IW 7696177

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

October 10, 2018

THIS IS TO CERTIFY THAT ANNEXED IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF THE FILE WRAPPER AND CONTENTS OF:

APPLICATION NUMBER: *10/684,776*
FILING DATE: *October 14, 2003*
PATENT NUMBER: *6,954,789*
ISSUE DATE: *October 11, 2005*

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

W. Montgomery
W. MONTGOMERY
Certifying Officer

UTILITY PATENT APPLICATION TRANSMITTAL (New Nonprovisional Applicati ns Under 37 CFR § 1.53(b))		
Attorney Docket No.	APPT-001-1-1	
First Inventor	Russell S. Dietz	
Title	METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK	
Express Mail Label No.	EV325162991US	

APPLICATION ELEMENTS

ADDRESS TO: Mail Stop Patent Application
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

1. [X] Fee transmittal form (in duplicate)
2. [ ] Applicant(s) claim(s) a small entity status.
3. [X] 67 sheet(s) of specification, claims, and abstract
4. [X] 18 sheet(s) of formal Drawing(s) with a submission letter to the Official Draftsperson
5. [X] Declaration and [X] Power of Attorney
   - a. [ ] Newly executed (original or copy)
   - b. [X] Copy from prior application for continuing application with box 18 completed
     - i. [ ] DELETION OF INVENTOR(S) signed statement attached deleting inventor(s) named in the prior application.
6. [ ] Application Data Sheet
7. [ ] CD-ROM in duplicate, large table, or computer program (Appendix)
8. [ ] Nucleotide &/or amino acid sequence submission

ACCOMPANYING APPLICATION PARTS

9. [ ] Assignment papers (cover sheet & documents)
10. [ ] 37 CFR 3.73(b) statement. [ ] Power of Attorney
11. [ ] English translation document.
12. [ ] Information Disclosure Statement (Form PTO-1449) and copies of IDS citations.
13. [X] Preliminary Amendment.
14. [X] Return Receipt postcard.
15. [ ] Certified copies of priority documents.
16. [X] Request and certification under 35 USC 122 (b)(2)(B)(i).
17. [X] Other: List of inventors, with residence city, state/country and citizenship for each

18. If a CONTINUING APPLICATION, check appropriate box, and supply the requisite information below and in a preliminary amendment, or in an Application Data Sheet under 37 CFR 1.76:

[X] Continuation . [ ] Divisional. [ ] Continuation in part (CIP). of prior application no:

Prior application information. Examiner: Khanh Q. DINH Group Art Unit: 2155

For CONTINUATION OR DIVISIONAL APPLICATIONS ONLY: the entire disclosure of the prior application, from which an oath or declaration is supplied under item 5b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

19. CORRESPONDENCE ADDRESS

[X] Customer Number:	21921.	(Name: Dov Rosenfeld, INVENTEK)	
Name:	Dov Rosenfeld,	Registration. No. :	38687
Signature:	[signature]	Date:	Oct 13th, 2003

Certificate of Mailing under 37 CFR 1.10

I hereby certify that this application and all attachments are being deposited with the United States Postal Service as Express Mail (Express Mail Label: EV325162991US in an envelope addressed to Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on.

Date: Oct ~~13~~ 14, 2003

Signed: [signature]

Name: Dov Rosenfeld, Reg. No. 38687

FEE TRANSMITTAL (New Nonprovisional Applications Under 37 CFR § 1.53(b))		***Attorney Docket No.***	APPT-001-1-1
		First Inventor	Russell S. Dietz
		Title	METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK
TOTAL PAYMENT	$1292.00 $~~750.00~~	***Express Mail Label No.***	EV325162991US

METHOD OF PAYMENT

1. ☒ The commissioner is hereby authorized to charge any missing fees and credit any overpayment to

Deposit Account Number 50-0292

Deposit Account Name INVENTEK/ROSENFELD

☐ Applicant(s) claim(s) a small entity status.

2. ☒ Payment is enclosed:

☐ check ☒ credit card. (Credit Card Charge form enclosed) ☐ Money order ☐ Other

FEE CALCULATION

CLAIMS AS FILED			OTHER THAN SMALL ENTITY	
FOR	**NO. FILED**	**NO. EXTRA**	**RATE**	**FEE**
Total Claims	49	29	$18.00	$ 522.00
Independent Claims	3	0	$86.00	$ 0.00
Multiple Dependent Claims (if applicable)				$0.00
Assignment Recording Fee				$0.00
Basic Filing Fee				$770.00
			Total Filing Fee	$1,292.00

SUBMITTED BY

☒ Customer Number:	21921.	(Dov Rosenfeld, INVENTEK)	
Name:	Dov Rosenfeld,	Registration. No. :	38687
Signature:	[signature]	Date:	Oct 13, 2003

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

INVENTOR(S)/APPLICANT(S) (New Nonprovisional Applications Under 37 CFR § 1.53(b))		
	Attorney Docket No.	APPT-001-1-1
	First Inventor	Russell S. Dietz
	Title	METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK
	Express Mail Label No.	EV325162991US

Last Name	First Name, MI	Residence (City and Either State or Foreign Country)	Citizenship
Dietz	Russell S.	San Jose, California, USA	US
Maixner	Joseph R.	Aptos, California, USA	US
Koppenhaver	Andrew A.	Littleton, CO, USA	US
Bares	William H.	Germantown, TN, USA	US
Sarkissian	Haig A.	San Antonio, Texas, USA	US
Torgerson	James F.	Andover, MN, USA	US

Our Ref./Docket No: APPT-001-1-1 Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): Dietz, *et al.* Title: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK	Group Art Unit: unassigned Examiner: unassigned

LETTER TO OFFICIAL DRAFTSPERSON SUBMISSION OF FORMAL DRAWINGS

The Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450
ATTN: Official Draftsperson

Dear Sir or Madam:

Attached please find 18 sheets of formal drawings to be made of record for the above-identified patent application submitted herewith.

Respectfully Submitted,

Oct 13, 2003
Date

Dov Rosenfeld, Reg. No. 38687

Address for correspondence and attorney for applicant(s):
Dov Rosenfeld, Reg. No. 38,687
5507 College Avenue, Suite 2
Oakland, CA 94618
Telephone: (510) 547-3378; Fax: (510) 653-7992

Certificate of Mailing under 37 CFR 1.10

I hereby certify that this application and all attachments are being deposited with the United States Postal Service as Express Mail (Express Mail Label: EV325162991US in an envelope addressed to Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on.

Date: Oct ~~13~~ 14, 2003

Signed:

Name: Dov Rosenfeld, Reg. No. 38687

04772 U.S. PTO
10/14/03

FEE TRANSMITTAL (New Nonprovisional Applications Under 37 CFR § 1.53(b))	*Attorney Docket No.*	APPT-001-1-1
	First Inventor	Russell S. Dietz
	Title	METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK
TOTAL PAYMENT $1292.00 ~~$760.00~~	*Express Mail Label No.*	EV325162991US

METHOD OF PAYMENT

1. ☒ The commissioner is hereby authorized to charge any missing fees and credit any overpayment to

Deposit Account Number: 50-0292

Deposit Account Name: INVENTEK/ROSENFELD

☐ Applicant(s) claim(s) a small entity status.

2. ☒ Payment is enclosed:

☐ check ☒ credit card. (Credit Card Charge form enclosed) ☐ Money order ☐ Other

FEE CALCULATION

CLAIMS AS FILED			OTHER THAN SMALL ENTITY	
FOR	**NO. FILED**	**NO. EXTRA**	**RATE**	**FEE**
Total Claims	49	29	$18.00	$ 522.00
Independent Claims	3	0	$86.00	$ 0.00
Multiple Dependent Claims (if applicable)				$0.00
Assignment Recording Fee				$0.00
Basic Filing Fee				$770.00
			Total Filing Fee	$1,292.00

SUBMITTED BY

☒ Customer Number:	21921.	(Dov Rosenfeld, INVENTEK)	
Name:	Dov Rosenfeld,	Registration. No. :	38687
Signature:	[signature]	Date:	Oct 13, 2003

Our Ref./Docket No.: APPT-001-1-1

METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK

Inventor(s):

DIETZ, Russell S.
San Jose, California, USA

MAIXNER, Joseph R.
Aptos, California, USA

KOPPENHAVER, Andrew A.
Littleton, CO, USA

BARES, William H.
Germantown, TN, USA

SARKISSIAN, Haig A.
San Antonio, Texas, USA

TORGERSON, James F.
Andover, MN, USA

Certificate of Mailing under 37 CFR 1.10

I hereby certify that this application and all attachments are being deposited with the United States Postal Service as Express Mail (Express Mail Label: EV325162991US in an envelope addressed to Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on.

Date: Oct ~~13~~ 14, 2003   Signed: ______________
Name: Dov Rosenfeld, Reg. No. 38687

METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This invention is a continuation of U.S. Patent Application Serial No. 09/608237 for METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK to inventors Dietz, et al., filed June 30, 2000, Attorney/Agent Reference Number APPT-001-1, the contents of which are incorporated herein by reference

[0002] This invention claims the benefit of U.S. Provisional Patent Application Serial No.: 60/141,903 for METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK to inventors Dietz, et al., filed June 30, 1999, the contents of which are incorporated herein by reference.

[0003] This application is related to the following U.S. patent applications, each filed concurrently with the present application, and each assigned to the assignee of the present invention:

[0004] U.S. Patent Application Serial No. 09/609179 for PROCESSING PROTOCOL SPECIFIC INFORMATION IN PACKETS SPECIFIED BY A PROTOCOL DESCRIPTION LANGUAGE, to inventors Koppenhaver, et al., filed June 30, 2000, Attorney/Agent Reference Number APPT-001-2, and incorporated herein by reference.

[0005] U.S. Patent Application Serial No. 09/608126 for RE-USING INFORMATION FROM DATA TRANSACTIONS FOR MAINTAINING STATISTICS IN NETWORK MONITORING, to inventors Dietz, et al., filed June 30, 2000, Attorney/Agent Reference Number APPT-001-3, and incorporated herein by reference.

[0006] U.S. Patent Application Serial No. 09/608266 for ASSOCIATIVE CACHE STRUCTURE FOR LOOKUPS AND UPDATES OF FLOW RECORDS IN A NETWORK MONITOR, to inventors Sarkissian, et al., filed June 30, 2000, Attorney/Agent Reference Number APPT-001-4, and incorporated herein by reference.

[0007] U.S. Patent Application Serial No. 09/608267 for STATE PROCESSOR FOR PATTERN MATCHING IN A NETWORK MONITOR DEVICE, to inventors Sarkissian, et al., filed June 30, 2000, Attorney/Agent Reference Number APPT-001-5, and incorporated herein by reference.

FIELD OF INVENTION

[0008] The present invention relates to computer networks, specifically to the real-time elucidation of packets communicated within a data network, including classification according to protocol and application program.

BACKGROUND TO THE PRESENT INVENTION

[0009] There has long been a need for network activity monitors. This need has become especially acute, however, given the recent popularity of the Internet and other internets—an "internet" being any plurality of interconnected networks which forms a larger, single network. With the growth of networks used as a collection of clients obtaining services from one or more servers on the network, it is increasingly important to be able to monitor the use of those services and to rate them accordingly. Such objective information, for example, as which services (*i.e.,* application programs) are being used, who is using them, how often they have been accessed, and for how long, is very useful in the maintenance and continued operation of these networks. It is especially important that selected users be able to access a network remotely in order to generate reports on network use in real time. Similarly, a need exists for a real-time network monitor that can provide alarms notifying selected users of problems that may occur with the network or site.

[0010] One prior art monitoring method uses log files. In this method, selected network activities may be analyzed retrospectively by reviewing log files, which are maintained by network servers and gateways. Log file monitors must access this data and analyze ("mine") its contents to determine statistics about the server or gateway. Several problems exist with this method, however. First, log file information does not provide a map of real-time usage; and secondly, log file mining does not supply complete information. This method relies on logs maintained by numerous network devices and servers, which requires that the information be subjected to refining and correlation. Also, sometimes information is simply not available to any gateway or server in order to make a log file entry.

[0011] One such case, for example, would be information concerning NetMeeting® (Microsoft Corporation, Redmond, Washington) sessions in which two computers connect directly on the network and the data is never seen by a server or a gateway.

[0012] Another disadvantage of creating log files is that the process requires data logging features of network elements to be enabled, placing a substantial load on the device , which results in a subsequent decline in network performance. Additionally, log files can grow rapidly, there is no standard means of storage for them, and they require a significant amount of maintenance.

[0013] Though Netflow® (Cisco Systems, Inc., San Jose, California), RMON2, and other network monitors are available for the real-time monitoring of networks, they lack visibility into application content and are typically limited to providing network layer level information.

[0014] Pattern-matching parser techniques wherein a packet is parsed and pattern filters are applied are also known, but these too are limited in how deep into the protocol stack they can examine packets.

[0015] Some prior art packet monitors classify packets into connection flows. The term "connection flow" is commonly used to describe all the packets involved with a single connection. A conversational flow, on the other hand, is the sequence of packets that are exchanged in any direction as a result of an activity—for instance, the running of an application on a server as requested by a client. It is desirable to be able to identify and classify conversational flows rather than only connection flows. The reason for this is that some conversational flows involve more than one connection, and some even involve more than one exchange of packets between a client and server. This is particularly true when using client/server protocols such as RPC, DCOMP, and SAP, which enable a service to be set up or defined prior to any use of that service.

[0016] An example of such a case is the SAP (Service Advertising Protocol), a NetWare (Novell Systems, Provo, Utah) protocol used to identify the services and addresses of servers attached to a network. In the initial exchange, a client might send a SAP request to a server for print service. The server would then send a SAP reply that identifies a particular address—for example, SAP#5—as the print service on that server. Such responses might be used to update a table in a router, for instance, known as a Server Information Table. A client who has inadvertently seen this reply or who has access to the table (via the router that has the Service Information Table) would know that SAP#5 for this particular server is a print service. Therefore, in order to print data on the server, such a client would not need to make a request for a print service, but would simply send data to be printed specifying SAP#5. Like the previous exchange, the transmission of data to be printed also involves an exchange between a client and a server, but requires a second connection and is therefore independent of the initial exchange. In order to eliminate the possibility of disjointed conversational exchanges, it is desirable for a network packet monitor to be able to "virtually concatenate"—that is, to link—the first exchange with the second. If the clients were the same, the two packet exchanges would then be correctly identified as being part of the same conversational flow.

[0017] Other protocols that may lead to disjointed flows, include RPC (Remote Procedure Call); DCOM (Distributed Component Object Model), formerly called Network OLE (Microsoft Corporation, Redmond, Washington); and CORBA (Common Object Request Broker Architecture). RPC is a programming interface from Sun Microsystems (Palo Alto, California) that allows one program to use the services of another program in a remote machine. DCOM, Microsoft's counterpart to CORBA, defines the remote procedure call that allows those objects—objects are self-contained software modules—to be run remotely over the network. And CORBA, a standard from the Object Management Group (OMG) for communicating between distributed objects, provides a way to execute programs (objects) written in different programming languages running on different platforms regardless of where they reside in a network.

[0018] What is needed, therefore, is a network monitor that makes it possible to continuously analyze all user sessions on a heavily trafficked network. Such a monitor should enable non-intrusive, remote detection, characterization, analysis, and capture of all information passing through any point on the network (*i.e.,* of all packets and packet streams passing through any location in the network). Not only should all the packets be detected and analyzed, but for each of these packets the network monitor should determine the protocol (*e.g.,* http, ftp, H.323, VPN, etc.), the application/use within the protocol (*e.g.,* voice, video, data, real-time data, etc.), and an end user's pattern of use within each application or the application context (*e.g.,* options selected, service delivered, duration, time of day, data requested, etc.). Also, the network monitor should not be reliant upon server resident information such as log files. Rather, it should allow a user such as a network administrator or an Internet service provider (ISP) the means to measure and analyze network activity objectively; to customize the type of data that is collected and analyzed; to undertake real time analysis; and to receive timely notification of network problems.

[0019] Considering the previous SAP example again, because one features of the invention is to correctly identify the second exchange as being associated with a print service on that server, such exchange would even be recognized if the clients were not the same. What distinguishes this invention from prior art network monitors is that it has the ability to recognize disjointed flows as belonging to the same conversational flow.

[0020] The data value in monitoring network communications has been recognized by many inventors. Chiu, *et al.*, describe a method for collecting information at the session level in a computer network in United States Patent 5,101,402, titled "APPARATUS AND METHOD FOR REAL-TIME MONITORING OF NETWORK SESSIONS AND A LOCAL AREA NETWORK" (the "402 patent"). The 402 patent specifies fixed locations for particular types of packets to extract information to identify session of a packet. For example, if a DECnet packet appears, the 402 patent looks at six specific fields (at 6 locations) in the packet in order to identify the session of the packet. If, on the other hand, an IP packet appears, a different set of six different locations is specified for an IP packet. With the proliferation of protocols, clearly the specifying of all the possible places to look to determine the session becomes more and more difficult. Likewise, adding a new protocol or application is difficult. In the present invention, the locations examined and the information extracted from any packet are adaptively determined from information in the packet for the particular type of packet. There is no fixed definition of what to look for and where to look in order to form an identifying signature. A monitor implementation of the present invention, for example, adapts to handle differently IEEE 802.3 packet from the older Ethernet Type 2 (or Version 2) DIX (Digital-Intel-Xerox) packet.

[0021] The 402 patent system is able to recognize up to the session layer. In the present invention, the number of levels examined varies for any particular protocol. Furthermore, the present invention is capable of examining up to whatever level is sufficient to uniquely identify to a required level, even all the way to the application level (in the OSI model).

[0022] Other prior art systems also are known. Phael describes a network activity monitor that processes only randomly selected packets in United States Patent 5,315,580, titled "NETWORK MONITORING DEVICE AND SYSTEM." Nakamura teaches a network monitoring system in United States Patent 4,891,639, titled "MONITORING SYSTEM OF NETWORK." Ross, *et al.*, teach a method and apparatus for analyzing and monitoring network activity in United States Patent 5,247,517, titled "METHOD AND APPARATUS FOR ANALYSIS NETWORKS," McCreery, *et al.*, describe an Internet activity monitor that decodes packet data at the Internet protocol level layer in United States Patent 5,787,253, titled "APPARATUS AND METHOD OF ANALYZING INTERNET ACTIVITY." The McCreery method decodes IP-packets. It goes through the decoding operations for each packet, and therefore uses the processing overhead for both recognized and unrecognized flows. In a monitor implementation of the present invention, a signature is built for every flow such that future packets of the flow are easily recognized. When a new packet in the flow arrives, the recognition process can commence from where it last left off, and a new signature built to recognize new packets of the flow.

SUMMARY

[0023] In its various embodiments the present invention provides a network monitor that can accomplish one or more of the following objects and advantages:

[0024] • Recognize and classify all packets that are exchanges between a client and server into respective client/server applications.

[0025] • Recognize and classify at all protocol layer levels conversational flows that pass in either direction at a point in a network.

[0026] • Determine the connection and flow progress between clients and servers according to the individual packets exchanged over a network.

[0027] • Be used to help tune the performance of a network according to the current mix of client/server applications requiring network resources.

[0028] • Maintain statistics relevant to the mix of client/server applications using network resources.

[0029] • Report on the occurrences of specific sequences of packets used by particular applications for client/server network conversational flows.

[0030] Other aspects of embodiments of the invention are:

[0031] • Properly analyzing each of the packets exchanged between a client and a server and maintaining information relevant to the current state of each of these conversational flows.

[0032] • Providing a flexible processing system that can be tailored or adapted as new applications enter the client/server market.

[0033] • Maintaining statistics relevant to the conversational flows in a client/sever network as classified by an individual application.

[0034] • Reporting a specific identifier, which may be used by other network-oriented devices to identify the series of packets with a specific application for a specific client/server network conversational flow.

[0035] In general, the embodiments of the present invention overcome the problems and disadvantages of the art.

[0036] As described herein, one embodiment analyzes each of the packets passing through any point in the network in either direction, in order to derive the actual application used to communicate between a client and a server. Note that there could be several simultaneous and overlapping applications executing over the network that are independent and asynchronous.

[0037] A monitor embodiment of the invention successfully classifies each of the individual packets as they are seen on the network. The contents of the packets are parsed and selected parts are assembled into a signature (also called a key) that may then be used identify further packets of the same conversational flow, for example to further analyze the flow and ultimately to recognize the application program. Thus the key is a function of the selected parts, and in the preferred embodiment, the function is a concatenation of the selected parts. The preferred embodiment forms and remembers the state of any conversational flow, which is determined by the relationship between individual packets and the entire conversational flow over the network. By remembering the state of a flow in this way, the embodiment determines the context of the conversational flow, including the application program it relates to and parameters such as the time, length of the conversational flow, data rate, etc.

[0038] The monitor is flexible to adapt to future applications developed for client/server networks. New protocols and protocol combinations may be incorporated by compiling files written in a high-level protocol description language.

[0039] The monitor embodiment of the present invention is preferably implemented in application-specific integrated circuits (ASIC) or field programmable gate arrays (FPGA). In one embodiment, the monitor comprises a parser subsystem that forms a signature from a packet. The monitor further comprises an analyzer subsystem that receives the signature from the parser subsystem.

[0040] A packet acquisition device such as a media access controller (MAC) or a segmentation and reassemble module is used to provide packets to the parser subsystem of the monitor.

[0041] In a hardware implementation, the parsing subsystem comprises two sub-parts, the pattern analysis and recognition engine (PRE), and an extraction engine (slicer). The PRE interprets each packet, and in particular, interprets individual fields in each packet according to a pattern database.

[0042] The different protocols that can exist in different layers may be thought of as nodes of one or more trees of linked nodes. The packet type is the root of a tree. Each protocol is either a parent node or a terminal node. A parent node links a protocol to other protocols (child protocols) that can be at higher layer levels. For example, An Ethernet packet (the root node) may be an Ethertype packet—also called an Ethernet Type/Version 2 and a DIX (DIGITAL-Intel-Xerox packet)—or an IEEE 802.3 packet. Continuing with the IEEE 802.3-type packet, one of the children nodes may be the IP protocol, and one of the children of the IP protocol may be the TCP protocol.

[0043] The pattern database includes a description of the different headers of packets and their contents, and how these relate to the different nodes in a tree. The PRE traverses the tree as far as it can. If a node does not include a link to a deeper level, pattern matching is declared complete. Note that protocols can be the children of several parents. If a unique node was generated for each of the possible parent/child trees, the pattern database might become excessively large. Instead, child nodes are shared among multiple parents, thus compacting the pattern database.

[0044] Finally the PRE can be used on its own when only protocol recognition is required.

[0045] For each protocol recognized, the slicer extracts important packet elements from the packet. These form a signature (*i.e.,* key) for the packet. The slicer also preferably generates a hash for rapidly identifying a flow that may have this signature from a database of known flows.

