a speaker phone. Exchange of <AUDIO> packets continues until either the callee WebPhone or the caller WebPhone transmits an <END> packet, as illustrated by message 12 of Fig. 16A. Following the receipt of an end packet, the WebPhone client process will cease to accept subsequent audio packets.

Following either transmission or receipt of an <END> packet by the caller WebPhone, the socket opened by the caller WebPhone to the callee WebPhone over which real-time audio communication occurred is closed. Similarly, the previously opened socket over which control information was transmitted between the callee and caller WebPhones is likewise closed.

Referring now Fig. 17B, if a WebPhone caller seeks to determine whether a prospective WebPhone callee is connected to the computer network, but, has little information regarding the client process, information server 1514 may be utilized as described. The WebPhone user defines One or more of the first name, last name, company, city, state, or country values of the Query field contained within the <INFO REQ> packet sends the packet to the global server. WebPhone 1536 opens a socket to global server 1500 and forwards <INFO REQ> packet to global server 1500, as illustrated by message 1 of Fig. 17B. Information server 1514 extracts the values

specified the query field of the <INFO REQ> packet and queries the database 1516, as previously described with reference to Fig. 16B. Global server 1500 then transmits a <INFO ACK> packet back to WebPhone 1536, as illustrated by message 2 of Fig. 17B. The <INFO ACK> packet has the format and data elements indicated in Table 7, including the number of parties satisfying the search criteria, specified in the <INFO REQ> packet. If the user of WebPhone 1536 wishes to receive the number of parties satisfying the search criteria global server 1500 automatically transmits to WebPhone 1536 one or more <INFO> packets, as illustrated by messages 3A-C of Fig. 17B. The <INFO> packet has the format and data elements as described in Tables 6-7. At any time following transmission of the <INFO ACK> packet, WebPhone 1536 may transmit an <INFO ABORT> packet to either prevent transmission of any <INFO> packets or to stop transmission of any remaining packets, as illustrated by message 4 of Fig. 17B. The <INFO ABORT> packet has the format and data elements as described in Table 6-7.

Once the user receives the information contained within the <INFO> packets satisfying the search criteria, the user may store such information in his/her personal WebPhone directory by dragging and dropping the information from the annunciator area to the direction dialog box using the WebPhone GUI.

The methods and apparatus described herein provide computer users with a powerful protocol in which to directly establish real-time, point-to-point communications over computer networks directly without server required linking. The a directory server assists in furnishing the current dynamically assigned internet protocol address of other similarly equipped computer users or information about such users.

WebPhone Graphic User Interface

Referring again to Fig. 14, the WebPhone GUI 1400 consists of a main window which has the look of a modern cellular flip phone and a set of dialog boxes launched from window. Operation of the WebPhone is controlled by selecting objects, i.e., buttons, text and images, and dragging objects, i.e., lines, parties, messages, etc., as explained hereinafter.

WebPhone GUI 1400 comprises a plurality of visual objects, including display 1402, number pad 1406, line pad 1404, call function buttons 1408, phone function buttons 1410 and audio controls (not shown). Display 1402 provides a number of distinct area for presentation of entering of information useful in operation of the WebPhone application. A party name field 1402A displays the name of the caller when an incoming call arrives and may also be used for entering the name of a party, up to 25 characters. By entering the name of a party in the party name field 1402A and pressing one or more of the phone function buttons 1410, various activities may be accommodated. For example, entering the name of a party in the party name field and pressing the [SND] button causes the WebPhone to first search the personal information directory for the information profile of the party entered. If such party's information is not already resident in the personal information directory, the WebPhone will open up a directory assistance dialog allowing the user to enter information to be submitted to the information server 1514 for searching, as described previously. Further, clicking the entered party name with the right mouse button causes a dialog box to appear enabling the user to modify the current directory entry, if any, for the party entered.

Entering the IP address of a party in the party IP address field followed by the [SND] button causes initiation of a call. If the callee's name exists within the caller's personal directory, or the call is established, the callee's name will appear in a party name field for caller ID purposes.

The third line of the display 1402 serves as a status annunciator line for displaying iconic feedback about the status of events within the WebPhone. Such status icons may include icons indicating enablement of call forwarding, call blocking, do not disturb, priority ringing, file transfer occurring, voice mail transfer occurring or call camping.

The line number annunciator indicates the line, i.e., lines 1-4, currently active, as illustrated by annunciated field 1402J. A main LED 1402F indicates when a line is active by changing color. Time field 1402C displays the local time when no lines are active. When one of the lines L1-L4 are active, time field 1402C displays the callee party's time. By single clicking the time field the user can cycle through the two different time values.

The line status field 1402H displays the status of the currently selected line, illustrated in Fig. 14 as displaying "talk" status. A call duration field 1402D displays the elapsed time in minutes and seconds since the currently displayed call commenced.

The V-mail field 1402G displays the number of the new voice mail messages and the total number of voice mail messages received.

When one or more call functions such as call conferencing, call blocking, priority ringing, call camping, or call forwarding are activated, the list of those parties within the WebPhone personal directory having such functionality active for their information profile can be viewed in the party name field by selecting a list arrow (not shown) icon which appears whenever one of the previously described functions is activated. Pressing the icon arrow allows the parties to be viewed sequentially.

The number pad buttons 0-9 also serve as speeddial buttons. Right clicking on any one of the number pad buttons 0-9 causes the name, alias, e-mail address and IP address, if known, of the party assigned to that speed dial position to be displayed on display 1402.

If a user right clicks on any of lines L1-L4 the name, alias, e-mail address and IP address of the party on that line will similarly appear for a predetermined period of time and then revert back to the normal display.

The keypad buttons displayed on WebPhone GUI 1400 may assume one of two states. A button may be a momentary button which, when pressed, i.e., left clicked, gets pushed in and then pops back out again. A second type of button is a toggle button which when pressed gets pushed in and stays in until pressed again. Number pad buttons 0-9 are momentary buttons which may be used to enter the Internet Protocol address of a party and which each house a speed-dial position. The user may assign a party to one of the ten speed-dial positions by selecting the user's information displayed in display 1402 and then dragging it onto the keypad button. To speed-dial one of the ten buttons the user simply presses the appropriate number followed by the [SND] button. As stated previously, if the user right clicks on one of the number pad buttons, the information about the party assigned to the speed-dial position will be displayed.