[0046] The flow signature of the packet, the hash and at least some of the payload are passed to an analyzer subsystem. In a hardware embodiment, the analyzer subsystem includes a unified flow key buffer (UFKB) for receiving parts of packets from the parser subsystem and for storing signatures in process, a lookup/update engine (LUE) to lookup a database of flow records for previously encountered conversational flows to determine whether a signature is from an existing flow, a state processor (SP) for performing state processing, a flow insertion and deletion engine (FIDE) for inserting new flows into the database of flows, a memory for storing the database of flows, and a cache for speeding up access to the memory containing the flow database. The LUE, SP, and FIDE are all coupled to the UFKB, and to the cache.

[0047] The unified flow key buffer thus contains the flow signature of the packet, the hash and at least some of the payload for analysis in the analyzer subsystem. Many operations can be performed to further elucidate the identity of the application program content of the packet involved in the client/server conversational flow while a packet signature exists in the unified flow signature buffer. In the particular hardware embodiment of the analyzer subsystem several flows may be processed in parallel, and multiple flow signatures from all the packets being analyzed in parallel may be held in the one UFKB.

[0048] The first step in the packet analysis process of a packet from the parser subsystem is to lookup the instance in the current database of known packet flow signatures. A lookup/update engine (LUE) accomplishes this task using first the hash, and then the flow signature. The search is carried out in the cache and if there is no flow with a matching signature in the cache, the lookup engine attempts to retrieve the flow from the flow database in the memory. The flow-entry for previously encountered flows preferably includes state information, which is used in the state processor to execute any operations defined for the state, and to determine the next state. A typical state operation may be to search for one or more known reference strings in the payload of the packet stored in the UFKB.

[0049] Once the lookup processing by the LUE has been completed a flag stating whether it is found or is new is set within the unified flow signature buffer structure for this packet flow signature. For an existing flow, the flow-entry is updated by a calculator component of the LUE that adds values to counters in the flow-entry database used to store one or more statistical measures of the flow. The counters are used for determining network usage metrics on the flow.

[0050] After the packet flow signature has been looked up and contents of the current flow signature are in the database, a state processor can begin analyzing the packet payload to further elucidate the identity of the application program component of this packet. The exact operation of the state processor and functions performed by it will vary depending on the current packet sequence in the stream of a conversational flow. The state processor moves to the next logical operation stored from the previous packet seen with this same flow signature. If any processing is required on this packet, the state processor will execute instructions from a database of state instruction for this state until there are either no more left or the instruction signifies processing.

[0051] In the preferred embodiment, the state processor functions are programmable to provide for analyzing new application programs, and new sequences of packets and states that can arise from using such application.

[0052] If during the lookup process for this particular packet flow signature, the flow is required to be inserted into the active database, a flow insertion and deletion engine (FIDE) is initiated. The state processor also may create new flow signatures and thus may instruct the flow insertion and deletion engine to add a new flow to the database as a new item.

[0053] In the preferred hardware embodiment, each of the LUE, state processor, and FIDE operate independently from the other two engines.

BRIEF DESCRIPTION OF THE DRAWINGS

[0054] Although the present invention is better understood by referring to the detailed preferred embodiments, these should not be taken to limit the present invention to any specific embodiment because such embodiments are provided only for the purposes of explanation. The embodiments, in turn, are explained with the aid of the following figures.

[0055] FIG. 1 is a functional block diagram of a network embodiment of the present invention in which a monitor is connected to analyze packets passing at a connection point.

[0056] FIG. 2 is a diagram representing an example of some of the packets and their formats that might be exchanged in starting, as an illustrative example, a conversational flow between a client and server on a network being monitored and analyzed. A pair of flow signatures particular to this example and to embodiments of the present invention is also illustrated. This represents some of the possible flow signatures that can be generated and used in the process of analyzing packets and of recognizing the particular server applications that produce the discrete application packet exchanges.

[0057] FIG. 3 is a functional block diagram of a process embodiment of the present invention that can operate as the packet monitor shown in FIG. 1. This process may be implemented in software or hardware.

[0058] FIG. 4 is a flowchart of a high-level protocol language compiling and optimization process, which in one embodiment may be used to generate data for monitoring packets according to versions of the present invention.

[0059] FIG. 5 is a flowchart of a packet parsing process used as part of the parser in an embodiment of the inventive packet monitor.

[0060] FIG. 6 is a flowchart of a packet element extraction process that is used as part of the parser in an embodiment of the inventive packet monitor.

[0061] FIG. 7 is a flowchart of a flow-signature building process that is used as part of the parser in the inventive packet monitor.

[0062] FIG. 8 is a flowchart of a monitor lookup and update process that is used as part of the analyzer in an embodiment of the inventive packet monitor.

[0063] FIG. 9 is a flowchart of an exemplary Sun Microsystems Remote Procedure Call application than may be recognized by the inventive packet monitor.

[0064] FIG. 10 is a functional block diagram of a hardware parser subsystem including the pattern recognizer and extractor that can form part of the parser module in an embodiment of the inventive packet monitor.

[0065] FIG. 11 is a functional block diagram of a hardware analyzer including a state processor that can form part of an embodiment of the inventive packet monitor.

[0066] FIG. 12 is a functional block diagram of a flow insertion and deletion engine process that can form part of the analyzer in an embodiment of the inventive packet monitor.

[0067] FIG. 13 is a flowchart of a state processing process that can form part of the analyzer in an embodiment of the inventive packet monitor.

[0068] FIG. 14 is a simple functional block diagram of a process embodiment of the present invention that can operate as the packet monitor shown in FIG. 1. This process may be implemented in software.

[0069] FIG. 15 is a functional block diagram of how the packet monitor of FIG. 3 (and FIGS. 10 and 11) may operate on a network with a processor such as a microprocessor.

[0070] FIG. 16 is an example of the top (MAC) layer of an Ethernet packet and some of the elements that may be extracted to form a signature according to one aspect of the invention.

[0071] FIG. 17A is an example of the header of an Ethertype type of Ethernet packet of FIG. 16 and some of the elements that may be extracted to form a signature according to one aspect of the invention.

[0072] FIG. 17B is an example of an IP packet, for example, of the Ethertype packet shown in FIGs. 16 and 17A, and some of the elements that may be extracted to form a signature according to one aspect of the invention.

[0073] FIG. 18A is a three dimensional structure that can be used to store elements of the pattern, parse and extraction database used by the parser subsystem in accordance to one embodiment of the invention.

[0074] FIG. 18B is an alternate form of storing elements of the pattern, parse and extraction database used by the parser subsystem in accordance to another embodiment of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0075] Note that this document includes hardware diagrams and descriptions that may include signal names. In most cases, the names are sufficiently descriptive, in other cases however the signal names are not needed to understand the operation and practice of the invention.

Operation in a Network

[0076] FIG. 1 represents a system embodiment of the present invention that is referred to herein by the general reference numeral 100. The system 100 has a computer network 102 that communicates packets (*e.g.*, IP datagrams) between various computers, for example between the clients 104–107 and servers 110 and 112. The network is shown schematically as a cloud with several network nodes and links shown in the interior of the cloud. A monitor 108 examines the packets passing in either direction past its connection point 121 and, according to one aspect of the invention, can elucidate what application programs are associated with each packet. The monitor 108 is shown examining packets (*i.e.*, datagrams) between the network interface 116 of the server 110 and the network. The monitor can also be placed at other points in the network, such as connection point 123 between the network 102 and the interface 118 of the client 104, or some other location, as indicated schematically by connection point 125 somewhere in network 102. Not shown is a network packet acquisition device at the location 123 on the network for converting the physical information on the network into packets for input into monitor 108. Such packet acquisition devices are common.

[0077] Various protocols may be employed by the network to establish and maintain the required communication, *e.g.,* TCP/IP, etc. Any network activity—for example an application program run by the client 104 (CLIENT 1) communicating with another running on the server 110 (SERVER 2)—will produce an exchange of a sequence of packets over network 102 that is characteristic of the respective programs and of the network protocols. Such characteristics may not be completely revealing at the individual packet level. It may require the analyzing of many packets by the monitor 108 to have enough information needed to recognize particular application programs. The packets may need to be parsed then analyzed in the context of various protocols, for example, the transport through the application session layer protocols for packets of a type conforming to the ISO layered network model.

[0078] Communication protocols are layered, which is also referred to as a protocol stack. The ISO (International Standardization Organization) has defined a general model that provides a framework for design of communication protocol layers. This model, shown in table form below, serves as a basic reference for understanding the functionality of existing communication protocols.

ISO MODEL

Layer	Functionality	Example
7	Application	Telnet, NFS, Novell NCP, HTTP, H.323
6	Presentation	XDR
5	Session	RPC, NETBIOS, SNMP, *etc.*
4	Transport	TCP, Novel SPX, UDP, *etc.*
3	Network	IP, Novell IPX, VIP, AppleTalk, *etc.*
2	Data Link	Network Interface Card (Hardware Interface). MAC layer
1	Physical	Ethernet, Token Ring, Frame Relay, ATM, T1 (Hardware Connection)

[0079] Different communication protocols employ different levels of the ISO model or may use a layered model that is similar to but which does not exactly conform to the ISO model. A protocol in a certain layer may not be visible to protocols employed at other layers. For example, an application (Level 7) may not be able to identify the source computer for a communication attempt (Levels 2–3).

[0080] In some communication arts, the term "frame" generally refers to encapsulated data at OSI layer 2, including a destination address, control bits for flow control, the data or payload, and CRC (cyclic redundancy check) data for error checking. The term "packet" generally refers to encapsulated data at OSI layer 3. In the TCP/IP world, the term "datagram" is also used. In this specification, the term "packet" is intended to encompass packets, datagrams, frames, and cells. In general, a packet format or frame format refers to how data is encapsulated with various fields and headers for transmission across a network. For example, a data packet typically includes an address destination field, a length field, an error correcting code (ECC) field, or cyclic redundancy check (CRC) field, as well as headers and footers to identify the beginning and end of the packet. The terms "packet format" and "frame format," also referred to as "cell format," are generally synonymous.

[0081] Monitor 108 looks at every packet passing the connection point 121 for analysis. However, not every packet carries the same information useful for recognizing all levels of the protocol. For example, in a conversational flow associated with a particular application, the application will cause the server to send a type-A packet, but so will another. If, though, the particular application program always follows a type-A packet with the sending of a type-B packet, and the other application program does not, then in order to recognize packets of that application's conversational flow, the monitor can be available to recognize packets that match the type-B packet to associate with the type-A packet. If such is recognized after a type-A packet, then the particular application program's conversational flow has started to reveal itself to the monitor 108.

[0082] Further packets may need to be examined before the conversational flow can be identified as being associated with the application program. Typically, monitor 108 is simultaneously also in partial completion of identifying other packet exchanges that are parts of conversational flows associated with other applications. One aspect of monitor 108 is its ability to maintain the state of a flow. The state of a flow is an indication of all previous events in the flow that lead to recognition of the content of all the protocol levels, *e.g.*, the ISO model protocol levels. Another aspect of the invention is forming a signature of extracted characteristic portions of the packet that can be used to rapidly identify packets belonging to the same flow.

[0083] In real-world uses of the monitor 108, the number of packets on the network 102 passing by the monitor 108's connection point can exceed a million per second. Consequently, the monitor has very little time available to analyze and type each packet and identify and maintain the state of the flows passing through the connection point. The monitor 108 therefore masks out all the unimportant parts of each packet that will not contribute to its classification. However, the parts to mask-out will change with each packet depending on which flow it belongs to and depending on the state of the flow.

[0084] The recognition of the packet type, and ultimately of the associated application programs according to the packets that their executions produce, is a multi-step process within the monitor 108. At a first level, for example, several application programs will all produce a first kind of packet. A first "signature" is produced from selected parts of a packet that will allow monitor 108 to identify efficiently any packets that belong to the same flow. In some cases, that packet type may be sufficiently unique to enable the monitor to identify the application that generated such a packet in the conversational flow. The signature can then be used to efficiently identify all future packets generated in traffic related to that application.

[0085] In other cases, that first packet only starts the process of analyzing the conversational flow, and more packets are necessary to identify the associated application program. In such a case, a subsequent packet of a second type—but that potentially belongs to the same conversational flow—is recognized by using the signature. At such a second level, then, only a few of those application programs will have conversational flows that can produce such a second packet type. At this level in the process of classification, all application programs that are not in the set of those that lead to such a sequence of packet types may be excluded in the process of classifying the conversational flow that includes these two packets. Based on the known patterns for the protocol and for the possible applications, a signature is produced that allows recognition of any future packets that may follow in the conversational flow.

[0086] It may be that the application is now recognized, or recognition may need to proceed to a third level of analysis using the second level signature. For each packet, therefore, the monitor parses the packet and generates a signature to determine if this signature identified a previously encountered flow, or shall be used to recognize future packets belonging to the same conversational flow. In real time, the packet is further analyzed in the context of the sequence of previously encountered packets (the state), and of the possible future sequences such a past sequence may generate in conversational flows associated with different applications. A new signature for recognizing future packets may also be generated. This process of analysis continues until the applications are identified. The last generated signature may then be used to efficiently recognize future packets associated with the same conversational flow. Such an arrangement makes it possible for the monitor 108 to cope with millions of packets per second that must be inspected.

[0087] Another aspect of the invention is adding Eavesdropping. In alternative embodiments of the present invention capable of eavesdropping, once the monitor 108 has recognized the executing application programs passing through some point in the network 102 (for example, because of execution of the applications by the client 105 or server 110), the monitor sends a message to some general purpose processor on the network that can input the same packets from the same location on the network, and the processor then loads its own executable copy of the application program and uses it to read the content being exchanged over the network. In other words, once the monitor 108 has accomplished recognition of the application program, eavesdropping can commence.

The Network Monitor

[0088] FIG. 3 shows a network packet monitor 300, in an embodiment of the present invention that can be implemented with computer hardware and/or software. The system 300 is similar to monitor 108 in FIG. 1. A packet 302 is examined, *e.g.,* from a packet acquisition device at the location 121 in network 102 (FIG. 1), and the packet evaluated, for example in an attempt to determine its characteristics, *e.g.,* all the protocol information in a multilevel model, including what server application produced the packet.

[0089] The packet acquisition device is a common interface that converts the physical signals and then decodes them into bits, and into packets, in accordance with the particular network

(Ethernet, frame relay, ATM, *etc.*). The acquisition device indicates to the monitor 108 the type of network of the acquired packet or packets.

[0090] Aspects shown here include: (1) the initialization of the monitor to generate what operations need to occur on packets of different types—accomplished by compiler and optimizer 310, (2) the processing—parsing and extraction of selected portions—of packets to generate an identifying signature—accomplished by parser subsystem 301, and (3) the analysis of the packets—accomplished by analyzer 303.

[0091] The purpose of compiler and optimizer 310 is to provide protocol specific information to parser subsystem 301 and to analyzer subsystem 303. The initialization occurs prior to operation of the monitor, and only needs to re-occur when new protocols are to be added.

[0092] A flow is a stream of packets being exchanged between any two addresses in the network. For each protocol there are known to be several fields, such as the destination (recipient), the source (the sender), and so forth, and these and other fields are used in monitor 300 to identify the flow. There are other fields not important for identifying the flow, such as checksums, and those parts are not used for identification.

[0093] Parser subsystem 301 examines the packets using pattern recognition process 304 that parses the packet and determines the protocol types and associated headers for each protocol layer that exists in the packet 302. An extraction process 306 in parser subsystem 301 extracts characteristic portions (signature information) from the packet 302. Both the pattern information for parsing and the related extraction operations, *e.g.,* extraction masks, are supplied from a parsing-pattern-structures and extraction-operations database (parsing/extractions database) 308 filled by the compiler and optimizer 310.

[0094] The protocol description language (PDL) files 336 describes both patterns and states of all protocols that an occur at any layer, including how to interpret header information, how to determine from the packet header information the protocols at the next layer, and what information to extract for the purpose of identifying a flow, and ultimately, applications and services. The layer selections database 338 describes the particular layering handled by the monitor. That is, what protocols run on top of what protocols at any layer level. Thus 336 and 338 combined describe how one would decode, analyze, and understand the information in packets, and, furthermore, how the information is layered. This information is input into compiler and optimizer 310.

[0095] When compiler and optimizer 310 executes, it generates two sets of internal data structures. The first is the set of parsing/extraction operations 308. The pattern structures include parsing information and describe what will be recognized in the headers of packets; the extraction operations are what elements of a packet are to be extracted from the packets based on the patterns that get matched. Thus, database 308 of parsing/extraction operations includes information describing how to determine a set of one or more protocol dependent extraction operations from data in the packet that indicate a protocol used in the packet.

[0096] The other internal data structure that is built by compiler 310 is the set of state patterns and processes 326. These are the different states and state transitions that occur in different conversational flows, and the state operations that need to be performed (*e.g.*, patterns that need to be examined and new signatures that need to be built) during any state of a conversational flow to further the task of analyzing the conversational flow.

[0097] Thus, compiling the PDL files and layer selections provides monitor 300 with the information it needs to begin processing packets. In an alternate embodiment, the contents of one or more of databases 308 and 326 may be manually or otherwise generated. Note that in some embodiments the layering selections information is inherent rather than explicitly described. For example, since a PDL file for a protocol includes the child protocols, the parent protocols also may be determined.

[0098] In the preferred embodiment, the packet 302 from the acquisition device is input into a packet buffer. The pattern recognition process 304 is carried out by a pattern analysis and recognition (PAR) engine that analyzes and recognizes patterns in the packets. In particular, the PAR locates the next protocol field in the header and determines the length of the header, and may perform certain other tasks for certain types of protocol headers. An example of this is type and length comparison to distinguish an IEEE 802.3 (Ethernet) packet from the older type 2 (or Version 2) Ethernet packet, also called a DIGITAL-Intel-Xerox (DIX) packet. The PAR also uses the pattern structures and extraction operations database 308 to identify the next protocol and parameters associated with that protocol that enables analysis of the next protocol layer. Once a pattern or a set of patterns has been identified, it/they will be associated with a set of none or more extraction operations. These extraction operations (in the form of commands and associated parameters) are passed to the extraction process 306 implemented by an extracting and information identifying (EII) engine that extracts selected parts of the packet, including identifying information from the packet as required for recognizing this packet as part of a flow. The extracted information is put in sequence and then processed in block 312 to build a unique flow signature (also called a "key") for this flow. A flow signature depends on the protocols used in the packet. For some protocols, the extracted components may include source and destination addresses. For example, Ethernet frames have end-point addresses that are useful in building a better flow signature. Thus, the signature typically includes the client and server address pairs. The signature is used to recognize further packets that are or may be part of this flow.

[0099] In the preferred embodiment, the building of the flow key includes generating a hash of the signature using a hash function. The purpose if using such a hash is conventional—to spread flow-entries identified by the signature across a database for efficient searching. The hash generated is preferably based on a hashing algorithm and such hash generation is known to those in the art.

[00100] In one embodiment, the parser passes data from the packet—a parser record—that includes the signature (i.e., selected portions of the packet), the hash, and the packet itself to allow for any state processing that requires further data from the packet. An improved embodiment of the parser subsystem might generate a parser record that has some predefined structure and that includes the signature, the hash, some flags related to some of the fields in the parser record, and parts of the packet's payload that the parser subsystem has determined might be required for further processing, e.g., for state processing.

[00101] Note that alternate embodiments may use some function other than concatenation of the selected portions of the packet to make the identifying signature. For example, some "digest function" of the concatenated selected portions may be used.

[00102] The parser record is passed onto lookup process 314 which looks in an internal data store of records of known flows that the system has already encountered, and decides (in 316) whether or not this particular packet belongs to a known flow as indicated by the presence of a flow-entry matching this flow in a database of known flows 324. A record in database 324 is associated with each encountered flow.

[00103] The parser record enters a buffer called the unified flow key buffer (UFKB). The UFKB stores the data on flows in a data structure that is similar to the parser record, but that includes a field that can be modified. In particular, one or the UFKB record fields stores the packet sequence number, and another is filled with state information in the form of a program counter for a state processor that implements state processing 328.

[00104] The determination (316) of whether a record with the same signature already exists is carried out by a lookup engine (LUE) that obtains new UFKB records and uses the hash in the UFKB record to lookup if there is a matching known flow. In the particular embodiment, the database of known flows 324 is in an external memory. A cache is associated with the database 324. A lookup by the LUE for a known record is carried out by accessing the cache using the hash, and if the entry is not already present in the cache, the entry is looked up (again using the hash) in the external memory.

[00105] The flow-entry database 324 stores flow-entries that include the unique flow-signature, state information, and extracted information from the packet for updating flows, and one or more statistical about the flow. Each entry completely describes a flow. Database 324 is organized into bins that contain a number, denoted N, of flow-entries (also called flow-entries, each a bucket), with N being 4 in the preferred embodiment. Buckets (i.e., flow-entries) are accessed via the hash of the packet from the parser subsystem 301 (i.e., the hash in the UFKB record). The hash spreads the flows across the database to allow for fast lookups of entries, allowing shallower buckets. The designer selects the bucket depth N based on the amount of memory attached to the monitor, and the number of bits of the hash data value used. For example, in one embodiment, each flow-entry is 128 bytes long, so for 128K flow-entries, 16 Mbytes are required. Using a 16-bit hash gives two flow-entries per bucket. Empirically, this has been shown to be more than adequate for the vast majority of cases. Note that another embodiment uses flow-entries that are 256 bytes long.

[00106] Herein, whenever an access to database 324 is described, it is to be understood that the access is via the cache, unless otherwise stated or clear from the context.

[00107] If there is no flow-entry found matching the signature, i.e., the signature is for a new flow, then a protocol and state identification process 318 further determines the state and protocol. That is, process 318 determines the protocols and where in the state sequence for a flow for this protocol's this packet belongs. Identification process 318 uses the extracted information and makes reference to the database 326 of state patterns and processes. Process 318 is then followed by any state operations that need to be executed on this packet by a state processor 328.

[00108] If the packet is found to have a matching flow-entry in the database 324 (e.g., in the cache), then a process 320 determines, from the looked-up flow-entry, if more classification by state processing of the flow signature is necessary. If not, a process 322 updates the flow-entry in the flow-entry database 324 (e.g., via the cache). Updating includes updating one or more statistical measures stored in the flow-entry. In our embodiment, the statistical measures are stored in counters in the flow-entry.

[00109] If state processing is required, state process 328 is commenced. State processor 328 carries out any state operations specified for the state of the flow and updates the state to the next state according to a set of state instructions obtained form the state pattern and processes database 326.

[00110] The state processor 328 analyzes both new and existing flows in order to analyze all levels of the protocol stack, ultimately classifying the flows by application (level 7 in the ISO model). It does this by proceeding from state-to-state based on predefined state transition rules and state operations as specified in state processor instruction database 326. A state transition rule is a rule typically containing a test followed by the next-state to proceed to if the test result is true. An operation is an operation to be performed while the state processor is in a particular state—for example, in order to evaluate a quantity needed to apply the state transition rule. The state processor goes through each rule and each state process until the test is true, or there are no more tests to perform.