The line pad 1404 comprises four toggle buttons L1-L4, each of which has a letter, a number and an LED indicating the status of the line. When one or more parties are associated, i.e., dragged and dropped, with a line, the letter designating the appropriate line turns from an L to C indicating a conference call. When only one party is left on the line the letter designation reverts from a C back to an L indicating a regular call. Only one line, button may be selected at a time when an incoming call arrives. Pressing any of the line buttons assigns the incoming call to the selected line. Pressing a line button, i.e., left clicking, when the line is in use places the line on hold. Subsequent depressing the line button takes the call off hold.

A number of call function buttons 1408, including the [RCL], [END], [SND], [DND], [MUT], [HLD], [CMP], [BLK], [PRI], [FWD], not all of which are shown in Fig. 14,

are used to control operation of calls. The [RCL] button is a momentary button used to recall the last number dialed. Pressing [RCL] recalls the last party called by displaying the party's name, alias, e-mail address and IP address, if known. Selecting a free line following depression of the [RCL] button followed by the [SND] button will cause the party last called to be dialed. The [END] button is a momentary button and terminates a call upon depression. The [SND] button is a momentary button and is used to both place and answer calls. Depressing the [SND] button when a call is being announced causes the call to be answered on a preselected line or a line indicated by the user. Depression of the [SND] button once a callee's information is entered into display 1402 causes the party to be called, if the required information is present, or otherwise causes an information server connection to be established and activated, as previously described.

The [DND] button is a toggle button and is used to activate the Do Not Disturb function of the WebPhone. When activated, the [DND] button causes all inbound calls to be routed to the answering machine.

The [MUT] button is a toggle button which, upon depression, causes disabling of the microphone associated with a user's WebPhone system. When the [MUT] button is enabled, the main LED 1402F and the status line 1402H change to indicate that the call muted. Depression of the [MUT] button is undetected by one or more callees.

The [HLD] button is a momentary button and is used to place a call on hold. When a user depresses the [HLD] button a party or parties to a conference call are placed on hold, e.g., the microphone and speaker of the system are effectively disabled. When a called is placed on hold, the main LED 1402F and call status field 1402H indicate the change. To take a call off hold, the user depresses the line button of the call being held.

The [CMP] button is a momentary button that causes the WebPhone user to camp on a party, i.e., perpetual redial. Camping on a party serves to insure that the user's call will go through when the party is available. After placing a call, if the callee responds with either a busy or on off-line status, the user may press the [CPM] button to camp on that party. To remove a camp from a party, the user presses the delete key from the computer keyboard.

The [BLK] button is a toggle button and enables or disables call blocking. Depression of the [BLK] button enables call blocking causing all inbound calls from parties who have call blocking designated in their information profile within the personal information directory to be either rejected or routed to the answer machine. Whether a call is to be rejected or routed to the answering machine is specified in a party's information profile record within the personal information directory, in a manner, as previously described.

The [PRI] button is a toggle button which enables or disables priority ringing. Depression of the button enables priority ringing of all inbound calls from parties, i.e. generation of customized sound effects and/or graphic announcements when a call arrives. As with call blocking, priority ringing is specified within a party's information profile record in the user's personal information directory.

The [FWD] button is a toggle button which enables or disables call forwarding. Depression of the button enables call forwarding of selected inbound calls to the party specified in the appropriate information profile record in the personal information directory. The WebPhone will first search in the personal information directory for an information profile record which matches the inbound call. If a match occurs, and call forwarding is enabled, the inbound call will be forwarded to the party designated within the matched information profile record. If no party is designated, the call will be forwarded to a default forwarding party.

In addition to the call function buttons, a number of phone function buttons 1410 including a [CFG], [DIR], [MSG], [DAT], [LOG], [], and ? buttons enable users to further direct functions of a phone. Specifically, the ? button is a momentary button which invokes an interactive, multimedia tutorial and help system about the WebPhone. The [CFG] button is a momentary button, depression of which launches a configuration dialog which enables the user to change the operating parameters of the WebPhone. The [DIR] button is a momentary button, depression of which launches the phone directory dialog which enables a user to add, store, update, view, and delete parties and to obtain directory assistance from global server 1500, as described previously. The [MSG] button is likewise a momentary button, depression of which launches the voicemail message dialog which enables a user to view, sort, playback, delete, save and restore voicemail messages, as well as to create, playback, delete, save, and restore custom outgoing messages and assign them to information profile records in the personal information directory.

The [DAT] button is a momentary button, depression of which launches a data file transfer dialog enabling a user to monitor and control the progress of a data file transferred over the communication link established with the WebPhone, such dialog further enables a user to retrieve and create E-mail.

The [LOG] button is a momentary button, depression of which launches a call activity log dialog which enables a user to use, sort, search for, print, and delete call related events. An "X" icon is provided to exit the WebPhone. If one or more calls are active when the X icon is selected, a dialog box will appear asking the user if he/she really wishes to exit and terminate active calls. Other icons are provided for minimizing or iconifying the WebPhone application.

In addition to the above-described display, the WebPhone GUI 1400 includes a number of audio control buttons and sliders (not shown in Fig. 14). These graphic

elements enable the user to control the recording the playback of voicemail and outgoing messages and operate similar to conventional audio tape deck controls. In the illustrative embodiment, and similar to that shown in Fig. 5, a progress bar is illustrated which displays the extent of progress during playback and audio recording processes. Momentary buttons may be provided for rewinding the “virtual tape” to the beginning and for fast forwarding the tape to the end of a recording. Further, momentary buttons are provided for aborting, as well as stopping, playback of audio. A speaker card button, implemented as a toggle button, is provided to play back audio on the sound card’s speaker. A special momentary button for audio playback is provided. When initially depressed, audio playing commences. The button then pops out and becomes a pause button. Subsequent depression pauses the audio. The button then pops out again to become a play button. A record button, in the form of a toggle button is provided to control recording of audio. When the button is depressed the user is in an audio record mode and can record voicemail or outgoing messages. To stop recording, the button is pressed again or the stop is button is pressed. A slider-type graphic potentiometer is provided to control speaker volume and enables the user to adjust output volume of the audio received during conversation and playback of voicemail and outgoing messages. The speaker control will attenuate the sound card speaker volume. A similar control is provided to control microphone volume and enables the user to adjust the input volume of audio recorded during conversation and recording of voicemail and outgoing messages. The microphone slider control attenuates the sound card’s microphone volume.

WebPhone Application Object Implementation

As previously described, with reference to Figs. 13A-B, the WebPhone application comprises a set of object modules which work together in a concerted

fashion to provide real-time, multitasking, network-based media transmission and reception. Specifically, the WebPhone GUI, user interface, event manager, and media engine utilize a number of objects to house and manipulate data associated with the operation of the WebPhone application. The GUI objects control the look and feel of the graphic user interface controls which comprise the WebPhone user interface. Some user interface objects maintain and manage many of the states of the WebPhone and control the behavior of the GUI controls, as illustrated in Figs. 18A-D.