[00111] In general, the set of state operations may be none or more operations on a packet, and carrying out the operation or operations may leave one in a state that causes exiting the system prior to completing the identification, but possibly knowing more about what state and state processes are needed to execute next, *i.e.*, when a next packet of this flow is encountered. As an example, a state process (set of state operations) at a particular state may build a new signature for future recognition packets of the next state.

[00112] By maintaining the state of the flows and knowing that new flows may be set up using the information from previously encountered flows, the network traffic monitor 300 provides for (a) single-packet protocol recognition of flows, and (b) multiple-packet protocol recognition of flows. Monitor 300 can even recognize the application program from one or more disjointed sub-flows that occur in server announcement type flows. What may seem to prior art monitors to be some unassociated flow, may be recognized by the inventive monitor using the flow signature to be a sub-flow associated with a previously encountered sub-flow.

[00113] Thus, state processor 328 applies the first state operation to the packet for this particular flow-entry. A process 330 decides if more operations need to be performed for this state. If so, the analyzer continues looping between block 330 and 328 applying additional state operations to this particular packet until all those operations are completed—that is, there are no more operations for this packet in this state. A process 332 decides if there are further states to be analyzed for this type of flow according to the state of the flow and the protocol, in order to fully characterize the flow. If not, the conversational flow has now been fully characterized and a process 334 finalizes the classification of the conversational flow for the flow.

[00114] In the particular embodiment, the state processor 328 starts the state processing by using the last protocol recognized by the parser as an offset into a jump table (jump vector). The jump table finds the state processor instructions to use for that protocol in the state patterns and processes database 326. Most instructions test something in the unified flow key buffer, or the flow-entry in the database of known flows 324, if the entry exists. The state processor may have to test bits, do comparisons, add, or subtract to perform the test. For example, a common operation carried out by the state processor is searching for one or more patterns in the payload part of the UFKB.

[00115] Thus, in 332 in the classification, the analyzer decides whether the flow is at an end state. If not at an end state, the flow-entry is updated (or created if a new flow) for this flow-entry in process 322.

[00116] Furthermore, if the flow is known and if in 332 it is determined that there are further states to be processed using later packets, the flow-entry is updated in process 322.

[00117] The flow-entry also is updated after classification finalization so that any further packets belonging to this flow will be readily identified from their signature as belonging to this fully analyzed conversational flow.

[00118] After updating, database 324 therefore includes the set of all the conversational flows that have occurred.

[00119] Thus, the embodiment of present invention shown in FIG. 3 automatically maintains flow-entries, which in one aspect includes storing states. The monitor of FIG. 3 also generates characteristic parts of packets—the signatures—that can be used to recognize flows. The flow-entries may be identified and accessed by their signatures. Once a packet is identified to be from a known flow, the state of the flow is known and this knowledge enables state transition analysis to be performed in real time for each different protocol and application. In a complex analysis, state transitions are traversed as more and more packets are examined. Future packets that are part of the same conversational flow have their state analysis continued from a previously achieved state. When enough packets related to an application of interest have been processed, a final recognition state is ultimately reached, *i.e.,* a set of states has been traversed by state analysis to completely characterize the conversational flow. The signature for that final state enables each new incoming packet of the same conversational flow to be individually recognized in real time.

[00120] In this manner, one of the great advantages of the present invention is realized. Once a particular set of state transitions has been traversed for the first time and ends in a final state, a short-cut recognition pattern—a signature—can be generated that will key on every new incoming packet that relates to the conversational flow. Checking a signature involves a simple operation, allowing high packet rates to be successfully monitored on the network.

[00121] In improved embodiments, several state analyzers are run in parallel so that a large number of protocols and applications may be checked for. Every known protocol and application will have at least one unique set of state transitions, and can therefore be uniquely identified by watching such transitions.

[00122] When each new conversational flow starts, signatures that recognize the flow are automatically generated on-the-fly, and as further packets in the conversational flow are encountered, signatures are updated and the states of the set of state transitions for any potential application are further traversed according to the state transition rules for the flow. The new states for the flow—those associated with a set of state transitions for one or more potential applications—are added to the records of previously encountered states for easy recognition and retrieval when a new packet in the flow is encountered.

Detailed operation

[00123] FIG. 4 diagrams an initialization system 400 that includes the compilation process. That is, part of the initialization generates the pattern structures and extraction operations database 308 and the state instruction database 328. Such initialization can occur off-line or from a central location.

[00124] The different protocols that can exist in different layers may be thought of as nodes of one or more trees of linked nodes. The packet type is the root of a tree (called level 0). Each protocol is either a parent node or a terminal node. A parent node links a protocol to other protocols (child protocols) that can be at higher layer levels. Thus a protocol may have zero or more children. Ethernet packets, for example, have several variants, each having a basic format that remains substantially the same. An Ethernet packet (the root or level 0 node) may be an Ethertype packet—also called an Ethernet Type/Version 2 and a DIX (DIGITAL-Intel-Xerox packet)—or an IEEE 803.2 packet. Continuing with the IEEE 802.3 packet, one of the children nodes may be the IP protocol, and one of the children of the IP protocol may be the TCP protocol.

[00125] FIG. 16 shows the header 1600 (base level 1) of a complete Ethernet frame (*i.e.,* packet) of information and includes information on the destination media access control address (Dst MAC 1602) and the source media access control address (Src MAC 1604). Also

shown in FIG. 16 is some (but not all) of the information specified in the PDL files for extraction the signature.

[00126] FIG. 17A now shows the header information for the next level (level-2) for an Ethertype packet 1700. For an Ethertype packet 1700, the relevant information from the packet that indicates the next layer level is a two-byte type field 1702 containing the child recognition pattern for the next level. The remaining information 1704 is shown hatched because it not relevant for this level. The list 1712 shows the possible children for an Ethertype packet as indicated by what child recognition pattern is found offset 12. FIG. 17B shows the structure of the header of one of the possible next levels, that of the IP protocol. The possible children of the IP protocol are shown in table 1752.

[00127] The pattern, parse, and extraction database (pattern recognition database, or PRD) 308 generated by compilation process 310, in one embodiment, is in the form of a three dimensional structure that provides for rapidly searching packet headers for the next protocol. FIG. 18A shows such a 3-D representation 1800 (which may be considered as an indexed set of 2-D representations). A compressed form of the 3-D structure is preferred.

[00128] An alternate embodiment of the data structure used in database 308 is illustrated in FIG. 18B. Thus, like the 3-D structure of FIG. 18A, the data structure permits rapid searches to be performed by the pattern recognition process 304 by indexing locations in a memory rather than performing address link computations. In this alternate embodiment, the PRD 308 includes two parts, a single protocol table 1850 (PT) which has an entry for each protocol known for the monitor, and a series of Look Up Tables 1870 (LUT's) that are used to identify known protocols and their children. The protocol table includes the parameters needed by the pattern analysis and recognition process 304 (implemented by PRE 1006) to evaluate the header information in the packet that is associated with that protocol, and parameters needed by extraction process 306 (implemented by slicer 1007) to process the packet header. When there are children, the PT describes which bytes in the header to evaluate to determine the child protocol. In particular, each PT entry contains the header length, an offset to the child, a slicer command, and some flags.

[00129] The pattern matching is carried out by finding particular "child recognition codes" in the header fields, and using these codes to index one or more of the LUT's. Each LUT entry has a node code that can have one of four values, indicating the protocol that has been recognized, a code to indicate that the protocol has been partially recognized (more LUT lookups are needed), a code to indicate that this is a terminal node, and a null node to indicate a null entry. The next LUT to lookup is also returned from a LUT lookup.

[00130] Compilation process is described in FIG. 4. The source-code information in the form of protocol description files is shown as 402. In the particular embodiment, the high level decoding descriptions includes a set of protocol description files 336, one for each protocol, and a set of packet layer selections 338, which describes the particular layering (sets of trees of protocols) that the monitor is to be able to handle.

[00131] A compiler 403 compiles the descriptions. The set of packet parse-and-extract operations 406 is generated (404), and a set of packet state instructions and operations 407 is generated (405) in the form of instructions for the state processor that implements state processing process 328. Data files for each type of application and protocol to be recognized by the analyzer are downloaded from the pattern, parse, and extraction database 406 into the memory systems of the parser and extraction engines. (See the parsing process 500 description and FIG. 5; the extraction process 600 description and FIG. 6; and the parsing subsystem hardware description and FIG. 10). Data files for each type of application and protocol to be recognized by the analyzer are also downloaded from the state-processor instruction database 407 into the state processor. (see the state processor 1108 description and FIG. 11.).

[00132] Note that generating the packet parse and extraction operations builds and links the three dimensional structure (one embodiment) or the or all the lookup tables for the PRD.

[00133] Because of the large number of possible protocol trees and subtrees, the compiler process 400 includes optimization that compares the trees and subtrees to see which children share common parents. When implemented in the form of the LUT's, this process can generate a single LUT from a plurality of LUT's. The optimization process further includes a compaction process that reduces the space needed to store the data of the PRD.

[00134] As an example of compaction, consider the 3-D structure of FIG. 18A that can be thought of as a set of 2-D structures each representing a protocol. To enable saving space by using only one array per protocol which may have several parents, in one embodiment, the pattern analysis subprocess keeps a "current header" pointer. Each location (offset) index for each protocol 2-D array in the 3-D structure is a relative location starting with the start of header for the particular protocol. Furthermore, each of the two-dimensional arrays is sparse. The next step of the optimization, is checking all the 2-D arrays against all the other 2-D arrays to find out which ones can share memory. Many of these 2-D arrays are often sparsely populated in that they each have only a small number of valid entries. So, a process of "folding" is next used to combine two or more 2-D arrays together into one physical 2-D array without losing the identity of any of the original 2-D arrays (i.e., all the 2-D arrays continue to exist logically). Folding can occur between any 2-D arrays irrespective of their location in the tree as long as certain conditions are met. Multiple arrays may be combined into a single array as long as the individual entries do not conflict with each other. A fold number is then used to associate each element with its original array. A similar folding process is used for the set of LUTs 1850 in the alternate embodiment of FIG. 18B.

[00135] In 410, the analyzer has been initialized and is ready to perform recognition.

[00136] FIG. 5 shows a flowchart of how actual parser subsystem 301 functions. Starting at 501, the packet 302 is input to the packet buffer in step 502. Step 503 loads the next (initially the first) packet component from the packet 302. The packet components are extracted from each packet 302 one element at a time. A check is made (504) to determine if the load-packet-component operation 503 succeeded, indicating that there was more in the packet to process. If not, indicating all components have been loaded, the parser subsystem 301 builds the packet signature (512)—the next stage (FIG 6).

[00137] If a component is successfully loaded in 503, the node and processes are fetched (505) from the pattern, parse and extraction database 308 to provide a set of patterns and processes for that node to apply to the loaded packet component. The parser subsystem 301 checks (506) to determine if the fetch pattern node operation 505 completed successfully, indicating there was a pattern node that loaded in 505. If not, step 511 moves to the next packet component. If yes, then the node and pattern matching process are applied in 507 to the

component extracted in 503. A pattern match obtained in 507 (as indicated by test 508) means the parser subsystem 301 has found a node in the parsing elements; the parser subsystem 301 proceeds to step 509 to extract the elements.

[00138] If applying the node process to the component does not produce a match (test 508), the parser subsystem 301 moves (510) to the next pattern node from the pattern database 308 and to step 505 to fetch the next node and process. Thus, there is an "applying patterns" loop between 508 and 505. Once the parser subsystem 301 completes all the patterns and has either matched or not, the parser subsystem 301 moves to the next packet component (511).

[00139] Once all the packet components have been the loaded and processed from the input packet 302, then the load packet will fail (indicated by test 504), and the parser subsystem 301 moves to build a packet signature which is described in FIG. 6

[00140] FIG. 6 is a flow chart for extracting the information from which to build the packet signature. The flow starts at 601, which is the exit point 513 of FIG. 5. At this point parser subsystem 301 has a completed packet component and a pattern node available in a buffer (602). Step 603 loads the packet component available from the pattern analysis process of FIG. 5. If the load completed (test 604), indicating that there was indeed another packet component, the parser subsystem 301 fetches in 605 the extraction and process elements received from the pattern node component in 602. If the fetch was successful (test 606), indicating that there are extraction elements to apply, the parser subsystem 301 in step 607 applies that extraction process to the packet component based on an extraction instruction received from that pattern node. This removes and saves an element from the packet component.

[00141] In step 608, the parser subsystem 301 checks if there is more to extract from this component, and if not, the parser subsystem 301 moves back to 603 to load the next packet component at hand and repeats the process. If the answer is yes, then the parser subsystem 301 moves to the next packet component ratchet. That new packet component is then loaded in step 603. As the parser subsystem 301 moved through the loop between 608 and 603, extra extraction processes are applied either to the same packet component if there is more to extract, or to a different packet component if there is no more to extract.

[00142] The extraction process thus builds the signature, extracting more and more components according to the information in the patterns and extraction database 308 for the particular packet. Once loading the next packet component operation 603 fails (test 604), all the components have been extracted. The built signature is loaded into the signature buffer (610) and the parser subsystem 301 proceeds to FIG. 7 to complete the signature generation process.

[00143] Referring now to FIG. 7, the process continues at 701. The signature buffer and the pattern node elements are available (702). The parser subsystem 301 loads the next pattern node element. If the load was successful (test 704) indicating there are more nodes, the parser subsystem 301 in 705 hashes the signature buffer element based on the hash elements that are found in the pattern node that is in the element database. In 706 the resulting signature and the hash are packed. In 707 the parser subsystem 301 moves on to the next packet component which is loaded in 703.

[00144] The 703 to 707 loop continues until there are no more patterns of elements left (test 704). Once all the patterns of elements have been hashed, processes 304, 306 and 312 of parser subsystem 301 are complete. Parser subsystem 301 has generated the signature used by the analyzer subsystem 303.

[00145] A parser record is loaded into the analyzer, in particular, into the UFKB in the form of a UFKB record which is similar to a parser record, but with one or more different fields.

[00146] FIG. 8 is a flow diagram describing the operation of the lookup/update engine (LUE) that implements lookup operation 314. The process starts at 801 from FIG. 7 with the parser record that includes a signature, the hash and at least parts of the payload. In 802 those elements are shown in the form of a UFKB-entry in the buffer. The LUE, the lookup engine 314 computes a "record bin number" from the hash for a flow-entry. A bin herein may have one or more "buckets" each containing a flow-entry. The preferred embodiment has four buckets per bin.

[00147] Since preferred hardware embodiment includes the cache, all data accesses to records in the flowchart of FIG. 8 are stated as being to or from the cache.

[00148] Thus, in 804, the system looks up the cache for a bucket from that bin using the hash. If the cache successfully returns with a bucket from the bin number, indicating there are more buckets in the bin, the lookup/update engine compares (807) the current signature (the UFKB-entry's signature) from that in the bucket (i.e., the flow-entry signature). If the signatures match (test 808), that record (in the cache) is marked in step 810 as "in process" and a timestamp added. Step 811 indicates to the UFKB that the UFKB-entry in 802 has a status of "found." The "found" indication allows the state processing 328 to begin processing this UFKB element. The preferred hardware embodiment includes one or more state processors, and these can operate in parallel with the lookup/update engine.

[00149] In the preferred embodiment, a set of statistical operations is performed by a calculator for every packet analyzed. The statistical operations may include one or more of counting the packets associated with the flow; determining statistics related to the size of packets of the flow; compiling statistics on differences between packets in each direction, for example using timestamps; and determining statistical relationships of timestamps of packets in the same direction. The statistical measures are kept in the flow-entries. Other statistical measures also may be compiled. These statistics may be used singly or in combination by a statistical processor component to analyze many different aspects of the flow. This may include determining network usage metrics from the statistical measures, for example to ascertain the network's ability to transfer information for this application. Such analysis provides for measuring the quality of service of a conversation, measuring how well an application is performing in the network, measuring network resources consumed by an application, and so forth.

[00150] To provide for such analyses, the lookup/update engine updates one or more counters that are part of the flow-entry (in the cache) in step 812. The process exits at 813. In our embodiment, the counters include the total packets of the flow, the time, and a differential time from the last timestamp to the present timestamp.

[00151] It may be that the bucket of the bin did not lead to a signature match (test 808). In such a case, the analyzer in 809 moves to the next bucket for this bin. Step 804 again looks up the cache for another bucket from that bin. The lookup/update engine thus continues lookup up buckets of the bin until there is either a match in 808 or operation 804 is not

successful (test 805), indicating that there are no more buckets in the bin and no match was found.

[00152] If no match was found, the packet belongs to a new (not previously encountered) flow. In 806 the system indicates that the record in the unified flow key buffer for this packet is new, and in 812, any statistical updating operations are performed for this packet by updating the flow-entry in the cache. The update operation exits at 813. A flow insertion/deletion engine (FIDE) creates a new record for this flow (again via the cache).

[00153] Thus, the update/lookup engine ends with a UFKB-entry for the packet with a "new" status or a "found" status.

[00154] Note that the above system uses a hash to which more than one flow-entry can match. A longer hash may be used that corresponds to a single flow-entry. In such an embodiment, the flow chart of FIG. 8 is simplified as would be clear to those in the art.

The hardware system

[00155] Each of the individual hardware elements through which the data flows in the system are now described with reference to FIGS. 10 and 11. Note that while we are describing a particular hardware implementation of the invention embodiment of FIG. 3, it would be clear to one skilled in the art that the flow of FIG. 3 may alternatively be implemented in software running on one or more general-purpose processors, or only partly implemented in hardware. An implementation of the invention that can operate in software is shown in FIG. 14. The hardware embodiment (FIGS. 10 and 11) can operate at over a million packets per second, while the software system of FIG. 14 may be suitable for slower networks. To one skilled in the art it would be clear that more and more of the system may be implemented in software as processors become faster.

[00156] FIG. 10 is a description of the parsing subsystem (301, shown here as subsystem 1000) as implemented in hardware. Memory 1001 is the pattern recognition database memory, in which the patterns that are going to be analyzed are stored. Memory 1002 is the extraction-operation database memory, in which the extraction instructions are stored. Both 1001 and 1002 correspond to internal data structure 308 of FIG. 3. Typically, the system is initialized from a microprocessor (not shown) at which time these memories are loaded through a host interface multiplexor and control register 1005 via the internal buses 1003 and 1004. Note that the contents of 1001 and 1002 are preferably obtained by compiling process 310 of FIG. 3.

[00157] A packet enters the parsing system via 1012 into a parser input buffer memory 1008 using control signals 1021 and 1023, which control an input buffer interface controller 1022. The buffer 1008 and interface control 1022 connect to a packet acquisition device (not shown). The buffer acquisition device generates a packet start signal 1021 and the interface control 1022 generates a next packet (i.e., ready to receive data) signal 1023 to control the data flow into parser input buffer memory 1008. Once a packet starts loading into the buffer memory 1008, pattern recognition engine (PRE) 1006 carries out the operations on the input buffer memory described in block 304 of FIG. 3. That is, protocol types and associated headers for each protocol layer that exist in the packet are determined.

[00158] The PRE searches database 1001 and the packet in buffer 1008 in order to recognize the protocols the packet contains. In one implementation, the database 1001 includes a series of linked lookup tables. Each lookup table uses eight bits of addressing. The first lookup table is always at address zero. The Pattern Recognition Engine uses a base packet offset from a control register to start the comparison. It loads this value into a current offset pointer (COP). It then reads the byte at base packet offset from the parser input buffer and uses it as an address into the first lookup table.

[00159] Each lookup table returns a word that links to another lookup table or it returns a terminal flag. If the lookup produces a recognition event the database also returns a command for the slicer. Finally it returns the value to add to the COP.

[00160] The PRE 1006 includes of a comparison engine. The comparison engine has a first stage that checks the protocol type field to determine if it is an 802.3 packet and the field should be treated as a length. If it is not a length, the protocol is checked in a second stage. The first stage is the only protocol level that is not programmable. The second stage has two full sixteen bit content addressable memories (CAMs) defined for future protocol additions.

[00161] Thus, whenever the PRE recognizes a pattern, it also generates a command for the extraction engine (also called a "slicer") 1007. The recognized patterns and the commands are sent to the extraction engine 1007 that extracts information from the packet to build the parser record. Thus, the operations of the extraction engine are those carried out in blocks 306 and 312 of FIG. 3. The commands are sent from PRE 1006 to slicer 1007 in the form of extraction instruction pointers which tell the extraction engine 1007 where to a find the instructions in the extraction operations database memory (i.e., slicer instruction database) 1002.

[00162] Thus, when the PRE 1006 recognizes a protocol it outputs both the protocol identifier and a process code to the extractor. The protocol identifier is added to the flow signature and the process code is used to fetch the first instruction from the instruction database 1002. Instructions include an operation code and usually source and destination offsets as well as a length. The offsets and length are in bytes. A typical operation is the MOVE instruction. This instruction tells the slicer 1007 to copy n bytes of data unmodified from the input buffer 1008 to the output buffer 1010. The extractor contains a byte-wise barrel shifter so that the bytes moved can be packed into the flow signature. The extractor contains another instruction called HASH. This instruction tells the extractor to copy from the input buffer 1008 to the HASH generator.

[00163] Thus these instructions are for extracting selected element(s) of the packet in the input buffer memory and transferring the data to a parser output buffer memory 1010. Some instructions also generate a hash.

[00164] The extraction engine 1007 and the PRE operate as a pipeline. That is, extraction engine 1007 performs extraction operations on data in input buffer 1008 already processed by PRE 1006 while more (i.e., later arriving) packet information is being simultaneously parsed by PRE 1006. This provides high processing speed sufficient to accommodate the high arrival rate speed of packets.

[00165] Once all the selected parts of the packet used to form the signature are extracted, the hash is loaded into parser output buffer memory 1010. Any additional payload from the packet that is required for further analysis is also included. The parser output memory 1010 is interfaced with the analyzer subsystem by analyzer interface control 1011. Once all the information of a packet is in the parser output buffer memory 1010, a data ready signal 1025 is asserted by analyzer interface control. The data from the parser subsystem 1000 is moved to the analyzer subsystem via 1013 when an analyzer ready signal 1027 is asserted.

[00166] FIG. 11 shows the hardware components and dataflow for the analyzer subsystem that performs the functions of the analyzer subsystem 303 of FIG. 3. The analyzer is initialized prior to operation, and initialization includes loading the state processing information generated by the compilation process 310 into a database memory for the state processing, called state processor instruction database (SPID) memory 1109.

[00167] The analyzer subsystem 1100 includes a host bus interface 1122 using an analyzer host interface controller 1118, which in turn has access to a cache system 1115. The cache system has bi-directional access to and from the state processor of the system 1108. State processor 1108 is responsible for initializing the state processor instruction database memory 1109 from information given over the host bus interface 1122.

[00168] With the SPID 1109 loaded, the analyzer subsystem 1100 receives parser records comprising packet signatures and payloads that come from the parser into the unified flow key buffer (UFKB) 1103. UFKB is comprised of memory set up to maintain UFKB records. A UFKB record is essentially a parser record; the UFKB holds records of packets that are to be processed or that are in process. Furthermore, the UFKB provides for one or more fields to act as modifiable status flags to allow different processes to run concurrently.

[00169] Three processing engines run concurrently and access records in the UFKB 1103: the lookup/update engine (LUE) 1107, the state processor (SP) 1108, and the flow insertion and deletion engine (FIDE) 1110. Each of these is implemented by one or more finite state machines (FSM's). There is bi-directional access between each of the finite state machines and the unified flow key buffer 1103. The UFKB record includes a field that stores the packet sequence number, and another that is filled with state information in the form of a program counter for the state processor 1108 that implements state processing 328. The status flags of the UFKB for any entry includes that the LUE is done and that the LUE is transferring processing of the entry to the state processor. The LUE done indicator is also used to indicate what the next entry is for the LUE. There also is provided a flag to indicate that the state processor is done with the current flow and to indicate what the next entry is for the state processor. There also is provided a flag to indicate the state processor is transferring processing of the UFKB-entry to the flow insertion and deletion engine.

[00170] A new UFKB record is first processed by the LUE 1107. A record that has been processed by the LUE 1107 may be processed by the state processor 1108, and a UFKB record data may be processed by the flow insertion/deletion engine 1110 after being processed by the state processor 1108 or only by the LUE. Whether or not a particular engine has been applied to any unified flow key buffer entry is determined by status fields set by the engines upon completion. In one embodiment, a status flag in the UFKB-entry indicates whether an entry is new or found. In other embodiments, the LUE issues a flag to pass the entry to the state processor for processing, and the required operations for a new record are included in the SP instructions.

[00171] Note that each UFKB-entry may not need to be processed by all three engines. Furthermore, some UFKB entries may need to be processed more than once by a particular engine.

[00172] Each of these three engines also has bi-directional access to a cache subsystem 1115 that includes a caching engine. Cache 1115 is designed to have information flowing in and out of it from five different points within the system: the three engines, external memory via a unified memory controller (UMC) 1119 and a memory interface 1123, and a microprocessor via analyzer host interface and control unit (ACIC) 1118 and host interface bus (HIB) 1122. The analyzer microprocessor (or dedicated logic processor) can thus directly insert or modify data in the cache.

[00173] The cache subsystem 1115 is an associative cache that includes a set of content addressable memory cells (CAMs) each including an address portion and a pointer portion pointing to the cache memory (e.g., RAM) containing the cached flow-entries. The CAMs are arranged as a stack ordered from a top CAM to a bottom CAM. The bottom CAM's pointer points to the least recently used (LRU) cache memory entry. Whenever there is a cache miss, the contents of cache memory pointed to by the bottom CAM are replaced by the flow-entry from the flow-entry database 324. This now becomes the most recently used entry, so the contents of the bottom CAM are moved to the top CAM and all CAM contents are shifted down. Thus, the cache is an associative cache with a true LRU replacement policy.

[00174] The LUE 1107 first processes a UFKB-entry, and basically performs the operation of blocks 314 and 316 in FIG. 3. A signal is provided to the LUE to indicate that a "new" UFKB-entry is available. The LUE uses the hash in the UFKB-entry to read a matching bin of up to four buckets from the cache. The cache system attempts to obtain the matching bin. If a matching bin is not in the cache, the cache 1115 makes the request to the UMC 1119 to bring in a matching bin from the external memory.

[00175] When a flow-entry is found using the hash, the LUE 1107 looks at each bucket and compares it using the signature to the signature of the UFKB-entry until there is a match or there are no more buckets.

[00176] If there is no match, or if the cache failed to provide a bin of flow-entries from the cache, a time stamp in set in the flow key of the UFKB record, a protocol identification and state determination is made using a table that was loaded by compilation process 310 during initialization, the status for the record is set to indicate the LUE has processed the record, and an indication is made that the UFKB-entry is ready to start state processing. The identification and state determination generates a protocol identifier which in the preferred embodiment is a "jump vector" for the state processor which is kept by the UFKB for this UFKB-entry and used by the state processor to start state processing for the particular protocol. For example, the jump vector jumps to the subroutine for processing the state.

[00177] If there was a match, indicating that the packet of the UFKB-entry is for a previously encountered flow, then a calculator component enters one or more statistical measures stored in the flow-entry, including the timestamp. In addition, a time difference from the last stored timestamp may be stored, and a packet count may be updated. The state of the flow is obtained from the flow-entry is examined by looking at the protocol identifier stored in the flow-entry of database 324. If that value indicates that no more classification is required, then the status for the record is set to indicate the LUE has processed the record. In the preferred embodiment, the protocol identifier is a jump vector for the state processor to a subroutine to state processing the protocol, and no more classification is indicated in the preferred

embodiment by the jump vector being zero. If the protocol identifier indicates more processing, then an indication is made that the UFKB-entry is ready to start state processing and the status for the record is set to indicate the LUE has processed the record.

[00178] The state processor 1108 processes information in the cache system according to a UFKB-entry after the LUE has completed. State processor 1108 includes a state processor program counter SPPC that generates the address in the state processor instruction database 1109 loaded by compiler process 310 during initialization. It contains an Instruction Pointer (SPIP) which generates the SPID address. The instruction pointer can be incremented or loaded from a Jump Vector Multiplexor which facilitates conditional branching. The SPIP can be loaded from one of three sources: (1) A protocol identifier from the UFKB, (2) an immediate jump vector form the currently decoded instruction, or (3) a value provided by the arithmetic logic unit (SPALU) included in the state processor.

[00179] Thus, after a Flow Key is placed in the UFKB by the LUE with a known protocol identifier, the Program Counter is initialized with the last protocol recognized by the Parser. This first instruction is a jump to the subroutine which analyzes the protocol that was decoded.

[00180] The State Processor ALU (SPALU) contains all the Arithmetic, Logical and String Compare functions necessary to implement the State Processor instructions. The main blocks of the SPALU are: The A and B Registers, the Instruction Decode & State Machines, the String Reference Memory the Search Engine, an Output Data Register and an Output Control Register

[00181] The Search Engine in turn contains the Target Search Register set, the Reference Search Register set, and a Compare block which compares two operands by exclusive-or-ing them together.

[00182] Thus, after the UFKB sets the program counter, a sequence of one or more state operations are be executed in state processor 1108 to further analyze the packet that is in the flow key buffer entry for this particular packet.

[00183] FIG. 13 describes the operation of the state processor 1108. The state processor is entered at 1301 with a unified flow key buffer entry to be processed. The UFKB-entry is new or corresponding to a found flow-entry. This UFKB-entry is retrieved from unified flow key buffer 1103 in 1301. In 1303, the protocol identifier for the UFKB-entry is used to set the state processor's instruction counter. The state processor 1108 starts the process by using the last protocol recognized by the parser subsystem 301 as an offset into a jump table. The jump table takes us to the instructions to use for that protocol. Most instructions test something in the unified flow key buffer or the flow-entry if it exists. The state processor 1108 may have to test bits, do comparisons, add or subtract to perform the test.

[00184] The first state processor instruction is fetched in 1304 from the state processor instruction database memory 1109. The state processor performs the one or more fetched operations (1304). In our implementation, each single state processor instruction is very primitive (e.g., a move, a compare, etc.), so that many such instructions need to be performed on each unified flow key buffer entry. One aspect of the state processor is its ability to search for one or more (up to four) reference strings in the payload part of the UFKB entry. This is implemented by a search engine component of the state processor responsive to special searching instructions.

[00185] In 1307, a check is made to determine if there are any more instructions to be performed for the packet. If yes, then in 1308 the system sets the state processor instruction pointer (SPIP) to obtain the next instruction. The SPIP may be set by an immediate jump vector in the currently decoded instruction, or by a value provided by the SPALU during processing.

[00186] The next instruction to be performed is now fetched (1304) for execution. This state processing loop between 1304 and 1307 continues until there are no more instructions to be performed.

[00187] At this stage, a check is made in 1309 if the processing on this particular packet has resulted in a final state. That is, is the analyzer is done processing not only for this particular packet, but for the whole flow to which the packet belongs, and the flow is fully determined. If indeed there are no more states to process for this flow, then in 1311 the processor finalizes the processing. Some final states may need to put a state in place that tells the system to remove a flow—for example, if a connection disappears from a lower level connection

identifier. In that case, in 1311, a flow removal state is set and saved in the flow-entry. The flow removal state may be a NOP (no-op) instruction which means there are no removal instructions.

[00188] Once the appropriate flow removal instruction as specified for this flow (a NOP or otherwise) is set and saved, the process is exited at 1313. The state processor 1108 can now obtain another unified flow key buffer entry to process.

[00189] If at 1309 it is determined that processing for this flow is not completed, then in 1310 the system saves the state processor instruction pointer in the current flow-entry in the current flow-entry. That will be the next operation that will be performed the next time the LRE 1107 finds packet in the UFKB that matches this flow. The processor now exits processing this particular unified flow key buffer entry at 1313.

[00190] Note that state processing updates information in the unified flow key buffer 1103 and the flow-entry in the cache. Once the state processor is done, a flag is set in the UFKB for the entry that the state processor is done. Furthermore, If the flow needs to be inserted or deleted from the database of flows, control is then passed on to the flow insertion/deletion engine 1110 for that flow signature and packet entry. This is done by the state processor setting another flag in the UFKB for this UFKB-entry indicating that the state processor is passing processing of this entry to the flow insertion and deletion engine.

[00191] The flow insertion and deletion engine 1110 is responsible for maintaining the flow-entry database. In particular, for creating new flows in the flow database, and deleting flows from the database so that they can be reused.

[00192] The process of flow insertion is now described with the aid of FIG. 12. Flows are grouped into bins of buckets by the hash value. The engine processes a UFKB-entry that may be new or that the state processor otherwise has indicated needs to be created. FIG. 12 shows the case of a new entry being created. A conversation record bin (preferably containing 4 buckets for four records) is obtained in 1203. This is a bin that matches the hash of the UFKB, so this bin may already have been sought for the UFKB-entry by the LUE. In 1204 the FIDE 1110 requests that the record bin/bucket be maintained in the cache system 1115. If in 1205 the cache system 1115 indicates that the bin/bucket is empty, step 1207 inserts the flow signature (with the hash) into the bucket and the bucket is marked "used" in the cache engine of cache 1115 using a timestamp that is maintained throughout the process. In 1209, the FIDE 1110 compares the bin and bucket record flow signature to the packet to verify that all the elements are in place to complete the record. In 1211 the system marks the record bin and bucket as "in process" and as "new" in the cache system (and hence in the external memory). In 1212, the initial statistical measures for the flow-record are set in the cache system. This in the preferred embodiment clears the set of counters used to maintain statistics, and may perform other procedures for statistical operations requires by the analyzer for the first packet seen for a particular flow.

[00193] Back in step 1205, if the bucket is not empty, the FIDE 1110 requests the next bucket for this particular bin in the cache system. If this succeeds, the processes of 1207, 1209, 1211 and 1212 are repeated for this next bucket. If at 1208, there is no valid bucket, the unified flow key buffer entry for the packet is set as "drop," indicating that the system cannot process the particular packet because there are no buckets left in the system. The process exits at 1213. The FIDE 1110 indicates to the UFKB that the flow insertion and deletion operations are completed for this UFKB-entry. This also lets the UFKB provide the FIDE with the next UFKB record.

[00194] Once a set of operations is performed on a unified flow key buffer entry by all of the engines required to access and manage a particular packet and its flow signature, the unified flow key buffer entry is marked as "completed." That element will then be used by the parser interface for the next packet and flow signature coming in from the parsing and extracting system.

[00195] All flow-entries are maintained in the external memory and some are maintained in the cache 1115. The cache system 1115 is intelligent enough to access the flow database and to understand the data structures that exists on the other side of memory interface 1123. The lookup/update engine 1107 is able to request that the cache system pull a particular flow or "buckets" of flows from the unified memory controller 1119 into the cache system for further processing. The state processor 1108 can operate on information found in the cache system once it is looked up by means of the lookup/update engine request, and the flow insertion/deletion engine 1110 can create new entries in the cache system if required based on information in the unified flow key buffer 1103. The cache retrieves information as required from the memory through the memory interface 1123 and the unified memory controller 1119, and updates information as required in the memory through the memory controller 1119.

[00196] There are several interfaces to components of the system external to the module of FIG. 11 for the particular hardware implementation. These include host bus interface 1122,which is designed as a generic interface that can operate with any kind of external processing system such as a microprocessor or a multiplexor (MUX) system. Consequently, one can connect the overall traffic classification system of FIGS. 11 and 12 into some other processing system to manage the classification system and to extract data gathered by the system.

[00197] The memory interface 1123 is designed to interface to any of a variety of memory systems that one may want to use to store the flow-entries. One can use different types of memory systems like regular dynamic random access memory (DRAM), synchronous DRAM, synchronous graphic memory (SGRAM), static random access memory (SRAM), and so forth.

[00198] FIG. 10 also includes some "generic" interfaces. There is a packet input interface 1012—a general interface that works in tandem with the signals of the input buffer interface control 1022. These are designed so that they can be used with any kind of generic systems that can then feed packet information into the parser. Another generic interface is the interface of pipes 1031 and 1033 respectively out of and into host interface multiplexor and control registers 1005. This enables the parsing system to be managed by an external system, for example a microprocessor or another kind of external logic, and enables the external system to program and otherwise control the parser.

[00199] The preferred embodiment of this aspect of the invention is described in a hardware description language (HDL) such as VHDL or Verilog. It is designed and created in an HDL so that it may be used as a single chip system or, for instance, integrated into another general-purpose system that is being designed for purposes related to creating and analyzing traffic within a network. Verilog or other HDL implementation is only one method of describing the hardware.

[00200] In accordance with one hardware implementation, the elements shown in FIGS. 10 and 11 are implemented in a set of six field programmable logic arrays (FPGA's). The boundaries of these FPGA's are as follows. The parsing subsystem of FIG. 10 is implemented as two FPGAS; one FPGA, and includes blocks 1006, 1008 and 1012, parts of 1005, and memory 1001. The second FPGA includes 1002, 1007, 1013, 1011 parts of 1005. Referring to FIG. 11, the unified look-up buffer 1103 is implemented as a single FPGA. State processor 1108 and part of state processor instruction database memory 1109 is another FPGA. Portions of the state processor instruction database memory 1109 are maintained in external SRAM's. The lookup/update engine 1107 and the flow insertion/deletion engine 1110 are in another FPGA. The sixth FPGA includes the cache system 1115, the unified memory control 1119, and the analyzer host interface and control 1118.

[00201] Note that one can implement the system as one or more VSLI devices, rather than as a set of application specific integrated circuits (ASIC's) such as FPGA's. It is anticipated that in the future device densities will continue to increase, so that the complete system may eventually form a sub-unit (a "core") of a larger single chip unit.

Operation of the Invention

[00202] Fig. 15 shows how an embodiment of the network monitor 300 might be used to analyze traffic in a network 102. Packet acquisition device 1502 acquires all the packets from a connection point 121 on network 102 so that all packets passing point 121 in either direction are supplied to monitor 300. Monitor 300 comprises the parser sub-system 301, which determines flow signatures, and analyzer sub-system 303 that analyzes the flow signature of each packet. A memory 324 is used to store the database of flows that are determined and updated by monitor 300. A host computer 1504, which might be any processor, for example, a general-purpose computer, is used to analyze the flows in memory 324. As is conventional, host computer 1504 includes a memory, say RAM, shown as host memory 1506. In addition, the host might contain a disk. In one application, the system can operate as an RMON probe, in which case the host computer is coupled to a network interface card 1510 that is connected to the network 102.

[00203] The preferred embodiment of the invention is supported by an optional Simple Network Management Protocol (SNMP) implementation. Fig. 15 describes how one would, for example, implement an RMON probe, where a network interface card is used to send RMON information to the network. Commercial SNMP implementations also are available, and using such an implementation can simplify the process of porting the preferred embodiment of the invention to any platform.

[00204] In addition, MIB Compilers are available. An MIB Compiler is a tool that greatly simplifies the creation and maintenance of proprietary MIB extensions.

Examples of Packet Elucidation

[00205] Monitor 300, and in particular, analyzer 303 is capable of carrying out state analysis for packet exchanges that are commonly referred to as "server announcement" type exchanges. Server announcement is a process used to ease communications between a server with multiple applications that can all be simultaneously accessed from multiple clients. Many applications use a server announcement process as a means of multiplexing a single port or socket into many applications and services. With this type of exchange, messages are sent on the network, in either a broadcast or multicast approach, to announce a server and application, and all stations in the network may receive and decode these messages. The messages enable the stations to derive the appropriate connection point for communicating that particular application with the particular server. Using the server announcement method, a particular application communicates using a service channel, in the form of a TCP or UDP socket or port as in the IP protocol suite, or using a SAP as in the Novell IPX protocol suite.

[00206] The analyzer 303 is also capable of carrying out "in-stream analysis" of packet exchanges. The "in-stream analysis" method is used either as a primary or secondary recognition process. As a primary process, in-stream analysis assists in extracting detailed information which will be used to further recognize both the specific application and application component. A good example of in-stream analysis is any Web-based application. For example, the commonly used PointCast Web information application can be recognized using this process; during the initial connection between a PointCast server and client, specific key tokens exist in the data exchange that will result in a signature being generated to recognize PointCast.

[00207] The in-stream analysis process may also be combined with the server announcement process. In many cases in-stream analysis will augment other recognition processes. An example of combining in-stream analysis with server announcement can be found in business applications such as SAP and BAAN.

[00208] "Session tracking" also is known as one of the primary processes for tracking applications in client/server packet exchanges. The process of tracking sessions requires an initial connection to a predefined socket or port number. This method of communication is used in a variety of transport layer protocols. It is most commonly seen in the TCP and UDP transport protocols of the IP protocol.

[00209] During the session tracking, a client makes a request to a server using a specific port or socket number. This initial request will cause the server to create a TCP or UDP port to exchange the remainder of the data between the client and the server. The server then replies to the request of the client using this newly created port. The original port used by the client to connect to the server will never be used again during this data exchange.

[00210] One example of session tracking is TFTP (Trivial File Transfer Protocol), a version of the TCP/IP FTP protocol that has no directory or password capability. During the client/server exchange process of TFTP, a specific port (port number 69) is always used to initiate the packet exchange. Thus, when the client begins the process of communicating, a request is made to UDP port 69. Once the server receives this request, a new port number is created on the server. The server then replies to the client using the new port. In this example, it is clear that in order to recognize TFTP; network monitor 300 analyzes the initial request from the client and generates a signature for it. Monitor 300 uses that signature to recognize the reply. Monitor 300 also analyzes the reply from the server with the key port information, and uses this to create a signature for monitoring the remaining packets of this data exchange.

[00211] Network monitor 300 can also understand the current state of particular connections in the network. Connection-oriented exchanges often benefit from state tracking to correctly identify the application. An example is the common TCP transport protocol that provides a reliable means of sending information between a client and a server. When a data exchange is initiated, a TCP request for synchronization message is sent. This message contains a specific sequence number that is used to track an acknowledgement from the server. Once the server has acknowledged the synchronization request, data may be exchanged between the client and the server. When communication is no longer required, the client sends a finish or complete message to the server, and the server acknowledges this finish request with a reply containing the sequence numbers from the request. The states of such a connection-oriented exchange relate to the various types of connection and maintenance messages.

Server Announcement Example

[00212] The individual methods of server announcement protocols vary. However, the basic underlying process remains similar. A typical server announcement message is sent to one or more clients in a network. This type of announcement message has specific content, which, in another aspect of the invention, is salvaged and maintained in the database of flow-entries in the system. Because the announcement is sent to one or more stations, the client involved in a future packet exchange with the server will make an assumption that the information announced is known, and an aspect of the inventive monitor is that it too can make the same assumption.

[00213] Sun-RPC is the implementation by Sun Microsystems, Inc. (Palo Alto, California) of the Remote Procedure Call (RPC), a programming interface that allows one program to use the services of another on a remote machine. A Sun-RPC example is now used to explain how monitor 300 can capture server announcements.

[00214] A remote program or client that wishes to use a server or procedure must establish a connection, for which the RPC protocol can be used.

[00215] Each server running the Sun-RPC protocol must maintain a process and database called the port Mapper. The port Mapper creates a direct association between a Sun-RPC program or application and a TCP or UDP socket or port (for TCP or UDP implementations). An application or program number is a 32-bit unique identifier assigned by ICANN (the Internet Corporation for Assigned Names and Numbers, www.icann.org), which manages the huge number of parameters associated with Internet protocols (port numbers, router protocols, multicast addresses, *etc.*) Each port Mapper on a Sun-RPC server can present the mappings between a unique program number and a specific transport socket through the use of specific request or a directed announcement. According to ICANN, port number 111 is associated with Sun RPC.

[00216] As an example, consider a client (*e.g.,* CLIENT 3 shown as 106 in FIG. 1) making a specific request to the server (*e.g.,* SERVER 2 of FIG. 1, shown as 110) on a predefined UDP or TCP socket. Once the port Mapper process on the sun RPC server receives the request, the specific mapping is returned in a directed reply to the client.

[00217] 1. A client (CLIENT 3, 106 in FIG. 1) sends a TCP packet to SERVER 2 (110 in FIG. 1) on port 111, with an RPC Bind Lookup Request (rpcBindLookup). TCP or UDP port 111 is always associated Sun RPC. This request specifies the program (as a program identifier), version, and might specify the protocol (UDP or TCP).

[00218] 2. The server SERVER 2 (110 in FIG. 1) extracts the program identifier and version identifier from the request. The server also uses the fact that this packet came in using the TCP transport and that no protocol was specified, and thus will use the TCP protocol for its reply.

[00219] 3. The server 110 sends a TCP packet to port number 111, with an RPC Bind Lookup Reply. The reply contains the specific port number (*e.g., port* number 'port') on which future transactions will be accepted for the specific RPC program identifier (*e.g.,* Program 'program') and the protocol (UDP or TCP) for use.

[00220] It is desired that from now on every time that port number 'port' is used, the packet is associated with the application program 'program' until the number 'port' no longer is to be associated with the program 'program'. Network monitor 300 by creating a flow-entry and a signature includes a mechanism for remembering the exchange so that future packets that use the port number 'port' will be associated by the network monitor with the application program 'program'.

[00221] In addition to the Sun RPC Bind Lookup request and reply, there are other ways that a particular program—say 'program'—might be associated with a particular port number, for example number 'port'. One is by a broadcast announcement of a particular association between an application service and a port number, called a Sun RPC portMapper Announcement. Another, is when some server—say the same SERVER 2—replies to some client—say CLIENT 1—requesting some portMapper assignment with a RPC portMapper Reply. Some other client—say CLIENT 2—might inadvertently see this request, and thus know that for this particular server, SERVER 2, port number 'port' is associated with the application service 'program'. It is desirable for the network monitor 300 to be able to associate any packets to SERVER 2 using port number 'port' with the application program 'program'.