Fig. 18A illustrates the hierarchical relationship between objects within the WebPhone. The UIVirtualBase 1812 is a class from which UIVirtualControl object 1810 and UIVirtual object 1808 inherit their respective attributes and member functions. GUIControl object 1802 inherits its attributes and member functions from UIVirtualControl 1810, as illustrated. UICollection object 1806 inherits its properties from the UIVirtual object class 1808. The UIControl object inherits its attributes and member functions from both the UIVirtual control object class 1810 and the UIVirtual object class 1808.

Referring to Fig. 18B the UIControl object 1804 itself serves as a class from which the UIButton object 1828, UISlider object 1826, UIScroller object 1824, UITab object 1822, UIDisplay object 1818, UIListBox object 1820, UIComboBox 1814, and UIEditBox 1816 are subclasses. As illustrated in Fig. 18C, the UIPushButton 1842, UIPlayRun object 1844 and UIToggle object 1846, are subclasses of the UIButton object 1848. As illustrated in Fig. 18D, the UIPhone object 1838, UICall object 1832, UILine object 1834, and UIPopUp object 1836 are derived from or inherit their attributes and member functions from the UICollection object class 1806.

Each WebPhone control has two objects associated therewith, a windowing system specific GUIcontrol object 802 and a generic UI control object 1804. When the GUIcontrol object's state is changed by the user, GUIcontrol 1802 verifies the change

with UIcontrol 1804 to validate the change. UIcontrol 1804 is a child of the UIcollection 1806. When UIcontrol's sibling, GUIcontrol 1802 requests UIcontrol 1804 to verify a change, and the change is accepted, GUIcontrol 1802 must verify the change with its parent object. The parent UIcollection 1806 may have its own parent, another UIcollection object, that it must verify the change with. The UIPhone object 1838 is a member of the UI collection class. UIPhone has final approval over all changes in the state of the WebPhone. UIPhone 1838 further tells child objects when the event manager changes the phone state and further creates jobs for the event manager based on user actions.

The WebPhone drag and drop functionality utilizes the standard Windows® drag and drop interface and adds several unique object types to interact therewith. Specifically, each UIcontrol and GUIcontrol object has two new member functions added, e.g., set dragtype and acceptdrop types. The set dragtype call sets the type of drag that the control will perform if the mouse or other pointing device is moved out of the control window with the left mouse button down. The accept droptype defines the types of drags the control will accept.

Event Manager and Media Engine

The event manager is a state machine consisting of an array of pointers to functions and states which make up a state-event table. When an event occurs as caused by the mouse, keyboard, mic, speaker, or socket, it is up to the user interface to determine if the event requires the attention of the event manager. The event manager is not notified of events which effect only the graphic user interface, e.g., the user depresses the [DIR] button to open the phone directory dialog.

Referring to Figs. 19A-C, a conceptual block diagram illustrating the event manager and media engine objects utilized by the WebPhone is presented.

Specifically, the following objects are utilized by both the user interface and the event manager to manager the state of calls and tasks that are to be performed:

- line
- job
- party
- task

As illustrated in Fig. 19A, a Line object is represented by the pentagon shape with a number contained therein. The Line object has the attributes of state and duration and a *job pointer. Member functions for the Line object include createcall () and removecall (). The Job object is illustrated with a rectangle having pointers extended therefrom as illustrated in Fig. 19A. Attributes of the job object include, ID, type, state, and parties, and pointer attributes party, inTask, outTask, nextJob, prevJob. The Job object has the member functions of AddParty, RemoveParty, CreateTask, and RemoveTask. The Party object, illustrated with a triangular symbol, includes the attributes of state, session, socket, and partyRec, and the member functions of LoadParty.

The Task object includes the attributes of command, source, destination, extent, fileHandle, fileType, fileLength, fileSize, mic, speaker, and flags, as wells as pointer attributes *job and *buf. The values assumable by the command attribute of the Task object may include initialize, close, start, stop, fill, and use, etc. The values assumable by the source and destination attributes of the task object may include microphone, speaker, socket, and file. Fig. 19B illustrates the relationship between Line objects and Job objects and the pointers linking the two. Fig. 19 illustrates the relationship between

Party objects, Job objects and Task objects and the pointers linking the Job objects to the parties and tasks.

Media Engine Implementation

Figs. 20A-D illustrate the process steps performed by the media engine of the WebPhone in accordance with the present invention. The coding of the process steps of the flowchart of Figs. 20A-D and to instructions suitable for use by the WebPhone will be understandable by those having ordinary skill in the programming arts. Fig. 20A illustrates the process executed by the media engine when the CMD attribute of a Task object is defined as a AE_USEME command, as previously illustrated in Fig. 19A. The Task objects are set up by the event manager. The media engine manages routing and resources. For example a microphone, file or socket may provide a source of data to media engine while a destination may comprise either a speaker file or socket. The media engine serves to perform compression/decompression as well as copying functions. For the purposes of describing flowcharts 20 A-D the media engine will be referred to as media engine 2000.

Referring to Fig. 20A, media engine 2000 first determines the source of a data stream, as illustrated by decisional block 2002. If the source is a microphone, media engine 2000 determines whether or not the current audio data from the microphone source is silence, as illustrated in decisional block 2004. If the audio stream from the microphone is not silent the data will be accumulated into a microphone buffer, as illustrated by procedural block 2006. Next, the media engine will determine whether or not the buffer is full, as illustrated by decisional 2008. If the buffer is full, process flow will proceed to a determination of the destination via connector Q. If in decisional block 2004 the determination was made that the audio data from the microphone was silence, the media engine notes the length of the silence, as illustrated by procedural block

2010. Next, the media engine determines whether or not the buffer is empty, as illustrated by decisional block 2012. If the buffer is empty, process flow proceeds to a determination of the source, via connector R, as illustrated by decisional block 2030.