[00222] FIG. 9 represents a dataflow 900 of some operations in the monitor 300 of FIG. 3 for Sun Remote Procedure Call. Suppose a client 106 (*e.g.*, CLIENT 3 in FIG. 1) is communicating via its interface to the network 118 to a server 110 (*e.g.*, SERVER 2 in FIG. 1) via the server's interface to the network 116. Further assume that Remote Procedure Call is used to communicate with the server 110. One path in the data flow 900 starts with a step 910 that a Remote Procedure Call bind lookup request is issued by client 106 and ends with the server state creation step 904. Such RPC bind lookup request includes values for the 'program,' 'version,' and 'protocol' to use, *e.g.*, TCP or UDP. The process for Sun RPC analysis in the network monitor 300 includes the following aspects. :

[00223] • Process 909: Extract the 'program,' 'version,' and 'protocol' (UDP or TCP). Extract the TCP or UDP port (process 909) which is 111 indicating Sun RPC.

[00224] • Process 908: Decode the Sun RPC packet. Check RPC type field for ID. If value is portMapper, save paired socket (*i.e.*, dest for destination address, src for source address). Decode ports and mapping, save ports with socket/addr key. There may be more than one pairing per mapper packet. Form a signature (e.g., a key). A flow-entry is created in database 324. The saving of the request is now complete.

[00225] At some later time, the server (process 907) issues a RPC bind lookup reply. The packet monitor 300 will extract a signature from the packet and recognize it from the

previously stored flow. The monitor will get the protocol port number (906) and lookup the request (905). A new signature (i.e., a key) will be created and the creation of the server state (904) will be stored as an entry identified by the new signature in the flow-entry database. That signature now may be used to identify packets associated with the server.

[00226] The server state creation step 904 can be reached not only from a Bind Lookup Request/Reply pair, but also from a RPC Reply portMapper packet shown as 901 or an RPC Announcement portMapper shown as 902. The Remote Procedure Call protocol can announce that it is able to provide a particular application service. Embodiments of the present invention preferably can analyze when an exchange occurs between a client and a server, and also can track those stations that have received the announcement of a service in the network.

[00227] The RPC Announcement portMapper announcement 902 is a broadcast. Such causes various clients to execute a similar set of operations, for example, saving the information obtained from the announcement. The RPC Reply portMapper step 901 could be in reply to a portMapper request, and is also broadcast. It includes all the service parameters.

[00228] Thus monitor 300 creates and saves all such states for later classification of flows that relate to the particular service 'program'.

[00229] FIG. 2 shows how the monitor 300 in the example of Sun RPC builds a signature and flow states. A plurality of packets 206-209 are exchanged, *e.g.*, in an exemplary Sun Microsystems Remote Procedure Call protocol. A method embodiment of the present invention might generate a pair of flow signatures, "signature-1" 210 and "signature-2" 212, from information found in the packets 206 and 207 which, in the example, correspond to a Sun RPC Bind Lookup request and reply, respectively.

[00230] Consider first the Sun RPC Bind Lookup request. Suppose packet 206 corresponds to such a request sent from CLIENT 3 to SERVER 2. This packet contains important information that is used in building a signature according to an aspect of the invention. A source and destination network address occupy the first two fields of each packet, and according to the patterns in pattern database 308, the flow signature (shown as KEY1 230 in FIG. 2) will also contain these two fields, so the parser subsystem 301 will include these two fields in signature KEY 1 (230). Note that in FIG. 2, if an address identifies the client 106 (shown also as 202), the label used in the drawing is "C_1". If such address identifies the server 110 (shown also as server 204), the label used in the drawing is "S_1". The first two fields 214 and 215 in packet 206 are "S_1" and C_1" because packet 206 is provided from the server 110 and is destined for the client 106. Suppose for this example, "S_1" is an address numerically less than address "C_1". A third field "p^1" 216 identifies the particular protocol being used, *e.g.,* TCP, UDP, etc.

[00231] In packet 206, a fourth field 217 and a fifth field 218 are used to communicate port numbers that are used. The conversation direction determines where the port number field is. The diagonal pattern in field 217 is used to identify a source-port pattern, and the hash pattern in field 218 is used to identify the destination-port pattern. The order indicates the client-server message direction. A sixth field denoted "i^1" 219 is an element that is being requested by the client from the server. A seventh field denoted "s_1a" 220 is the service requested by the client from server 110. The following eighth field "QA" 221 (for question mark) indicates that the client 106 wants to know what to use to access application "s_1a". A tenth field "QP" 223 is used to indicate that the client wants the server to indicate what protocol to use for the particular application.

[00232] Packet 206 initiates the sequence of packet exchanges, *e.g.,* a RPC Bind Lookup Request to SERVER 2. It follows a well-defined format, as do all the packets, and is transmitted to the server 110 on a well-known service connection identifier (port 111 indicating Sun RPC).

[00233] Packet 207 is the first sent in reply to the client 106 from the server. It is the RPC Bind Lookup Reply as a result of the request packet 206.

[00234] Packet 207 includes ten fields 224–233. The destination and source addresses are carried in fields 224 and 225, *e.g.,* indicated "C_1" and "S_1", respectively. Notice the order is now reversed, since the client-server message direction is from the server 110 to the client 106. The protocol "p^1" is used as indicated in field 226. The request "i^1" is in field 229.

Values have been filled in for the application port number, *e.g.*, in field 233 and protocol ""p^2"" in field 233.

[00235] The flow signature and flow states built up as a result of this exchange are now described. When the packet monitor 300 sees the request packet 206 from the client, a first flow signature 210 is built in the parser subsystem 301 according to the pattern and extraction operations database 308. This signature 210 includes a destination and a source address 240 and 241. One aspect of the invention is that the flow keys are built consistently in a particular order no matter what the direction of conversation. Several mechanisms may be used to achieve this. In the particular embodiment, the numerically lower address is always placed before the numerically higher address. Such least to highest order is used to get the best spread of signatures and hashes for the lookup operations. In this case, therefore, since we assume "S_1"<"C_1", the order is address "S_1" followed by client address "C_1". The next field used to build the signature is a protocol field 242 extracted from packet 206's field 216, and thus is the protocol "p^1". The next field used for the signature is field 243, which contains the destination source port number shown as a crosshatched pattern from the field 218 of the packet 206. This pattern will be recognized in the payload of packets to derive how this packet or sequence of packets exists as a flow. In practice, these may be TCP port numbers, or a combination of TCP port numbers. In the case of the Sun RPC example, the crosshatch represents a set of port numbers of UDS for p^1 that will be used to recognize this flow (*e.g., port* 111). Port 111 indicates this is Sun RPC. Some applications, such as the Sun RPC Bind Lookups, are directly determinable ("known") at the parser level. So in this case, the signature KEY-1 points to a known application denoted "a^1" (Sun RPC Bind Lookup), and a next-state that the state processor should proceed to for more complex recognition jobs, denoted as state "st_D" is placed in the field 245 of the flow-entry.

[00236] When the Sun RPC Bind Lookup reply is acquired, a flow signature is again built by the parser. This flow signature is identical to KEY-1. Hence, when the signature enters the analyzer subsystem 303 from the parser subsystem 301, the complete flow-entry is obtained, and in this flow-entry indicates state "st_D". The operations for state "st_D" in the state processor instruction database 326 instructs the state processor to build and store a new flow signature, shown as KEY-2 (212) in FIG. 2. This flow signature built by the state processor also includes the destination and a source addresses 250 and 251, respectively, for server "S_1" followed by (the numerically higher address) client "C_1". A protocol field 252 defines the protocol to be used, *e.g.*, "p^2" which is obtained from the reply packet. A field 253 contains a recognition pattern also obtained from the reply packet. In this case, the application is Sun RPC, and field 254 indicates this application "a^2". A next-state field 255 defines the next state that the state processor should proceed to for more complex recognition jobs, *e.g.*, a state "st^1". In this particular example, this is a final state. Thus, KEY-2 may now be used to recognize packets that are in any way associated with the application "a^2". Two such packets 208 and 209 are shown, one in each direction. They use the particular application service requested in the original Bind Lookup Request, and each will be recognized because the signature KEY-2 will be built in each case.

[00237] The two flow signatures 210 and 212 always order the destination and source address fields with server "S_1" followed by client "C_1". Such values are automatically filled in when the addresses are first created in a particular flow signature. Preferably, large collections of flow signatures are kept in a lookup table in a least-to-highest order for the best spread of flow signatures and hashes.

[00238] Thereafter, the client and server exchange a number of packets, *e.g.*, represented by request packet 208 and response packet 209. The client 106 sends packets 208 that have a destination and source address S_1 and C_1, in a pair of fields 260 and 261. A field 262 defines the protocol as "p^2", and a field 263 defines the destination port number.

[00239] Some network-server application recognition jobs are so simple that only a single state transition has to occur to be able to pinpoint the application that produced the packet. Others require a sequence of state transitions to occur in order to match a known and predefined climb from state-to-state.

[00240] Thus the flow signature for the recognition of application "a^2" is automatically set up by predefining what packet-exchange sequences occur for this example when a relatively simple Sun Microsystems Remote Procedure Call bind lookup request instruction executes. More complicated exchanges than this may generate more than two flow signatures and their corresponding states. Each recognition may involve setting up a complex state transition diagram to be traversed before a "final" resting state such as "st_1" in field 255 is reached. All these are used to build the final set of flow signatures for recognizing a particular application in the future.

[00241] Embodiments of the present invention automatically generate flow signatures with the necessary recognition patterns and state transition climb procedure. Such comes from analyzing packets according to parsing rules, and also generating state transitions to search for. Applications and protocols, at any level, are recognized through state analysis of sequences of packets.

[00242] Note that one in the art will understand that computer networks are used to connect many different types of devices, including network appliances such as telephones, "Internet" radios, pagers, and so forth. The term computer as used herein encompasses all such devices and a computer network as used herein includes networks of such computers.

[00243] Although the present invention has been described in terms of the presently preferred embodiments, it is to be understood that the disclosure is not to be interpreted as limiting. Various alterations and modifications will no doubt become apparent to those or ordinary skill in the art after having read the above disclosure. Accordingly, it is intended that the claims be interpreted as covering all alterations and modifications as fall within the true spirit and scope of the present invention.

CLAIM

We claim:

1. A packet monitor for examining packets passing through a connection point on a computer network in real-time, the packets provided to the packet monitor via a packet acquisition device connected to the connection point, the packet monitor comprising:

   (a) a packet-buffer memory configured to accept a packet from the packet acquisition device;

   (b) a parsing/extraction operations memory configured to store a database of parsing/extraction operations that includes information describing how to determine at least one of the protocols used in a packet from data in the packet;

   (c) a parser subsystem coupled to the packet buffer and to the pattern/extraction operations memory, the parser subsystem configured to examine the packet accepted by the buffer, extract selected portions of the accepted packet, and form a function of the selected portions sufficient to identify that the accepted packet is part of a conversational flow-sequence;

   (d) a memory storing a flow-entry database including a plurality of flow-entries for conversational flows encountered by the monitor;

   (e) a lookup engine connected to the parser subsystem and to the flow-entry database, and configured to determine using at least some of the selected portions of the accepted packet if there is an entry in the flow-entry database for the conversational flow sequence of the accepted packet;

   (f) a state patterns/operations memory configured to store a set of predefined state transition patterns and state operations such that traversing a particular transition pattern as a result of a particular conversational flow-sequence of packets indicates that the particular conversational flow-sequence is associated with the operation of a particular application program, visiting each state in a traversal including carrying out none or more predefined state operations;

(g) a protocol/state identification mechanism coupled to the state patterns/operations memory and to the lookup engine, the protocol/state identification engine configured to determine the protocol and state of the conversational flow of the packet; and

(h) a state processor coupled to the flow-entry database, the protocol/state identification engine, and to the state patterns/operations memory, the state processor, configured to carry out any state operations specified in the state patterns/operations memory for the protocol and state of the flow of the packet,

the carrying out of the state operations furthering the process of identifying which application program is associated with the conversational flow-sequence of the packet, the state processor progressing through a series of states and state operations until there are no more state operations to perform for the accepted packet, in which case the state processor updates the flow-entry, or until a final state is reached that indicates that no more analysis of the flow is required, in which case the result of the analysis is announced.

2. A packet monitor according to claim 1, wherein the flow-entry includes the state of the flow, such that the protocol/state identification mechanism determines the state of the packet from the flow-entry in the case that the lookup engine finds a flow-entry for the flow of the accepted packet.

3. A packet monitor according to claim 1, wherein the parser subsystem includes a mechanism for building a hash from the selected portions, and wherein the hash is used by the lookup engine to search the flow-entry database, the hash designed to spread the flow-entries across the flow-entry database.

4. A packet monitor according to claim 1, further comprising:

a compiler processor coupled to the parsing/extraction operations memory, the compiler processor configured to run a compilation process that includes:

receiving commands in a high-level protocol description language that describe the protocols that may be used in packets encountered by the monitor, and translating the protocol description language commands into a plurality of parsing/extraction operations that are initialized into the parsing/extraction operations memory.

5. A packet monitor according to claim 4, wherein the protocol description language commands also describe a correspondence between a set of one or more application programs and the state transition patterns/operations that occur as a result of particular conversational flow-sequences associated with an application program, wherein the compiler processor is also coupled to the state patterns/operations memory, and wherein the compilation process further includes translating the protocol description language commands into a plurality of state patterns and state operations that are initialized into the state patterns/operations memory.

6. A packet monitor according to claim 1, further comprising:

a cache memory coupled to and between the lookup engine and the flow-entry database providing for fast access of a set of likely-to-be-accessed flow-entries from the flow-entry database.

7. A packet monitor according to claim 6, wherein the cache functions as a fully associative, least-recently-used cache memory.

8. A packet monitor according to claim 7, wherein the cache functions as a fully associative, least-recently-used cache memory and includes content addressable memories configured as a stack.

9. A packet monitor according to claim 1, wherein one or more statistical measures about a flow are stored in each flow-entry, the packet monitor further comprising:

a calculator for updating the statistical measures in a flow-entry of the accepted packet.

10. A packet monitor according to claim 9, wherein, when the application program of a flow is determined, one or more network usage metrics related to said application and determined from the statistical measures are presented to a user for network performance monitoring.

11. A method of examining packets passing through a connection point on a computer network, each packets conforming to one or more protocols, the method comprising:

(a) receiving a packet from a packet acquisition device;

(b) performing one or more parsing/extraction operations on the packet to create a parser record comprising a function of selected portions of the packet;

(c) looking up a flow-entry database comprising none or more flow-entries for previously encountered conversational flows, the looking up using at least some of the selected packet portions and determining if the packet is of an existing flow;

(d) if the packet is of an existing flow, classifying the packet as belonging to the found existing flow; and

(e) if the packet is of a new flow, storing a new flow-entry for the new flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry,

wherein the parsing/extraction operations depend on one or more of the protocols to which the packet conforms.

12. A method according to claim 11, wherein each packet passing through the connection point is examined in real time.

13. A method according to claim 11, wherein classifying the packet as belonging to the found existing flow includes updating the flow-entry of the existing flow.

14. A method according to claim 13, wherein updating includes storing one or more statistical measures stored in the flow-entry of the existing flow.

15. A method according to claim 14, wherein the one or more statistical measures include measures selected from the set consisting of the total packet count for the flow, the time, and a differential time from the last entered time to the present time.

16. A method according to claim 11, wherein the function of the selected portions of the packet forms a signature that includes the selected packet portions and that can identify future packers, wherein the lookup operation uses the signature and wherein the

identifying information stored in the new or updated flow-entry is a signature for identifying future packets.

17. A method according to claim 11, wherein at least one of the protocols of the packet uses source and destination addresses, and wherein the selected portions of the packet include the source and destination addresses.

18. A method according to claim 17, wherein the function of the selected portions for packets of the same flow is consistent independent of the direction of the packets.

19. A method according to claim 18, wherein the source and destination addresses are placed in an order determined by the order of numerical values of the addresses in the function of selected portions.

20. A method according to claim 19, wherein the numerically lower address is placed before the numerically higher address in the function of selected portions.

21. A method according to claim 11, wherein the looking up of the flow-entry database uses a hash of the selected packet portions.

22. A method according to claim 11, wherein the parsing/extraction operations are according to a database of parsing/extraction operations that includes information describing how to determine a set of one or more protocol dependent extraction operations from data in the packet that indicate a protocol used in the packet.

23. A method according to claim 11, wherein step (d) includes if the packet is of an existing flow, obtaining the last encountered state of the flow and performing any state operations specified for the state of the flow starting from the last encountered state of the flow; and wherein step (e) includes if the packet is of a new flow, performing any state operations required for the initial state of the new flow.

24. A method according to claim 23, wherein the state processing of each received packet of a flow furthers the identifying of the application program of the flow.

25. A method according to claim 23, wherein the state operations include updating the flow-entry, including storing identifying information for future packets to be identified with the flow-entry.

26. A method according to claim 25, wherein the state processing of each received packet of a flow furthers the identifying of the application program of the flow.

27. A method according to claim 23, wherein the state operations include searching the parser record for the existence of one or more reference strings.

28. A method according to claim 23, wherein the state operations are carried out by a programmable state processor according to a database of protocol dependent state operations.

29. A packet monitor for examining packets passing through a connection point on a computer network, each packets conforming to one or more protocols, the monitor comprising:

(a) a packet acquisition device coupled to the connection point and configured to receive packets passing through the connection point;

(b) an input buffer memory coupled to and configured to accept a packet from the packet acquisition device;

(c) a parser subsystem coupled to the input buffer memory and including a slicer, the parsing subsystem configured to extract selected portions of the accepted packet and to output a parser record containing the selected portions;

(d) a memory for storing a database comprising none or more flow-entries for previously encountered conversational flows, each flow-entry identified by identifying information stored in the flow-entry;

(e) a lookup engine coupled to the output of the parser subsystem and to the flow-entry memory and configured to lookup whether the particular packet whose parser record is output by the parser subsystem has a matching flow-entry, the looking up using at least some of the selected packet portions and determining if the packet is of an existing flow; and

(f) a flow insertion engine coupled to the flow-entry memory and to the lookup engine and configured to create a flow-entry in the flow-entry database, the flow-entry including identifying information for future packets to be identified with the new flow-entry,

the lookup engine configured such that if the packet is of an existing flow, the monitor classifies the packet as belonging to the found existing flow; and if the packet is of a new flow, the flow insertion engine stores a new flow-entry for the new flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry,

wherein the operation of the parser subsystem depends on one or more of the protocols to which the packet conforms.

30. A monitor according to claim 29, wherein each packet passing through the connection point is accepted by the packet buffer memory and examined by the monitor in real time.

31. A monitor according to claim 29, wherein the lookup engine updates the flow-entry of an existing flow in the case that the lookup is successful.

32. A monitor according to claim 29, further including a mechanism for building a hash from the selected portions, wherein the hash is included in the input for a particular packet to the lookup engine, and wherein the hash is used by the lookup engine to search the flow-entry database.

33. A monitor according to claim 29, further including a memory containing a database of parsing/extraction operations, the parsing/extraction database memory coupled to the parser subsystem, wherein the parsing/extraction operations are according to one or more parsing/extraction operations looked up from the parsing/extraction database.

34. A monitor according to claim 33, wherein the database of parsing/extraction operations includes information describing how to determine a set of one or more

protocol dependent extraction operations from data in the packet that indicate a protocol used in the packet.

35. A monitor according to claim 29, further including a flow-key-buffer (UFKB) coupled to the output of the parser subsystem and to the lookup engine and to the flow insertion engine, wherein the output of the parser monitor is coupled to the lookup engine via the UFKB, and wherein the flow insertion engine is coupled to the lookup engine via the UFKB.

36. A method according to claim 29, further including a state processor coupled to the lookup engine and to the flow-entry-database memory, and configured to perform any state operations specified for the state of the flow starting from the last encountered state of the flow in the case that the packet is from an existing flow, and to perform any state operations required for the initial state of the new flow in the case that the packet is from an existing flow.

37. A method according to claim 29, wherein the set of possible state operations that the state processor is configured to perform includes searching for one or more patterns in the packet portions.

38. A monitor according to claim 36, wherein the state processor is programmable, the monitor further including a state patterns/operations memory coupled to the state processor, the state operations memory configured to store a database of protocol dependent state patterns/operations.

39. A monitor according to claim 35, further including a state processor coupled to the UFKB and to the flow-entry-database memory, and configured to perform any state operations specified for the state of the flow starting from the last encountered state of the flow in the case that the packet is from an existing flow, and to perform any state operations required for the initial state of the new flow in the case that the packet is from an existing flow.

40. A monitor according to claim 36, wherein the state operations include updating the flow-entry, including identifying information for future packets to be identified with the flow-entry.

41. A packet monitor according to claim 29, further comprising:

a compiler processor coupled to the parsing/extraction operations memory, the compiler processor configured to run a compilation process that includes:

receiving commands in a high-level protocol description language that describe the protocols that may be used in packets encountered by the monitor and any children protocols thereof, and

translating the protocol description language commands into a plurality of parsing/extraction operations that are initialized into the parsing/extraction operations memory.

42. A packet monitor according to claim 38, further comprising:

a compiler processor coupled to the parsing/extraction operations memory, the compiler processor configured to run a compilation process that includes:

receiving commands in a high-level protocol description language that describe a correspondence between a set of one or more application programs and the state transition patterns/operations that occur as a result of particular conversational flow-sequences associated with an application programs, and

translating the protocol description language commands into a plurality of state patterns and state operations that are initialized into the state patterns/operations memory.

43. A packet monitor according to claim 29, further comprising:

a cache subsystem coupled to and between the lookup engine and the flow-entry database memory providing for fast access of a set of likely-to-be-accessed flow-entries from the flow-entry database.

44. A packet monitor according to claim 43, wherein the cache subsystem is an associative cache subsystem including one or more content addressable memory cells (CAMs).

45. A packet monitor according to claim 44, wherein the cache subsystem is also a least-recently-used cache memory such that a cache miss updates the least recently used cache entry.

46. A packet monitor according to claim 29, wherein each flow-entry stores one or more statistical measures about the flow, the monitor further comprising

a calculator for updating at least one of the statistical measures in the flow-entry of the accepted packet.

47. A packet monitor according to claim 46, wherein the one or more statistical measures include measures selected from the set consisting of the total packet count for the flow, the time, and a differential time from the last entered time to the present time.

48. A packet monitor according to claim 46, further including a statistical processor configured to determine one or more network usage metrics related to the flow from one or more of the statistical measures in a flow-entry.

49. A monitor according to claim 29, wherein:

flow-entry-database is organized into a plurality of bins that each contain N-number of flow-entries, and wherein said bins are accessed via a hash data value created by a parser subsystem based on the selected packet portions, wherein N is one or more.