Returning again to decisional 2014, a determination of the destination of the audio data made after either a determination that the buffer is full, via connector Q, or that the source of the audio data is a socket, e.g., one of the branches of decisional block 2002. If in decisional block 2014 a determination is made that the destination is a socket, media engine 2000 determines if a party is online, as illustrated by decisional block 2028. If the party is online media engine 2000 will write to the socket associated with that party, as illustrated by procedural block 2026. The process as illustrated by decisional 2028 and process block 2026 are repeated for every party associated with the Job object, i.e., conference calls include multiple parties. Following writing to the parties socket, process flow returns decisional block 2030 for a determination of the source, as illustrated. If in decisional block 2014 a determination was made that the speaker was the destination, media engine makes a further determination to whether or not there is more than one party on the conversation, i.e., conference call, as illustrated by decisional block 2020. If there is only one other party besides the user on the call, process flow proceeds to junction K where the audio data is written to the speaker, as illustrated by process block 2022. If in decisional block 2020 a determination was made that multiple parties were associated with a call media engine 2000 mixes the audio data into a mixing buffer, as illustrated by process block 2016. Next media engine 2000 determines whether or not the speaker is idle. If so, the audio data from the mixing buffer is written to the speaker as illustrated by procedural block 2022. Otherwise, process flow proceeds to junction R. In decisional block 2030 media engine 2000 determines again what the source of an audio data stream is. If the source is determined to be a socket, media engine 2000 will place the empty buffer on

the winSock queue, as illustrated by process block 2036. If the source is determined to be a microphone, and the microphone is enabled, as determined in decisional block 2032, media engine 2000 will place the empty buffer on the mic sampling queue, as illustrated by process block 2034. Otherwise, media engine 2000 will place the empty buffer in the free pool of buffer space, as illustrated by process 2038. Either branch of decisional block 2030 will result in a return from the task execution process, as illustrated.

Fig. 20B, illustrates the process flow performed by media engine 2000 upon receiving a task object from the event manager having the CMD attribute defined with a AE_START, i.e., the event manager instructs the media engine to start a copy operation from a source to a destination. First, media engine 2000 determines whether or not the source is a microphone or a file, as illustrated by decisional block 2040. If the source is a file, process flow proceeds to block 2062 of Fig. 20C via connector F, as described hereinafter. If the source is determined to be a microphone, media engine 2000 will determine whether or not the microphone is on, as illustrated by decisional 2044. If the microphone is not on, an internal error notification will be generated, as illustrated by procedural block 2046. If the microphone is on, media engine 2000 will enable microphone sampling, obtain space from the buffer pool, and perform an asynchronous read from the microphone, as illustrated by process blocks 2048, 2050 and 2052, respectively. If in decisional block 2040 media engine 2000 determined that the source was a socket, buffer space will be retrieved from the buffer pool, as illustrated by process block 2042, and an asynchronous read from the socket will be performed, as illustrated by process block 2045. Following the an asynchronous read from either a socket or a microphone, media engine 2000 will return the task to the event manager, as illustrated.

Fig. 20 illustrates the process flow performed by media engine 2000 upon receiving a Task object from the event manager in which the CMD attribute is defined with a AE_FILLME command value, i.e., an empty packet has been returned from either an MCI or WINSOCK asynchronous write operation upon completion. First, media engine 2000 determines whether the source is from a file or either a socket or speaker, as illustrated by decisional block 2054. If the source is a file, media engine 2000 will read a portion of the file, as illustrated by process block 2062. Next, media engine 2000 will make a determination as to whether the destination is either a socket or a speaker, as illustrated by decisional block 2068. If the destination is a socket process flow will return to decisional block 2028 of Fig. 20A via connector S, as illustrated. If the destination is a speaker, process flow will proceed to process block 2022 of Fig. 20A via connector K as illustrated.

If a determination was made in decision 2056 that the destination is a socket, media engine 2000 will place the buffer associated with the task or message in the WINSOCK free pool of buffer space, as illustrated by process block 2058. If the destination is determined to be a speaker, media engine 2000 next determines whether or not the buffer is empty, as illustrated by decision block 2060. If the buffer is not empty, the data within the mixing buffer will be written to the speaker, as illustrated by message 2064. If the buffer is empty, the buffer associated with the message, i.e., task, will be placed in the MCI message free pool, as illustrated by process block 2066. Both branches decisional block 2056 result in a return from the task by media engine 2000, as illustrated. In the above-described flow diagrams, a message may be a task implementation similar to the manner in which Microsoft Windows uses messages for task completion operations.

Fig. 20D illustrates the process path taken by media engine 2000 when the CMD attribute of a Task object is defined as a AE_STOP value, i.e., the event manager

instructs the media engine to stop the current operation on behalf of a specified task. The process begins with the determination of whether or not the source is a microphone or file, as illustrated by decisional block 2070. If it is determined that the source is a file, process flow proceeds to block 280 where the source is set to none, i.e., no further data will be retrieved or processed. If the process is determined to be a socket, media engine 2000 cancels any pending asynchronous reads from the socket, as illustrated by process block 2074. If a determination is made that the source is a microphone, media engine 2000 will determine whether or not the microphone is on, as illustrated by decisional block 2072. If the microphone is on, media engine 2000 cancels sampling of the audio signal from the microphone, as illustrated by process block 2076, and, discards the pending data in the mix buffer, as illustrated by process block 2078. Regardless of the determination of the source, all branches of the process flow terminate with the setting of the source to none or null, indicating a termination of the operation and a return by media 2000 from the task, as illustrated.

In an alternate embodiment, the various aspects of the invention may be implemented as a computer program product for use with a computer system. Such implementation may comprise a series of computer instructions either fixed on a tangible medium, such as a computer readable media, e.g. diskette 1142, CD-ROM 1147, ROM 1115, or fixed disk 1152 of Fig. 11, or transmittable to a computer system, via a modem or other interface device, such as communications adapter 1190 connected to the network 1195 over a medium 1191. Medium 1191 can be either a tangible medium, including but not limited to optical or analog communications lines, or may be implemented with wireless techniques, including but not limited to microwave, infrared or other transmission techniques. The series of computer instructions embodies all or part of the functionality previously described herein with respect to the invention. Those skilled in the art will appreciate that such computer instructions can be

written in a number of programming languages for use with many computer architectures or operating systems. Further, such instructions may be stored using any memory technology, present or future, including, but not limited to, semiconductor, magnetic, optical or other memory devices, or transmitted using any communications technology, present or future, including but not limited to optical, infrared, microwave, or other transmission technologies. It is contemplated that such a computer program product may be distributed as a removable media with accompanying printed or electronic documentation, e.g., shrink wrapped software, preloaded with a computer system, e.g., on system ROM or fixed disk, or distributed from a server or electronic bulletin board over a network, e.g., the Internet or World Wide Web.

Although various exemplary embodiments of the invention have been disclosed, it will be apparent to those skill in the art that various changes and modifications can be made which will achieve some of the advantages of the invention without departing from the spirit and scope of the invention. These and other obvious modifications are intended to be covered by the appended claims.