50. A monitor according to claim 49, wherein the hash data value is used to spread a plurality of flow-entries across the flow-entry-database and allows fast lookup of a flow-entry and shallower buckets.

51. A monitor according to claim 36, wherein the state processor analyzes both new and existing flows in order to classify them by application and proceeds from state-to-state based on a set of predefined rules.

52. A monitor according to claim 29, wherein the lookup engine begins processing as soon as a parser record arrives from the parser subsystem.

53. A monitor according to claim 36, wherein the lookup engine provides for flow state entry checking to see if a flow key should be sent to the state processor, and that outputs a protocol identifier for the flow.

54. A method of examining packets passing through a connection point on a computer network, the method comprising:

(a) receiving a packet from a packet acquisition device;

(b) performing one or more parsing/extraction operations on the packet according to a database of parsing/extraction operations to create a parser record comprising a function of selected portions of the packet, the database of parsing/extraction operations including information on how to determine a set of one or more protocol dependent extraction operations from data in the packet that indicate a protocol is used in the packet;

(c) looking up a flow-entry database comprising none or more flow-entries for previously encountered conversational flows, the looking up using at least some of the selected packet portions, and determining if the packet is of an existing flow;

(d) if the packet is of an existing flow, obtaining the last encountered state of the flow and performing any state operations specified for the state of the flow starting from the last encountered state of the flow; and

(e) if the packet is of a new flow, performing any analysis required for the initial state of the new flow and storing a new flow-entry for the new flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry.

55. A method according to claim 54, wherein one of the state operations specified for at least one of the states includes updating the flow-entry, including identifying information for future packets to be identified with the flow-entry.

56. A method according to claim 54, wherein one of the state operations specified for at least one of the states includes searching the contents of the packet for at least one reference string.

57. A method according to claim 55, wherein one of the state operations specified for at least one of the states includes creating a new flow-entry for future packets to be identified with the flow, the new flow-entry including identifying information for future packets to be identified with the flow-entry.

58. A method according to claim 54, further comprising forming a signature from the selected packet portions, wherein the lookup operation uses the signature and wherein the identifying information stored in the new or updated flow-entry is a signature for identifying future packets.

59. A method according to claim 54, wherein the state operations are according to a database of protocol dependent state operations.

ABSTRACT

A monitor for and a method of examining packets passing through a connection point on a computer network. Each packets conforms to one or more protocols. The method includes receiving a packet from a packet acquisition device and performing one or more parsing/extraction operations on the packet to create a parser record comprising a function of selected portions of the packet. The parsing/extraction operations depend on one or more of the protocols to which the packet conforms. The method further includes looking up a flow-entry database containing flow-entries for previously encountered conversational flows. The lookup uses the selected packet portions and determining if the packet is of an existing flow. If the packet is of an existing flow, the method classifies the packet as belonging to the found existing flow, and if the packet is of a new flow, the method stores a new flow-entry for the new flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry. For the packet of an existing flow, the method updates the flow-entry of the existing flow. Such updating may include storing one or more statistical measures. Any stage of a flow, state is maintained, and the method performs any state processing for an identified state to further the process of identifying the flow. The method thus examines each and every packet passing through the connection point in real time until the application program associated with the conversational flow is determined.

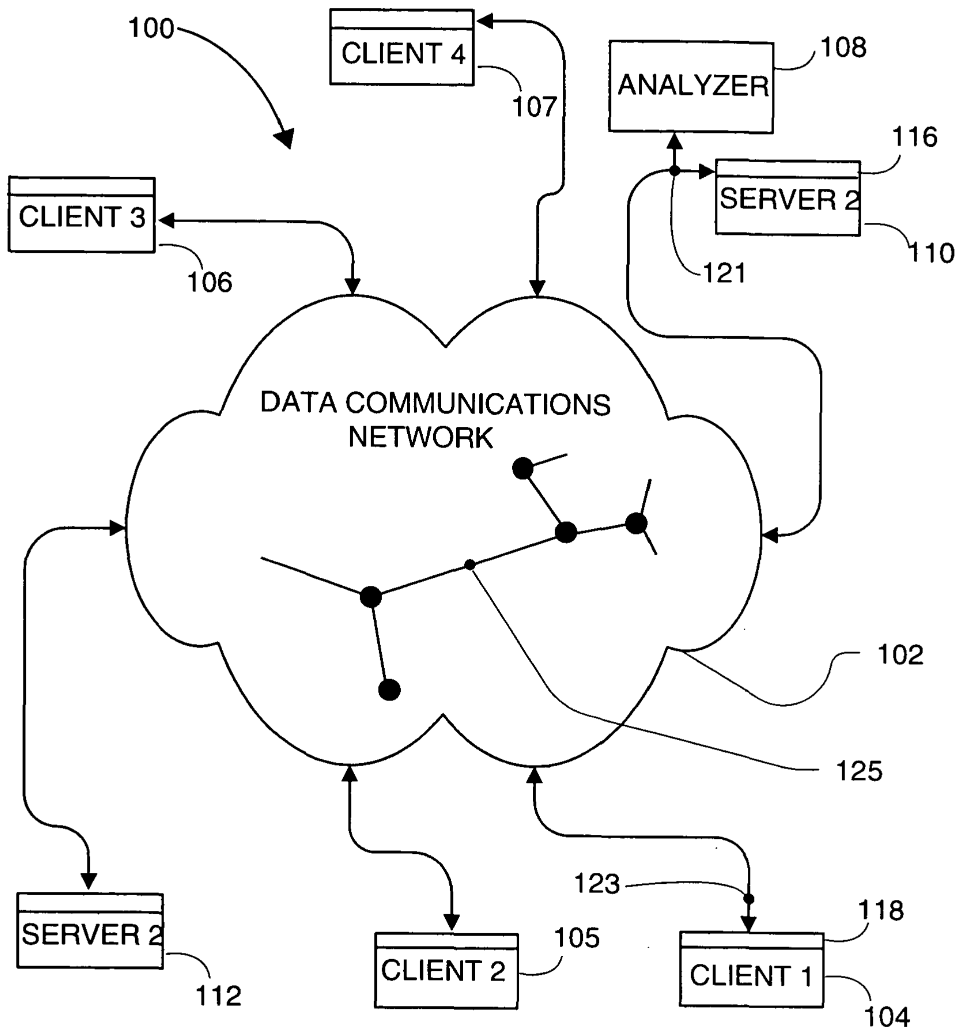

100
CLIENT 4
107
ANALYZER
108
116
SERVER 2
110
121
CLIENT 3
106
DATA COMMUNICATIONS NETWORK
102
125
123
118
SERVER 2
112
CLIENT 2
105
CLIENT 1
104

FIG. 1

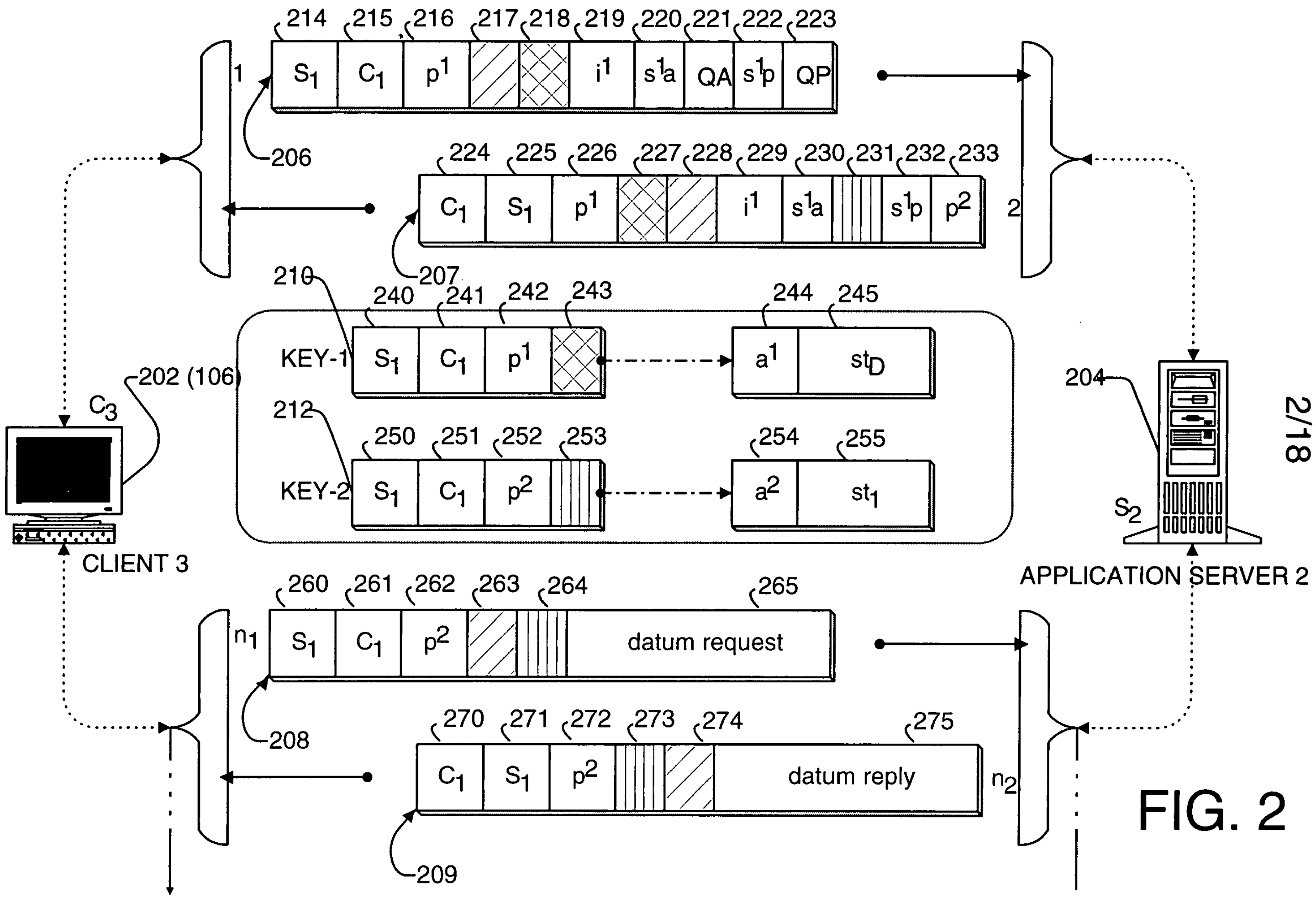

214
215
216
217
218
219
220
221
222
223
S1
C1
p1
i1
s1a
QA
s1p
QP
1
206
224
225
226
227
228
229
230
231
232
233
C1
S1
p1
i1
s1a
s1p
p2
2
207
210
240
241
242
243
KEY-1
S1
C1
p1
244
245
a1
stD
212
250
251
252
253
KEY-2
S1
C1
p2
254
255
a2
st1
202 (106)
C3
CLIENT 3
204
S2
APPLICATION SERVER 2
260
261
262
263
264
265
n1
S1
C1
p2
datum request
208
270
271
272
273
274
275
C1
S1
p2
datum reply
n2
209

FIG. 2

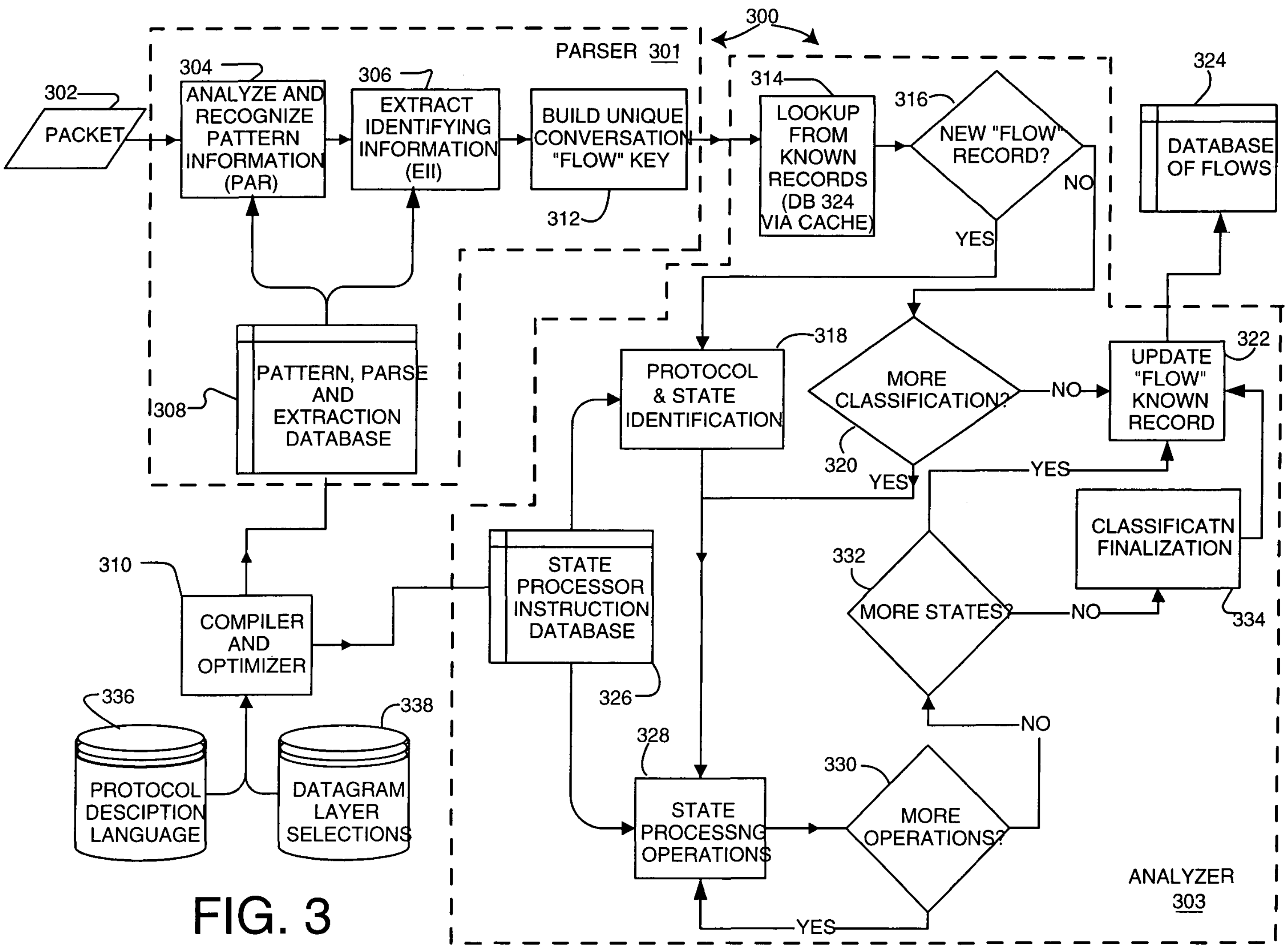

300
PARSER 301
302
PACKET
304
ANALYZE AND RECOGNIZE PATTERN INFORMATION (PAR)
306
EXTRACT IDENTIFYING INFORMATION (EII)
BUILD UNIQUE CONVERSATION "FLOW" KEY
312
308
PATTERN, PARSE AND EXTRACTION DATABASE
314
LOOKUP FROM KNOWN RECORDS (DB 324 VIA CACHE)
316
NEW "FLOW" RECORD?
NO
YES
324
DATABASE OF FLOWS
318
PROTOCOL & STATE IDENTIFICATION
MORE CLASSIFICATION?
NO
320
YES
322
UPDATE "FLOW" KNOWN RECORD
YES
CLASSIFICATN FINALIZATION
332
MORE STATES?
NO
334
310
COMPILER AND OPTIMIZER
STATE PROCESSOR INSTRUCTION DATABASE
326
336
PROTOCOL DESCIPTION LANGUAGE
338
DATAGRAM LAYER SELECTIONS
328
STATE PROCESSNG OPERATIONS
330
MORE OPERATIONS?
NO
YES
ANALYZER 303

FIG. 3

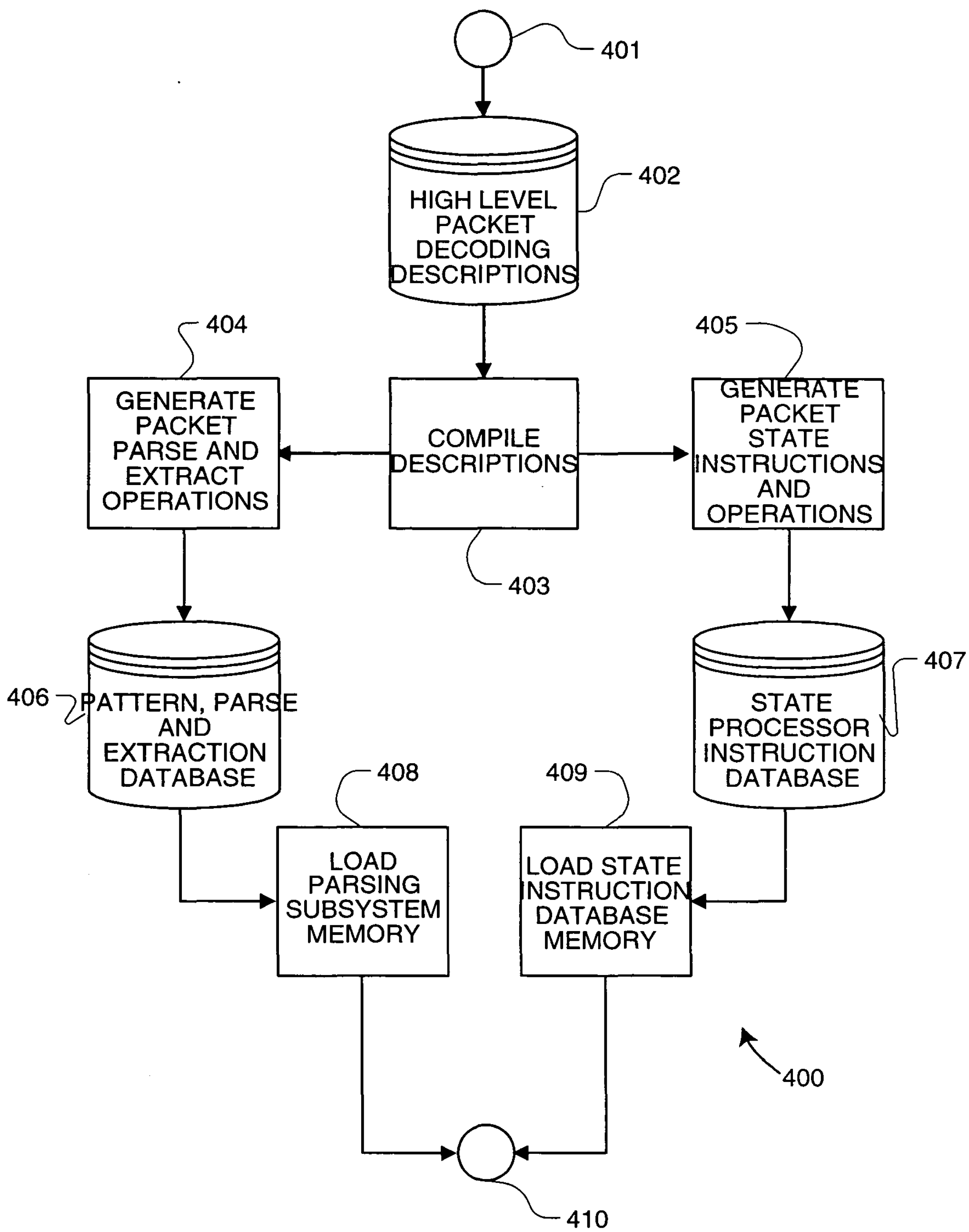

401
HIGH LEVEL PACKET DECODING DESCRIPTIONS
402
404
GENERATE PACKET PARSE AND EXTRACT OPERATIONS
COMPILE DESCRIPTIONS
403
405
GENERATE PACKET STATE INSTRUCTIONS AND OPERATIONS
406
PATTERN, PARSE AND EXTRACTION DATABASE
407
STATE PROCESSOR INSTRUCTION DATABASE
408
LOAD PARSING SUBSYSTEM MEMORY
409
LOAD STATE INSTRUCTION DATABASE MEMORY
400
410