TABLE 1
Client Table

<u>Field</u>	<u>Data Type</u>	<u>Comments</u>
id	ulong	Unique ID Sequence
activated	char	0 = NO, 1 = YES
activationDate	ulong	Secs since 00:00 UTC Jan 1, 1970
version capability	ushort	Version of the Webphone
version protocol	ushort	
version vendor	ushort	
paid	char	0 = NO, 1 = YES
prePaidCode	varchar(16)	
firstName	varchar(10)	
lastName	varchar(25)	
alias	varchar(20)	
emailAddr	varchar(90)	
IPAddr	varchar(80)	0.0.0.0 if not known
street	varchar(50)	
apt	varchar(5)	
city	varchar(20)	
state	varchar(20)	
country	varchar(20)	
postalCode	varchar(20)	
phone	varchar(25)	
fax	varchar(25)	
feature bits	ulong	WebPhone Feature Definitions
company	varchar(25)	Company Name
addrChanges	char	No. of address changes
addrChangeDate	ulong	Secs since 00:00 UTC
publish	char	0 = NO, 1 = YES
accessDate	ulong	Secs since 00:00 UTC

Twitter

TABLE 1 (Con't)

<u>Field</u>	<u>Data Type</u>	<u>Comments</u>
accessCount	ulong	# of log ons
callCount	ulong	# of outbound calls
social security number	ulong	optional
age	ushort	optional
occupation code	ushort	optional
interest codes	ushort	optional
household income range	ushort	optional

**TABLE 2
Online Table**

<u>Field</u>	<u>Data Type</u>	<u>Comments</u>
emailAddr	varchar(90)	
IPAddr	varchar(80)	
flags	char	
onlineDate	ulong	Secs since 00:00 UTC

**TABLE 3
WebBoard Table**

<u>Field</u>	<u>Data Type</u>	<u>Comments</u>
id	ulong	Unique ID Sequence
image	varchar(255)	Filename of image file
imageType	char	.GIF =0, JPG = 1, RLE = 3
audio	varchar(255)	Filename of TSP encoded.WAV file
audioType	char	GSM=0, TRUESPEECH = 1
hits	ulong	Number of accrued hits
hitsprofile	varchar(8)	Filename of Demographics
version	ulong	version of WebBoard
URL	varchar (255)	home page url

**TABLE 4
Webboard Config Table**

<u>Field</u>	<u>Data Type</u>	<u>Comments</u>
count	ulong	Number of WebBoards

TABLE 5
Source Table

<u>Field</u>	<u>Data Type</u>	<u>Comments</u>
id	ulong	Unique ID Sequence
weboardID	ulong	Link to WebBoard record
name	varchar(50)	Company's name
url	varchar(80)	URL to Home Page
street	varchar(50)	
apt	varchar(5)	
city	varchar(20)	
state	varchar(20)	
country	varchar(20)	
postalCode	varchar(20)	
phone	varchar(25)	
fax	varchar(25)	
contact	varchar(35)	Name of contact

1690X

TABLE 6
WebPhone Protocol (WPP) Packet Definitions

<u>Packet</u>	<u>Packet Type</u>	<u>Direction</u>	<u>Data</u>
Invalid	WPP_INVALID	--	WPP_INVALID
Online Req	WPP_ONLINEREQ	-	WPP_ONLINEREQ, sid, version, emailAddr, IPAddr, onlineState, feature bits
OnlineACK	WPP_ONLINEACK	-	WPP_ONLINEACK, sid, onlineStatus, feature bits
Offline	WPP_OFFLINE	--	WPP_OFFLINE, sid
Hello	WPP_HELLO	--	WPP_HELLO, sid, version
Connect Req	WPP_CONNECTREQ	-	WPP_CONNECTREQ, sid, version, callType, partyEmailAddr, emailAddr, IPAddr, connectState
Connect ACK	WPP_CONNECTACK	--	WPP_CONNECTACK, sid, connectStatus, partyIPAddr
Call	WPP_CALL	--	WPP_CALL, sid, version, emailAddr, IPAddr, userInfo
CallACK	WPP_CALLACK	--	WPP_CALLACK, sid, version, emailAddr, IPAddr, userInfo
CnfCall	WPP_CNFCALL	--	WPP_CNFCALL, sid, version, emailAddr, IPAddr, userInfo
CnfCallACK	WPP_CNFCALLACK	--	WPP_CNFCALLACK, sid, version
Answer	WPP_ANSWER	--	WPP_ANSWER, sid
Busy	WPP_BUSY	--	WPP_BUSY, sid
AnsMachine	WPP_ANSMACH	--	WPP_ANSMACH, sid, state
End	WPP_END	--	WPP_END, sid
Hold	WPP_HOLD	--	WPP_HOLD, SID, (ON/OFF)
Reject	WPP_REJECT	--	WPP_REJECT, sid
Camp	WPP_CAMP	--	WPP_CAMP, sid
CampACK	WPP_CAMPACK	--	WPP_CAMPACK, sid
Audio	WPP_AUDIO	--	WPP_AUDIO, sid, audioType, silence, length, audioData
Pulse	WPP_PULSE	-	WPP_PULSE, sid
Adjpulse	WPP_PULSE	-	WPP_ADJPULSE, sid, adjPulse

T700X

TABLE 6 (con't)

<u>Packet</u>	<u>Packet Type</u>	<u>Direction</u>	<u>Data</u>
Vmail	WPP_VMAIL	--	WPP_VMAIL, sid, audioType, silence, length, audioData
VmailEnd	WPP_VMAILEND	--	WPP_VMAILEND, sid
OgmEnd	WPP_OGMEND	--	WPP_OGMEND, sid
CnfAdd	WPP_CNFADD	--	WPP_CNFADD, sid, partyEmailAddr, partyIPAddr, partInfo
CnfDrop	WPP_CNFDROP	--	WPP_CNFDROP, sid
FileXmtReq	WPP_FILEXMTREQ	--	WPP_FILEXMTREQ, sid, fileType, fileName, fileSize