FIG. 4

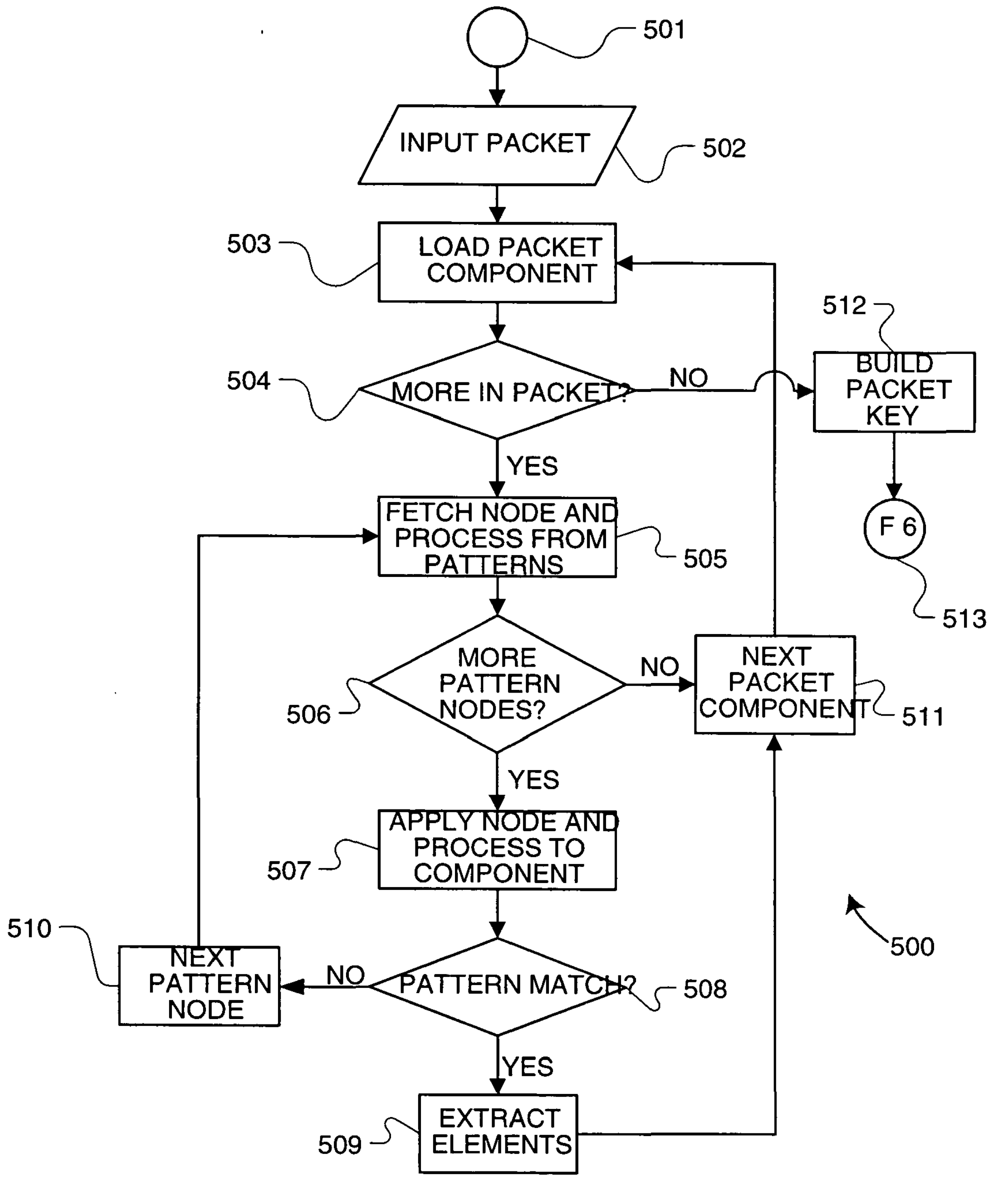
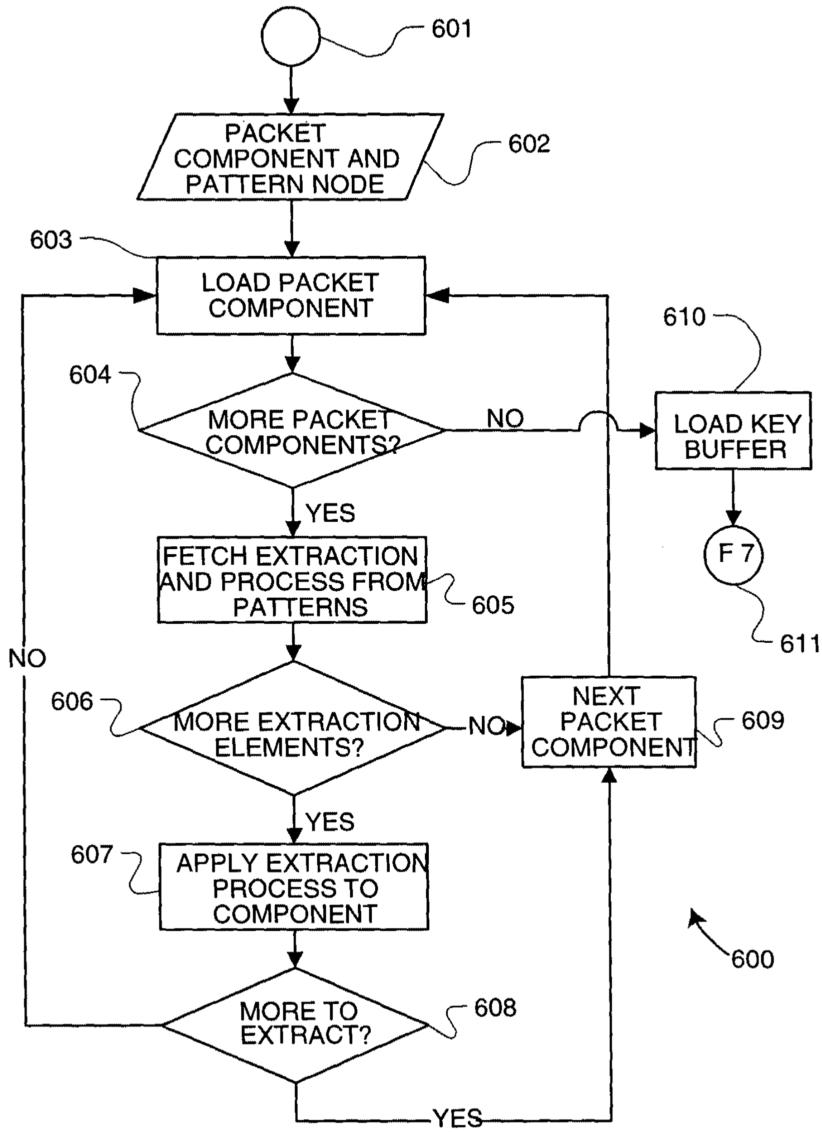

501
INPUT PACKET
502
503
LOAD PACKET COMPONENT
504
MORE IN PACKET?
NO
512
BUILD PACKET KEY
F 6
513
YES
FETCH NODE AND PROCESS FROM PATTERNS
505
MORE PATTERN NODES?
506
NO
NEXT PACKET COMPONENT
511
YES
APPLY NODE AND PROCESS TO COMPONENT
507
510
NEXT PATTERN NODE
NO
PATTERN MATCH?
508
500
YES
EXTRACT ELEMENTS
509

FIG. 5



601
PACKET COMPONENT AND PATTERN NODE
602
603
LOAD PACKET COMPONENT
604
MORE PACKET COMPONENTS?
NO
610
LOAD KEY BUFFER
F 7
611
YES
FETCH EXTRACTION AND PROCESS FROM PATTERNS
605
NO
606
MORE EXTRACTION ELEMENTS?
NO
NEXT PACKET COMPONENT
609
YES
607
APPLY EXTRACTION PROCESS TO COMPONENT
600
MORE TO EXTRACT?
608
YES

FIG. 6

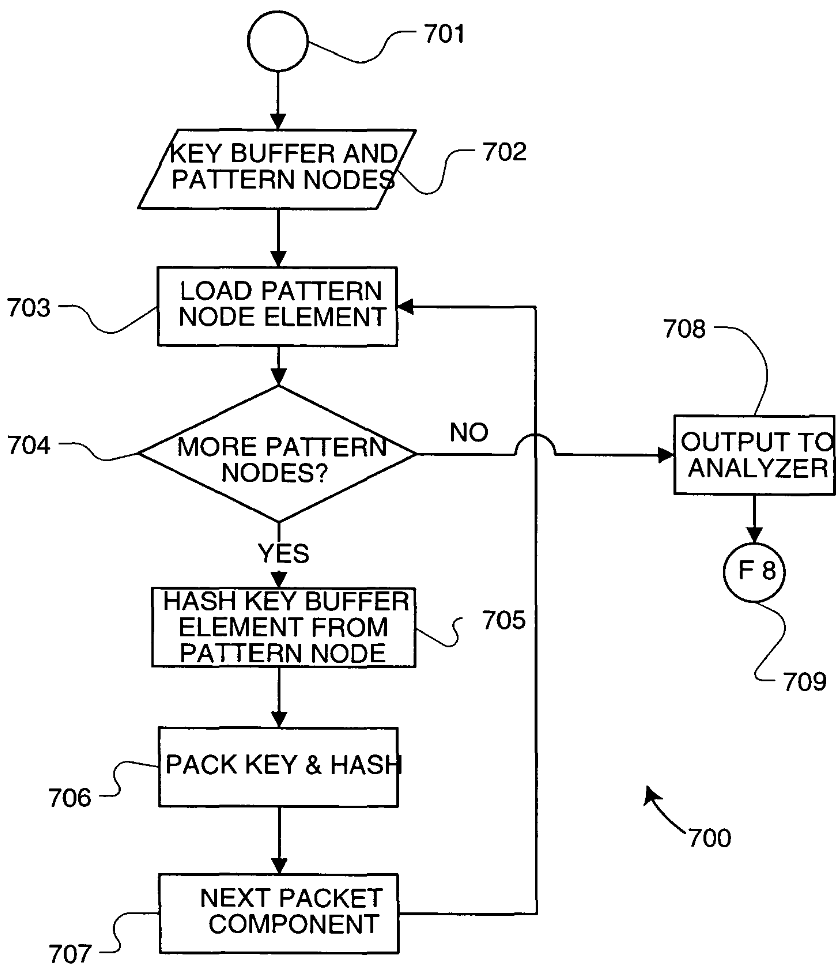

701
KEY BUFFER AND PATTERN NODES
702
703
LOAD PATTERN NODE ELEMENT
704
MORE PATTERN NODES?
NO
YES
708
OUTPUT TO ANALYZER
F 8
709
HASH KEY BUFFER ELEMENT FROM PATTERN NODE
705
706
PACK KEY & HASH
NEXT PACKET COMPONENT
707
700

FIG. 7

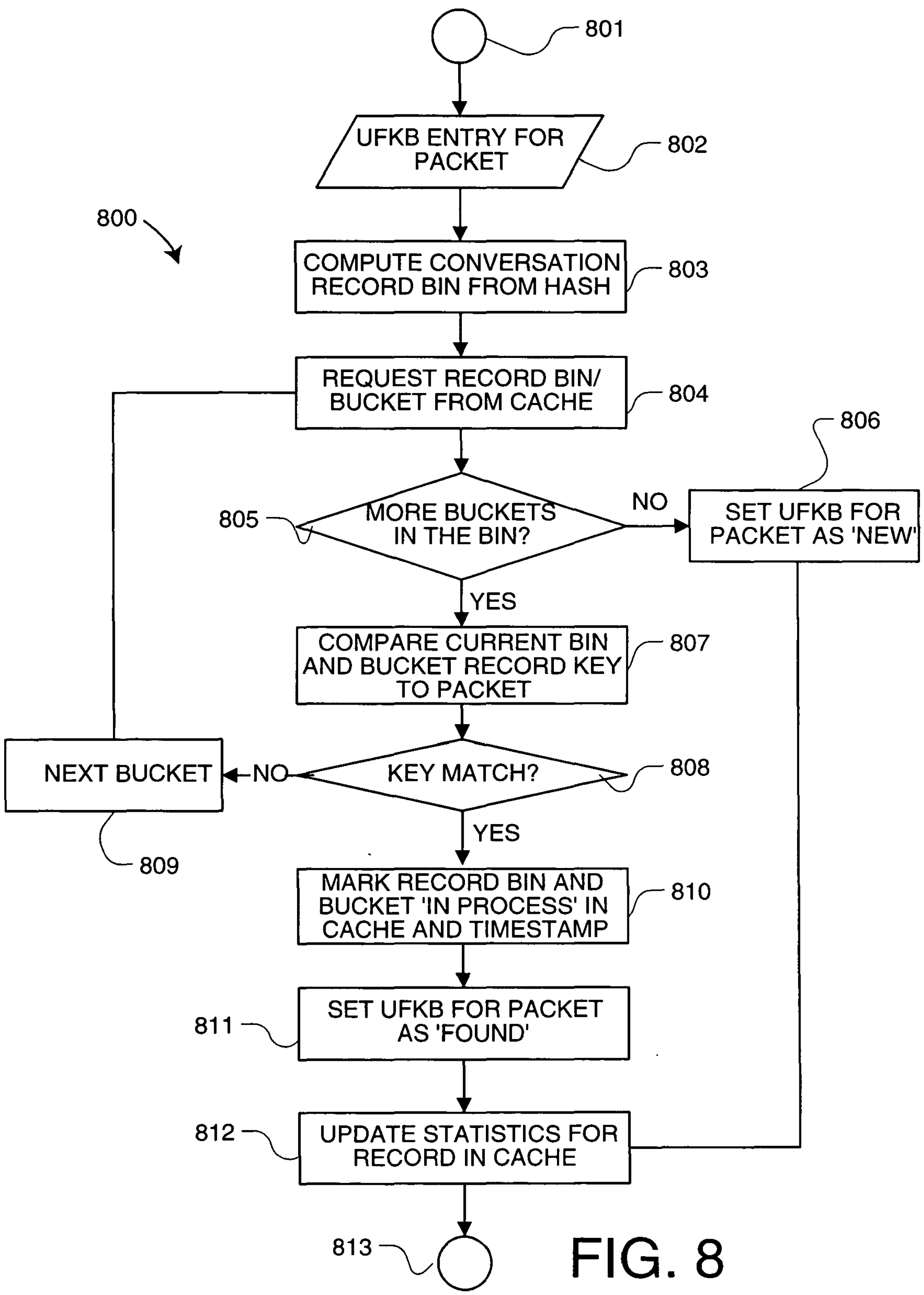

801
UFKB ENTRY FOR PACKET
802
800
COMPUTE CONVERSATION RECORD BIN FROM HASH
803
REQUEST RECORD BIN/ BUCKET FROM CACHE
804
806
805
MORE BUCKETS IN THE BIN?
NO
SET UFKB FOR PACKET AS 'NEW'
YES
COMPARE CURRENT BIN AND BUCKET RECORD KEY TO PACKET
807
NEXT BUCKET
NO
KEY MATCH?
808
809
YES
MARK RECORD BIN AND BUCKET 'IN PROCESS' IN CACHE AND TIMESTAMP
810
811
SET UFKB FOR PACKET AS 'FOUND'
812
UPDATE STATISTICS FOR RECORD IN CACHE
813

FIG. 8

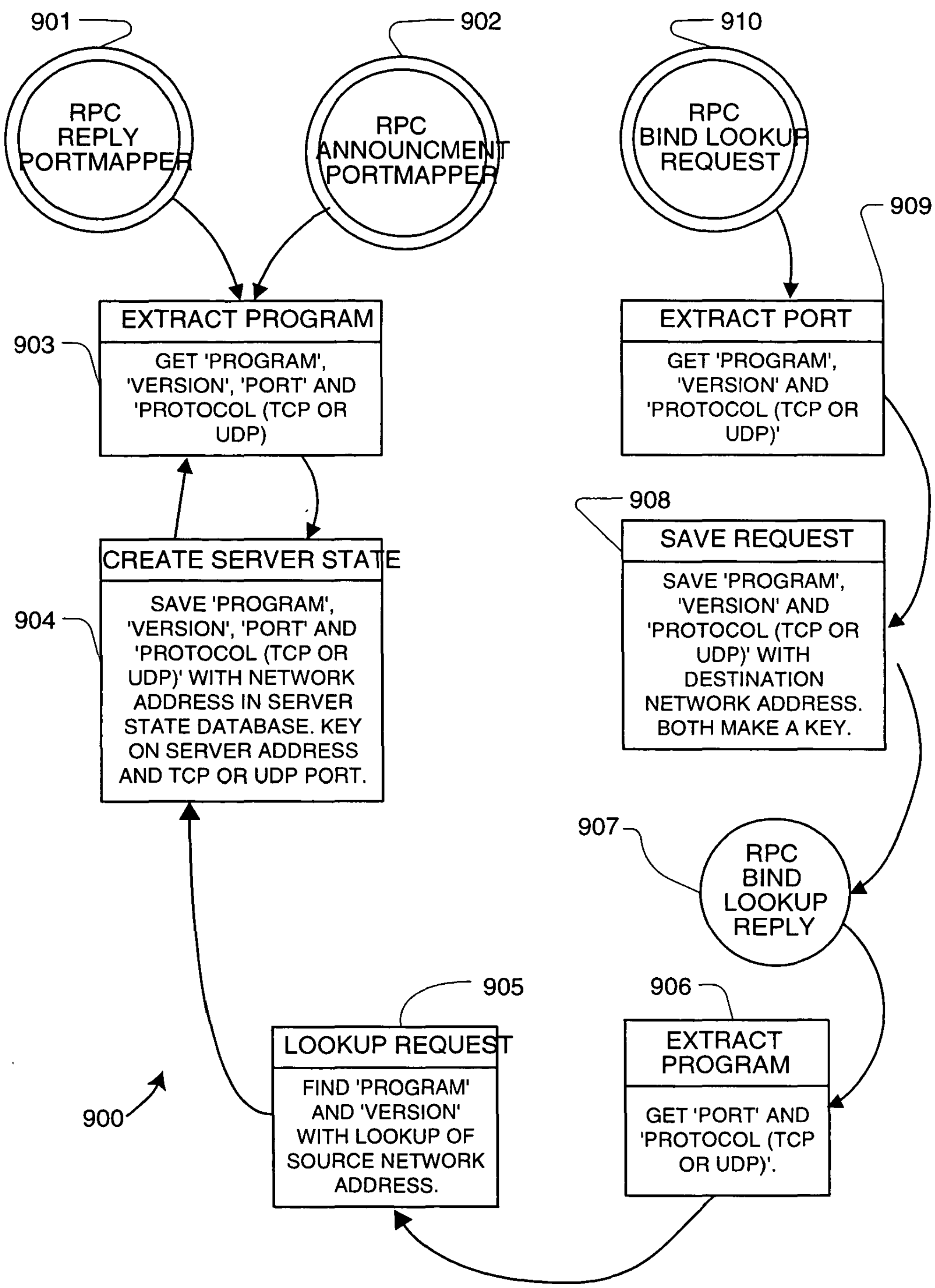

901
902
910
RPC REPLY PORTMAPPER
RPC ANNOUNCMENT PORTMAPPER
RPC BIND LOOKUP REQUEST
909
903
EXTRACT PROGRAM
GET 'PROGRAM', 'VERSION', 'PORT' AND 'PROTOCOL (TCP OR UDP)
EXTRACT PORT
GET 'PROGRAM', 'VERSION' AND 'PROTOCOL (TCP OR UDP)'
908
SAVE REQUEST
SAVE 'PROGRAM', 'VERSION' AND 'PROTOCOL (TCP OR UDP)' WITH DESTINATION NETWORK ADDRESS. BOTH MAKE A KEY.
904
CREATE SERVER STATE
SAVE 'PROGRAM', 'VERSION', 'PORT' AND 'PROTOCOL (TCP OR UDP)' WITH NETWORK ADDRESS IN SERVER STATE DATABASE. KEY ON SERVER ADDRESS AND TCP OR UDP PORT.
907
RPC BIND LOOKUP REPLY
905
906
LOOKUP REQUEST
FIND 'PROGRAM' AND 'VERSION' WITH LOOKUP OF SOURCE NETWORK ADDRESS.
EXTRACT PROGRAM
GET 'PORT' AND 'PROTOCOL (TCP OR UDP)'.
900

FIG. 9

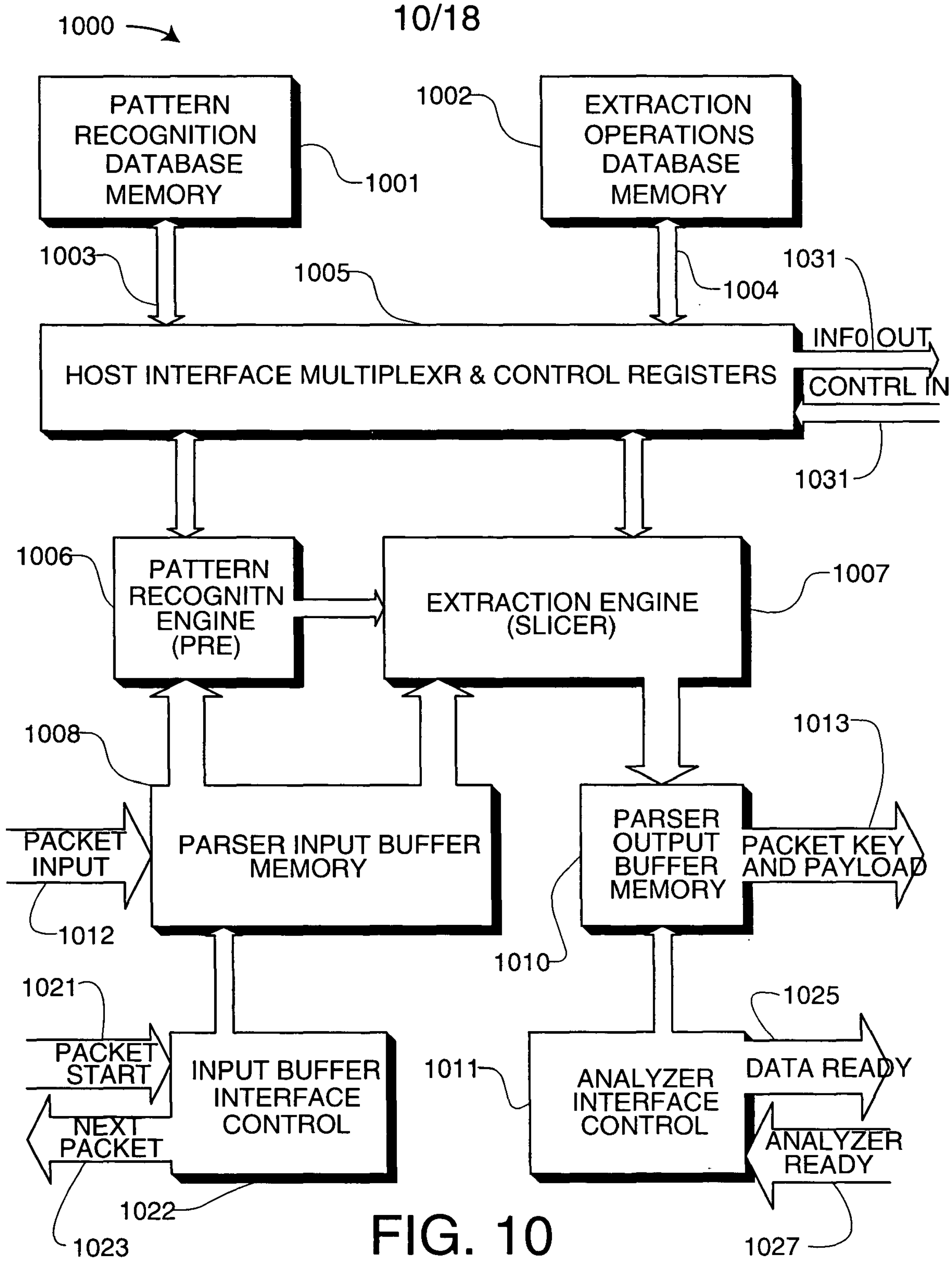

1000
PATTERN RECOGNITION DATABASE MEMORY
1001
1002
EXTRACTION OPERATIONS DATABASE MEMORY
1003
1005
1004
1031
HOST INTERFACE MULTIPLEXR & CONTROL REGISTERS
INF0 OUT
CONTRL IN
1031
1006
PATTERN RECOGNITN ENGINE (PRE)
EXTRACTION ENGINE (SLICER)
1007
1008
1013
PACKET INPUT
PARSER INPUT BUFFER MEMORY
PARSER OUTPUT BUFFER MEMORY
PACKET KEY AND PAYLOAD
1012
1010
1021
1025
PACKET START
INPUT BUFFER INTERFACE CONTROL
1011
ANALYZER INTERFACE CONTROL
DATA READY
NEXT PACKET
ANALYZER READY
1022
1023
1027