TABLE 7
WebPhone Protocol (WPP) Packet Definitions

<u>Packet</u>	<u>Packet Type</u>	<u>Direction</u>	<u>Data</u>
FileXmtAck	WPP_FILEXMTACK	↔	WPP_FILEXMTACK, sid
File	WPP_FILE	↔	WPP_FILE, sid, length, fileData
FileXmtEnd	WPP_FILEXMTEND	↔	WPP_FILEXMTEND, sid
FileXmtAbort	WPP_FILEXMTABORT	↔	WPP_FILEXMTABORT, sid
InfoReq	WPP_INFOREQ	-	WPP_INFOREQ, sid, query
InfoACK	WPP_INFOACK	-	WPP_INFOACK, sid, nparties
Info	WPP_INFO	-	WPP_INFO, sid, partyInfo
InfoAbort	WPP_INFOABORT	-	WPP_INFOABORT, sid
UserInfoReq	WPP_USRINFOREQ	-	WPP_USRINFOREQ, sid
UserInfo	WPP_USRINFO	-	WPP_USRINFO, sid, version, userInfo
WBImageStart	WPP_WBIMAGESTART	-	WPP_WBIMAGESTART, sid, fileSize, imageType, url
WBImage	WPP_WBIMAGE	-	WPP_WBIMAGE, sid, length, imageData
WBImageEnd	WPP_WBIMAGEEND	-	WPP_WBIMAGEEND, sid
WBAudioStart	WPP_WBAUDIOSTART	-	WPP_WBAUDIOSTART, sid, fileSize, audioType
WBAudio	WPP_WBAUDIO	-	WPP_WBAUDIO, sid, length, audioData
WBAudioEnd	WPP_WBAUDIOEND	-	WPP_WBAUDIOEND, sid
Registration	WPP_REG	-	WPP_REG, sid, feature bits, EMAILAddr, customer id
Audio Start	WPP_AUDIO START	↔	WPP_AUDIO START, sid
Audio End	WPP_AUDIO END	↔	WPP_AUDIO END, sid
Caller OK	WPP_CALLEROK	-	WPP_CALLEROK, sid, version, emailAddr, feature bits
Caller ACK	WPP_CALLERACK	-	WPP_CALLERACK, sid, callerStatus, feature bits
Key Pad	WPP_KEYPAD	-	WPP_KEYPAD, sid (ON/OFF)
Key	WPP_KEY	-	WPP_KEY, sid, ascii character
WBLIST	WPP_WBLIST	-	WPP_WBLIST, sid, list of WB IDs
WBLIST REQ	WPP_WBLISTREQ	-	WPP_WBLISTREQ, sid

TRBOX

TABLE 7 (con't)

<u>Packet</u>	<u>Packet Type</u>	<u>Direction</u>	<u>Data</u>
WB REQ	WPP_WEBBOARDREQ	-	WPP_WEBBOARDREQ, sid, WBIid, Client id
WB HIT	WPP_WEBBOARDHIT	-	WPP_WWBOARDHIT, sid, WB id, Client id
ANS FULL	WPP_ANS FULL	-	WPP_ANS FULL, sid

TABLE 8

WebPhone Protocol (WPP) Packet Data Definitions

<u>Element</u>	<u>Data Type</u>	<u>Comment</u>
WPP_*	unsigned char	WPP message identifier
sid	unsigned long	session id unique per call
version	unsigned(3)	version of the webphone (capability, protocol, vendor)
emailAddr	varchar(90)	email address of caller
IPAddr	varchar(80)	IP Address
onlineState	unsigned char	bit 0 (ACTIVE/INACTIVE) bit 1 (Merchant Phone) bit 2 (Connection Server) bit 3 (webboard disabled) bit 4 Not Used bit 5 Not Used bit 6 Not Used bit 7 Not Used
call Type	unsigned char	call type 0: EMAIL / 1:IPCALL
partyEmailAddr	varchar(90)	email address of person to call
connectStatus	unsigned char	0:NO WEBPHONE 1: ONLINE

7730X

TABLE 8 (con't)

<u>Element</u>	<u>Data Type</u>	<u>Comment</u>
		2:OFFLINE
		3:RECONNECT
		4:PERM_RECONNECT
partyIPAddr	varchar(80)	IP Address of person to call
userInfo	varchar(120)	firstName, LastName, alias, emailAddr, street, apt, city, state, country, postalCode, phone, fax, company
audioType	unsigned char	audio compress type
		0:GSM
		1:TRUESPEECH

TABLE 9
WebPhone Protocol (WPP) Packet Data Definitions

<u>Element</u>	<u>Data Type</u>	<u>Comment</u>
length	unsigned short	length of audio or data in bytes
audioData	512 Bytes	compressed audio data
feature bits	unsigned long	WebPhone feature definition
fileType	unsigned char	file type 0:DATA 1:EMAIL 2:TEXT 3:BINARY
fileName	varchar(13)	name of file to be transmitted.
fileSize	unsigned long	size of file to be transmitted in bytes
fileData	variable	file data
query	varchar(120)	firstName, lastName, company, city, state, country
nparties	unsigned long	number of parties or query records being sent
size	unsigned long	size of file (IMAGE or AUDIO) to be sent
imageType	unsigned char	image type 0:GIF 1:JPG
imageData	512 Bytes	image data
emailAddr	varchar(90)	encrypted email Address
onlineStatus	unsigned char	0 OK -1 Error
callerStatus	unsigned char	0 is unpaid 1 if paid
onlineState	unsigned char	bit 0 webboard disabled bit 1 Not Used bit 2 Not Used bit 3 Not Used

7750X

TABLE 9 (con't)

<u>Element</u>	<u>Data Type</u>	<u>Comment</u>
		bit 4 Not Used
		bit 5 Not Used
		bit 6 Not Used
		bit 7 Not Used
WBid	unsigned long	link to WebBoard record
adjpulse	unsigned long	timer offset in secs

**Table 10
Feature Definition**

feature bit 0	0 = 1 line	1 = 4 lines
bit 1	0 = Limited Call Time	1 = Unrestricted Call Time
bit 2	0 = Limited VMail OGM	1 = Unlimited Vmail OGM
bit 3	0 = Limited Directory Entries	1 = Unlimited Dir Entries
bit 4	0 = Webboard Not Disabled	1 = Allowed to Disable
bit 5	0 = Conferencing(audio) Disabled	1 = Conferencing Enabled
bit 6	0 = Conferencing(video) Disabled	1 = Conferencing Enabled
bit 7	0 = Whiteboard Disabled	1 = Whiteboard Enabled
bit 8	0 = Offline voicemail Disabled	1 = Offline voicemail Enabled
bit 9 - 27	Reserved	
bit 28 - 30	Type of Phone	
	0 - Normal webphone 1 - Agent 2 - Business webphone 3 - Gateway 4 - ACD 5-7 reserved	
bit 31	1 = Disable all WebPhone features	

7760X

TABLE 11

<u>Offset</u>	<u>Name</u>	<u>Size</u>	<u>Description</u>
	Reserved		Reserved
+1	SessionID	4	Unique value for duration of this connection
+5	Version	6	WebPhone version and distributor stamp
+11	Codec	1	Audio compression algorithm selected
+12	FirstName	10	Given name, middle initial
+22	LastName	25	Surname
+47	Alias	20	Nickname
+67	EmailAddr	90	Caller's electronic mail address
+157	IpAddr	80	Caller's WebPhone's Internet address
+237	Street	50	Street address of user
+287	Apt	20	Apartment or suite number
+307	City	20	City name
+327	State	20	State or province
+347	Country	20	Country name
+367	ZipCode	20	Zip or postal code
+387	Phone	25	Telephone number
+412	Fax	25	Facsimile telephone number
+437	Company	25	Employer or organization name
+487	File Name	25	Name of file
+512	Action Code	25	Action descriptor
+537	File Type	10	File type descriptor
+547	Status	25	Status of WebPhone utility

T770X

CLAIMS

1.
Sub
a

A computer program product for use with a computer system having a display and an audio transducer, the computer system operatively coupled to other computers and a server over a computer network, the computer program product comprising a computer usable medium having computer readable code means embodied in the medium comprising:

- a. program code means for generating a user-interface through which a user may coact with the computer system;
- b. program code means responsive to user input commands for establishing a point-to-point communications link with another computer over the computer network; and
- c. program code means, responsive to audio data from the audio transducer, for transmitting the audio data over the communication link to the other computer.

2. The computer program product of claim 1 wherein the program code means for establishing a point-to-point communication link further comprises:

- c.1 program code means, responsive to the network protocol address of the second processor, for establishing a point-to-point communication link between the first processor and the second processor over the computer network.

3. The computer program product of claim 2 wherein the program code means for establishing a point-to-point communication link further comprise:

- c.2 program code means for transmitting, from the first processor to the server, a query as to whether the second processor is connected to the computer network; and

- c.3 program code means for receiving a network protocol address of the second processor from the server, when the second processor is connected to the computer network.
4. The computer program product of claim 2 wherein the program code means for establishing a point-to-point communication link further comprises:
- c.2 program code means for transmitting an E-mail signal containing a network protocol address from the first processor to the server over the computer network;
 - c.3 program code means for receiving a second network protocol address from the second processor over the computer network.
5. The computer program product of claim 1 further comprising:
- d. program code means for processing audio data.
6. The computer program product of claim 5 wherein the program code means for processing the audio data comprises:
- program code means for compressing or decompressing the audio data.

add a2

08/22/316

ABSTRACT

A communication utility for establishing real-time, point-to-point communications between processes over a computer network includes apparatus for querying a server as to the network protocol address of another client process, and apparatus for directly establishing a communication link with the client process upon receipt of the network protocol address from the server. In one embodiment, the utility includes a sophisticated user interface having features similar to typical telephony hardware but implementing greater flexibility with software.

BAR CODE LABEL



U.S. PATENT APPLICATION

SERIAL NUMBER 08/721,316	FILING DATE 09/25/96	CLASS 395	GROUP ART UNIT 2302
---------------------------------	-----------------------------	------------------	----------------------------

APPLICANT

SHANE D. MATTAWAY, BOCA RATON, FL; GLENN W. HUTTON, MIAMI, FL; CRAIG B. STRICKLAND, TAMARAC, FL.

CONTINUING DATA***
VERIFIED THIS APPLN IS A CIP OF 08/533,115 09/25/95

FOREIGN/PCT APPLICATIONS***
VERIFIED

FOREIGN FILING LICENSE GRANTED 01/25/97

STATE OR COUNTRY FL	SHEETS DRAWING 27	TOTAL CLAIMS 6	INDEPENDENT CLAIMS 1	FILING FEE RECEIVED \$880.00	ATTORNEY DOCKET NO.
----------------------------	--------------------------	-----------------------	-----------------------------	-------------------------------------	---------------------

ADDRESS

BRUCE D JOBSE
BOOKSTEIN & KUDIRKA
ONE BEACON STREET
BOSTON MA 02108

TITLE

GRAPHIC USER INTERFACE FOR INTERNET TELEPHONY APPLICATION

This is to certify that annexed hereto is a true copy from the records of the United States Patent and Trademark Office of the application which is identified above.

By authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS

Date

Certifying Officer

PATENT APPLICATION SERIAL NO. 08/721316

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

290 SB 10/22/96 08721316
1 101 750.00 CK N0003/7002

PTO-1556
(5/87)

PATENT APPLICATION FEE DETERMINATION RECORD

Effective October 1, 1995

Application or Docket Number

0272-16

CLAIMS AS FILED - PART I

(Column 1)

(Column 2)

SMALL ENTITY

OR

OTHER THAN
SMALL ENTITY

FOR	NUMBER FILED		NUMBER EXTRA
BASIC FEE			
TOTAL CLAIMS	6	minus 20 =	*
INDEPENDENT CLAIMS	1	minus 3 =	*
MULTIPLE DEPENDENT CLAIM PRESENT			

RATE	FEE
	375.00
x\$11=	
x39=	
+125=	
TOTAL	

RATE	FEE
	750.00
x\$22=	
x78=	
+250=	
TOTAL	750

* If the difference in column 1 is less than zero, enter "0" in column 2

CLAIMS AS AMENDED - PART II

(Column 1)

(Column 2)

(Column 3)

SMALL ENTITY

OR

OTHER THAN
SMALL ENTITY

AMENDMENT A		CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
	Total	*	18	Minus	** 20
Independent	*	3	Minus	*** 3	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					

RATE	ADDITIONAL FEE
x\$11=	
x39=	
+125=	
TOTAL ADDIT. FEE	

RATE	ADDITIONAL FEE
x\$22=	
x78=	
+250=	
TOTAL ADDIT. FEE	

(Column 1)

(Column 2)

(Column 3)

AMENDMENT B		CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
	Total	*		Minus	**
Independent	*		Minus	***	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					

RATE	ADDITIONAL FEE
x\$11=	
x39=	
+125=	
TOTAL ADDIT. FEE	

RATE	ADDITIONAL FEE
x\$22=	
x78=	
+250=	
TOTAL ADDIT. FEE	

(Column 1)

(Column 2)

(Column 3)

AMENDMENT C		CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA
	Total	*		Minus	**
Independent	*		Minus	***	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					

RATE	ADDITIONAL FEE
x\$11=	
x39=	
+125=	
TOTAL ADDIT. FEE	

RATE	ADDITIONAL FEE
x\$22=	
x78=	
+250=	
TOTAL ADDIT. FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20."

*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3."

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

SCORE Placeholder Sheet for IFW Content

Application Number: **08721316**

Document Date: **09/25/1996**

The presence of this form in the IFW record indicates that the following document type was received in paper and is scanned and stored in the SCORE database.

- Drawings

Images of the original documents are scanned in gray scale or color and stored in SCORE. Bi-tonal images are also stored in IFW. Defects visible in both IFW and SCORE are indicative of defects in the original paper documents.

To access the documents in the SCORE database, refer to instructions developed by SIRA.

At the time of document entry (noted above):

- Examiners may access SCORE content via the eDAN interface.
- Other USPTO employees can bookmark the current SCORE URL (<http://es/ScoreAccessWeb/>).
- External customers may access SCORE content via the Public and Private PAIR interfaces.

Form Revision Date: December 8, 2006

08/721316

Class	709
Subclass	27
ISSUE CLASSIFICATION	

UTILITY SERIAL NUMBER	08/721316	PATENT DATE	DEC-28 1999	PATENT NUMBER	
SERIAL NUMBER	08/721,316	FILING DATE	09/25/96	SUBCLASS	27
		CLASS	709	GROUP ART UNIT	2757

6009469

 6009469

APPLICANTS SHANE D. MATTAWAY, BOCA RATON, FL; GLENN W. HUTTON, MIAMI, FL; STRICKLAND, TAMARAC, FL.

CONTINUING DATA***
 VERIFIED THIS APPLN IS A CIP OF 08/533,115 09/25/95
 PROVISIONAL APPLICATION NO. 60/025,415 09/04/96
 PROVISIONAL APPLICATION NO. 60/024,251 08/21/96

FOREIGN/PCT APPLICATIONS***
 VERIFIED

CERTIFICATE
 MAY - 8 2001
OF CORRECTION

CERTIFICATE
 MAY - 8 2001
OF CORRECTION

FOREIGN FILING LICENSE GRANTED 01/25/97

Foreign priority claimed 35 USC 119 conditions met	<input type="checkbox"/> yes <input checked="" type="checkbox"/> no	AS FILED	STATE OR COUNTRY	SHEETS DRWGS.	TOTAL CLAIMS	INDEP. CLAIMS	FILING FEE RECEIVED	ATTORNEY'S DOCKET NO.
Verified and Acknowledged	Examiner's Initials	→	FL	27	6	1	\$550.00	

BRUCE D JOSE
 BOOKSTEIN & KUDIRKA
 ONE BEACON STREET
 BOSTON MA 02108

TITLE GRAPHIC USER INTERFACE FOR INTERNET TELEPHONY APPLICATION

U.S. DEPT. OF COMM./PAT. & TM - PTO-436L (Rev.12-94)

12-6-99 Formal Drawings (27 sheets) set 8-20-99

PARTS OF APPLICATION FILED SEPARATELY		Applications Examiner	
NOTICE OF ALLOWANCE MAILED		CLAIMS ALLOWED	
Assistant Examiner		Total Claims	Print Claim
6-22-99		18	1
ISSUE FEE		DRAWING	
Amount Due	Date Paid	Sheets Drwg.	Figs. Drwg.
\$210.00	8-4-99	07	33
Label Area		ISSUE BATCH NUMBER	
		091	
PREPARED FOR ISSUE			
WARNING: The information disclosed herein may be restricted. Unauthorized disclosure may be prohibited by the United States Code Title 35, Sections 122, 181 and 368. Possession outside the U.S. Patent & Trademark Office is restricted to authorized employees and contractors only.			

Form PTO-436A (Rev. 8/92)

T.D.

STAPLE AREA

PATENT NUMBER

ORIGINAL CLASSIFICATION

CLASS

SUBCLASS

709

227

APPLICATION SERIAL NUMBER

08/221316

CROSS REFERENCE(S)

CLASS

SUBCLASS
(ONE SUBCLASS PER BLOCK)

APPLICANT'S NAME (PLEASE PRINT)

Martinez et al.

IF REISSUE, ORIGINAL PATENT NUMBER

INTERNATIONAL CLASSIFICATION

G06F

17/00

GROUP
ART UNIT

2757

ASSISTANT EXAMINER (PLEASE STAMP OR PRINT FULL NAME)

PRIMARY EXAMINER (PLEASE STAMP OR PRINT FULL NAME)

E. J. B. Kramer

PTO 270
(REV. 5-81)

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE

ISSUE CLASSIFICATION SLIP

SEARCHED			
Class	Sub.	Date	Exmr.
395	200.39	3/12/98	ASW
"	200.34	3/25/98	
"	200.35	"	
"	200.48	"	
395	200.57	3/26/98	ASW
395	200.58		
	200.20	4/15/99	Q
709	227	6/10/99	Q
	228		
370	352	6/10/99	Q
	355		
	357		
	389		
	395		

SEARCH NOTES		
Discussed Search w/ Examiner L. Barry and Prior Art Reviewed	Date	Exmr.
	3/25/98	ASW
2 Internet Searches (see inside FW)	2/15/98 3/24/98	ASW
Revised Claim for P.P.	6/10/99	Q

Note:
Check for D.P.

INTERFERENCE SEARCHED			
Class	Sub.	Date	Exmr.
709	227	6/10/99	Q

721316

UTIL
SERIAL
SERIAL
NO.

APPLICANTS
NAME
LAST

RECEIVED
DATE

FOR
VERIFICATION

FOR

Foreign priority
35 USC 119 cc

Verified and Ac

BOOKS
ONE B
BOOKS

GRAPH

TITLE

PARTS OF A

FILED SEPAR

NOTICE OF A

Amount Due

Form PTO-436A
(Rev. 8/92)

T.D.

Staple Issue. Sere

POSITION	ID NO.	DATE
CLASSIFIER		W 10-30-96
EXAMINER	277	11-25
TYPIST	502	1/25
VERIFIER	204	1-20-97
CORPS CORR.		
SPEC. HAND	416	11/21/97
FILE MAINT.	415	11-20-96
DRAFTING		

INDEX OF CLAIMS

Claim	Date
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	
37	
38	
39	
40	
41	
42	
43	
44	
45	
46	
47	
48	
49	
50	

Claim	Date
51	
52	
53	
54	
55	
56	
57	
58	
59	
60	
61	
62	
63	
64	
65	
66	
67	
68	
69	
70	
71	
72	
73	
74	
75	
76	
77	
78	
79	
80	
81	
82	
83	
84	
85	
86	
87	
88	
89	
90	
91	
92	
93	
94	
95	
96	
97	
98	
99	
100	

SYMBOLS

- ✓ Rejected
- Allowed
- (Through numeral) Canceled
- + Restricted
- N Non-elected
- I Interference
- A Appeal
- O Objected