FIG. 10

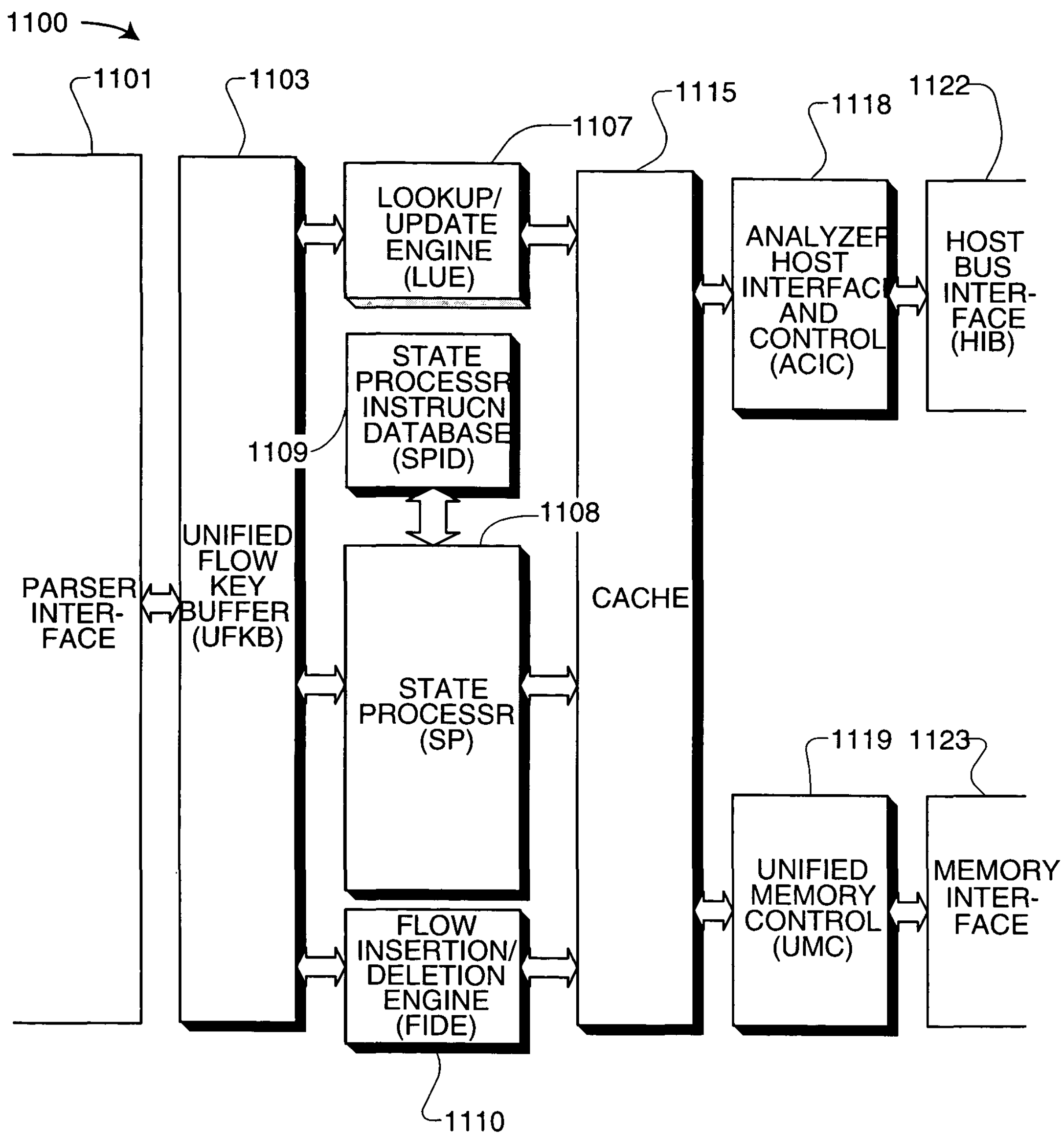

1100
1101
1103
1107
1115
1118
1122
LOOKUP/ UPDATE ENGINE (LUE)
STATE PROCESSR INSTRUCN DATABASE (SPID)
1109
1108
UNIFIED FLOW KEY BUFFER (UFKB)
PARSER INTER-FACE
CACHE
ANALYZER HOST INTERFACE AND CONTROL (ACIC)
HOST BUS INTER-FACE (HIB)
STATE PROCESSR (SP)
1119
1123
UNIFIED MEMORY CONTROL (UMC)
MEMORY INTER-FACE
FLOW INSERTION/ DELETION ENGINE (FIDE)
1110

FIG. 11

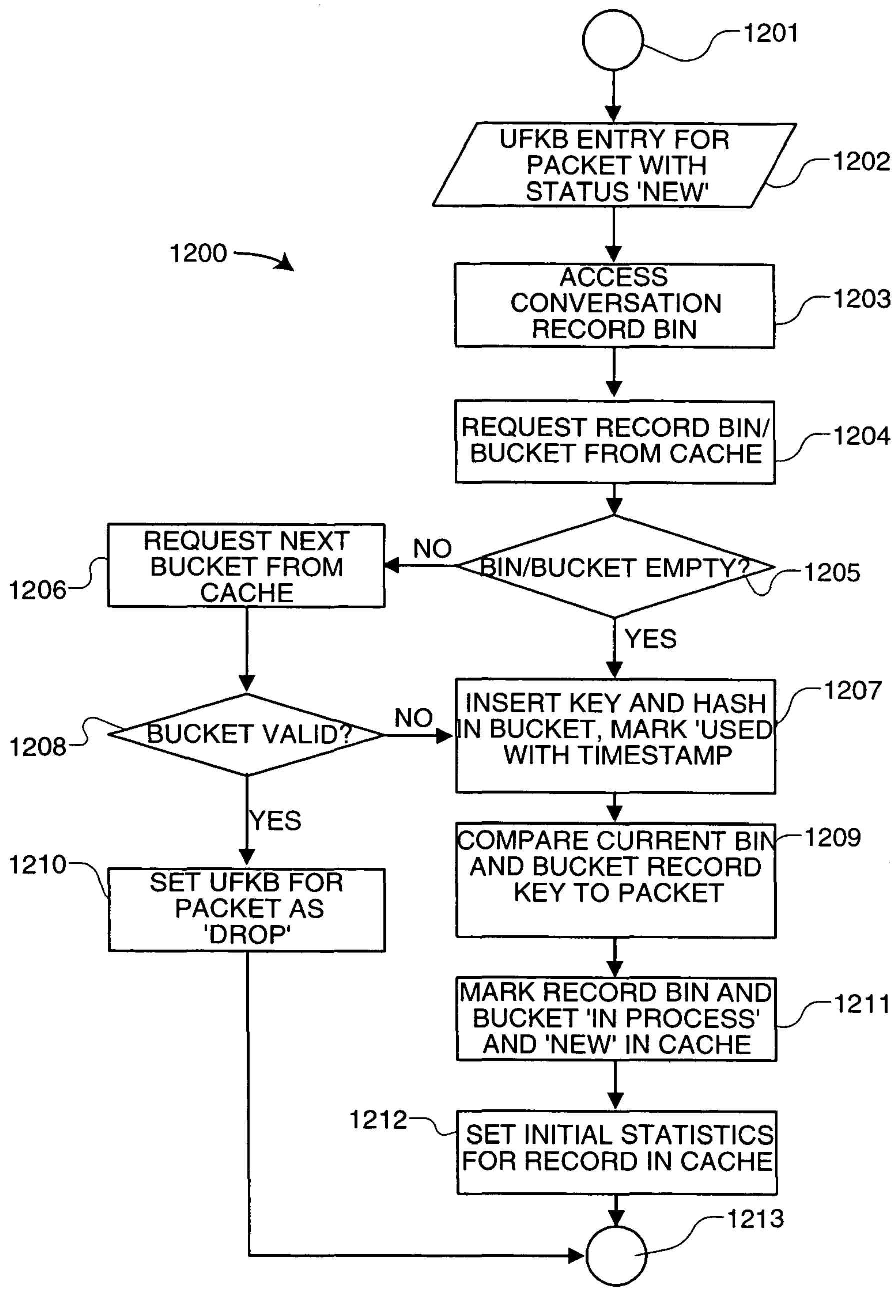

1201
UFKB ENTRY FOR PACKET WITH STATUS 'NEW'
1202
1200
ACCESS CONVERSATION RECORD BIN
1203
REQUEST RECORD BIN/ BUCKET FROM CACHE
1204
REQUEST NEXT BUCKET FROM CACHE
1206
NO
BIN/BUCKET EMPTY?
1205
YES
BUCKET VALID?
1208
NO
INSERT KEY AND HASH IN BUCKET, MARK 'USED' WITH TIMESTAMP
1207
YES
COMPARE CURRENT BIN AND BUCKET RECORD KEY TO PACKET
1209
1210
SET UFKB FOR PACKET AS 'DROP'
MARK RECORD BIN AND BUCKET 'IN PROCESS' AND 'NEW' IN CACHE
1211
1212
SET INITIAL STATISTICS FOR RECORD IN CACHE
1213

FIG. 12

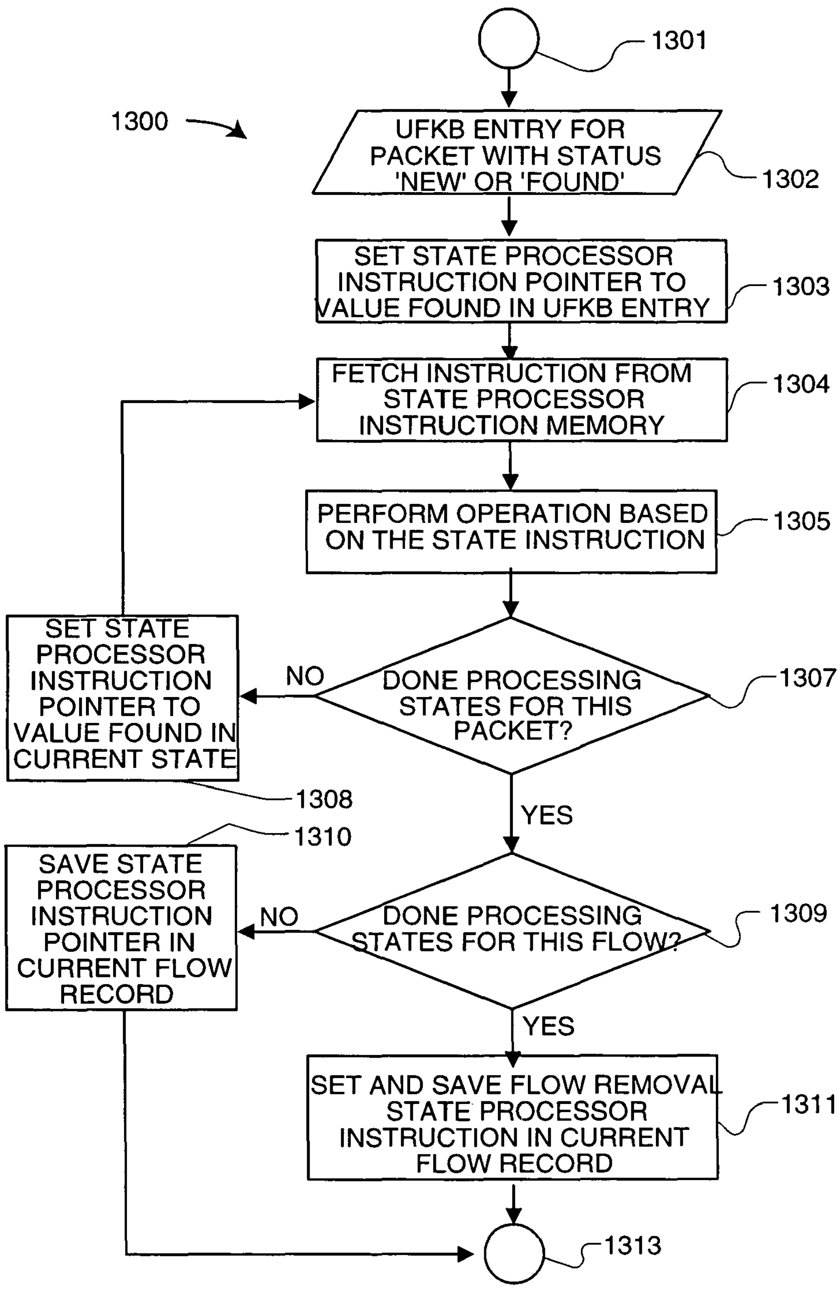

1301
1300
UFKB ENTRY FOR PACKET WITH STATUS 'NEW' OR 'FOUND'
1302
SET STATE PROCESSOR INSTRUCTION POINTER TO VALUE FOUND IN UFKB ENTRY
1303
FETCH INSTRUCTION FROM STATE PROCESSOR INSTRUCTION MEMORY
1304
PERFORM OPERATION BASED ON THE STATE INSTRUCTION
1305
DONE PROCESSING STATES FOR THIS PACKET?
1307
NO
SET STATE PROCESSOR INSTRUCTION POINTER TO VALUE FOUND IN CURRENT STATE
1308
YES
DONE PROCESSING STATES FOR THIS FLOW?
1309
NO
SAVE STATE PROCESSOR INSTRUCTION POINTER IN CURRENT FLOW RECORD
1310
YES
SET AND SAVE FLOW REMOVAL STATE PROCESSOR INSTRUCTION IN CURRENT FLOW RECORD
1311
1313

FIG. 13

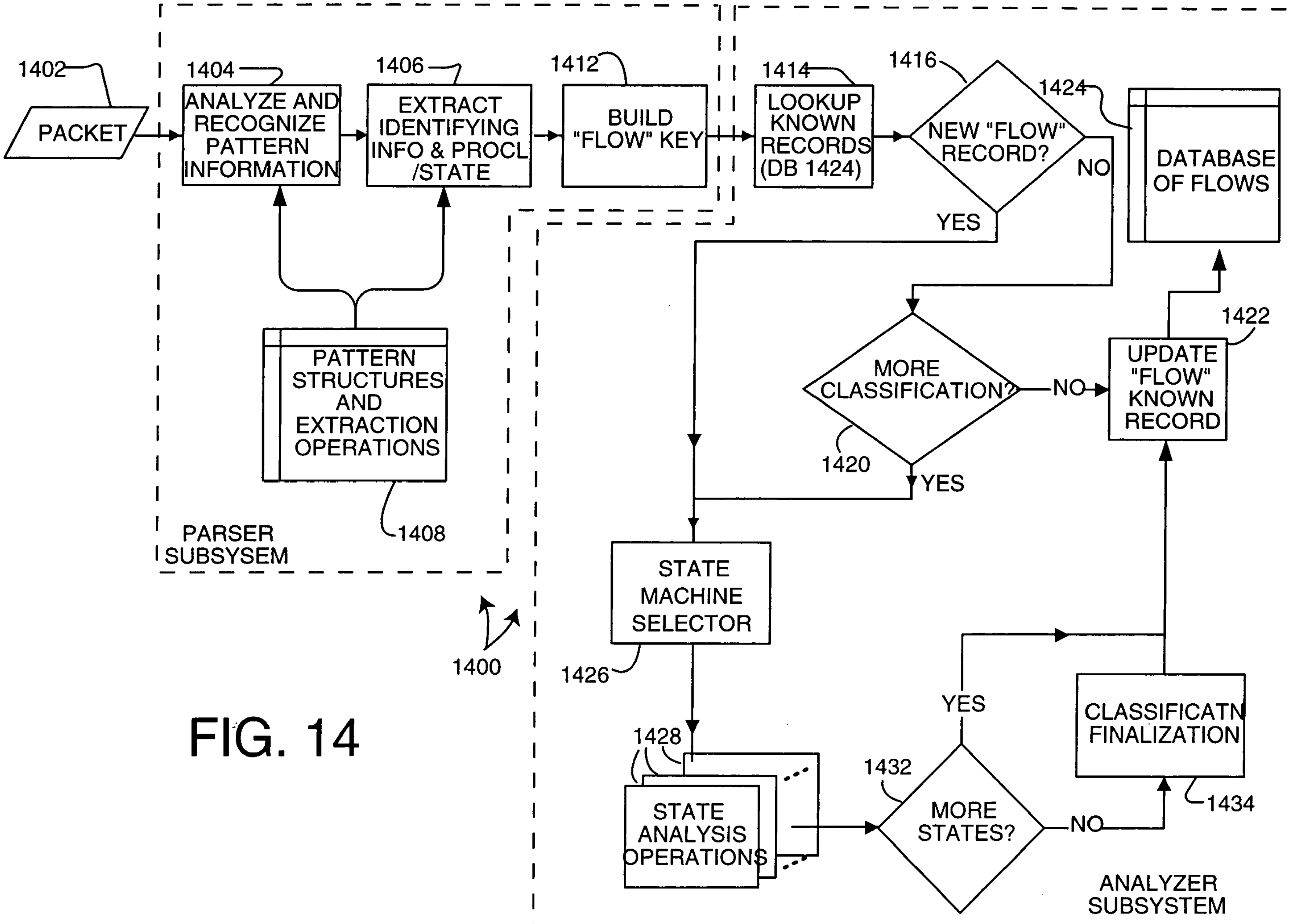

1402
PACKET
1404
ANALYZE AND RECOGNIZE PATTERN INFORMATION
1406
EXTRACT IDENTIFYING INFO & PROCL /STATE
1412
BUILD "FLOW" KEY
1414
LOOKUP KNOWN RECORDS (DB 1424)
1416
NEW "FLOW" RECORD?
NO
YES
1424
DATABASE OF FLOWS
PATTERN STRUCTURES AND EXTRACTION OPERATIONS
1408
PARSER SUBSYSEM
MORE CLASSIFICATION?
1420
NO
YES
1422
UPDATE "FLOW" KNOWN RECORD
STATE MACHINE SELECTOR
1426
1400
1428
STATE ANALYSIS OPERATIONS
1432
MORE STATES?
YES
NO
CLASSIFICATN FINALIZATION
1434
ANALYZER SUBSYSTEM

FIG. 14

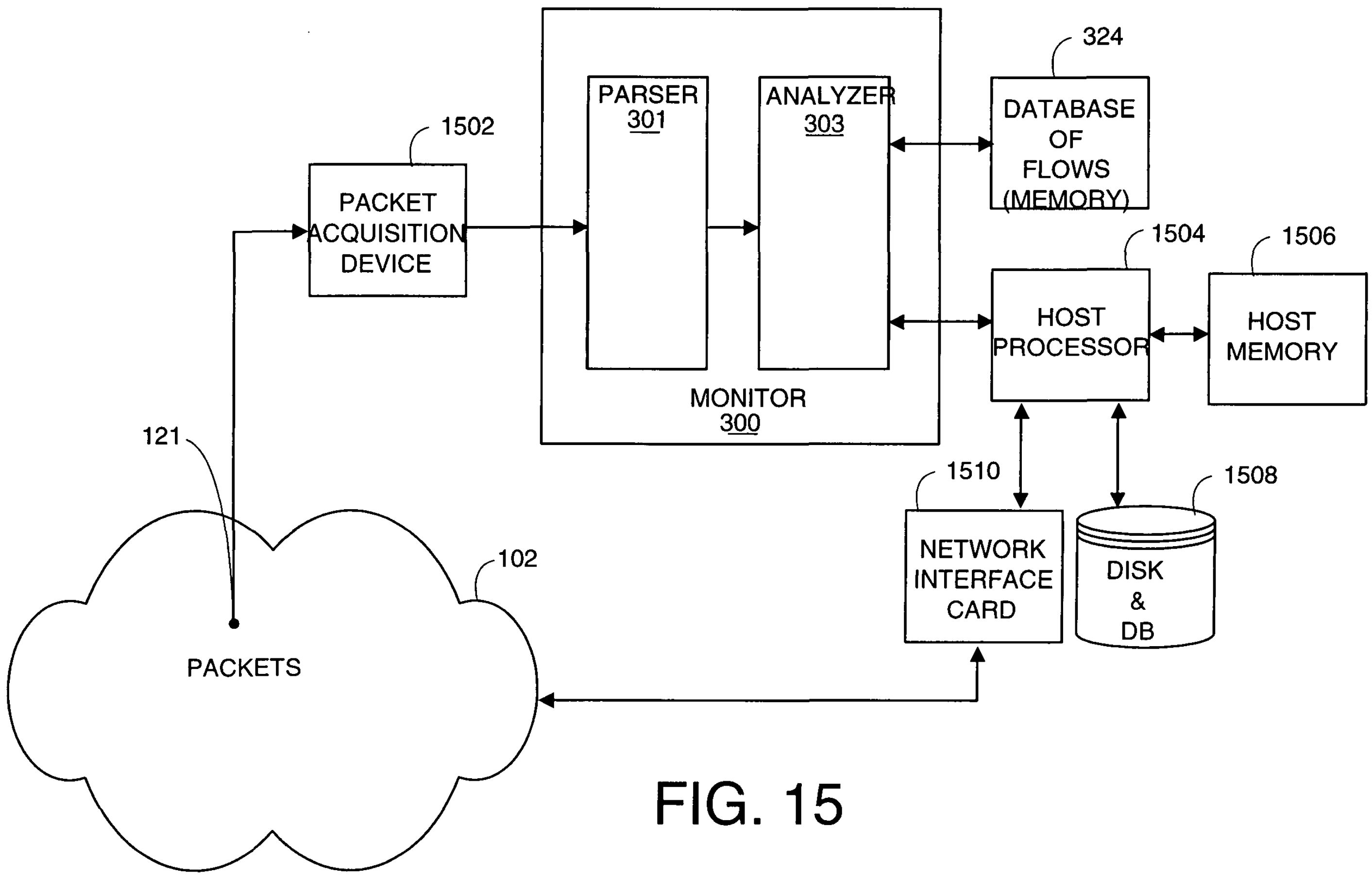
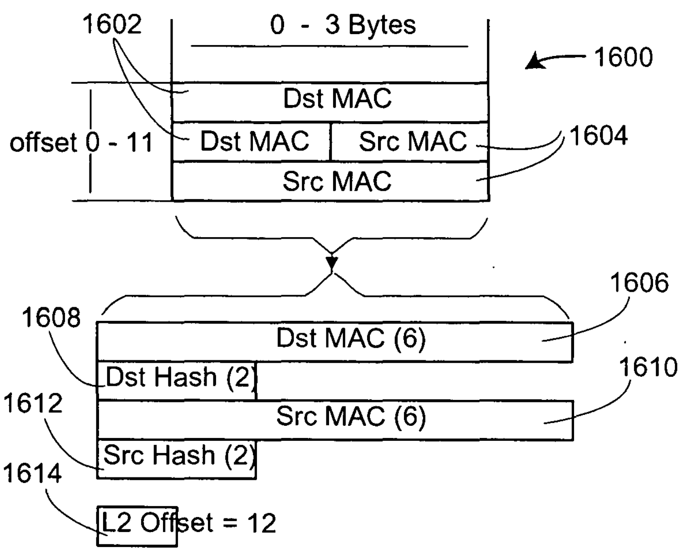

MONITOR 300
PARSER 301
ANALYZER 303
DATABASE OF FLOWS (MEMORY)
324
PACKET ACQUISITION DEVICE
1502
HOST PROCESSOR
1504
HOST MEMORY
1506
NETWORK INTERFACE CARD
1510
DISK & DB
1508
121
102
PACKETS

FIG. 15



0 - 3 Bytes
1602
1600
Dst MAC
offset 0 - 11
Dst MAC
Src MAC
1604
Src MAC
1606
1608
Dst MAC (6)
Dst Hash (2)
1610
1612
Src MAC (6)
Src Hash (2)
1614
L2 Offset = 12

FIG. 16

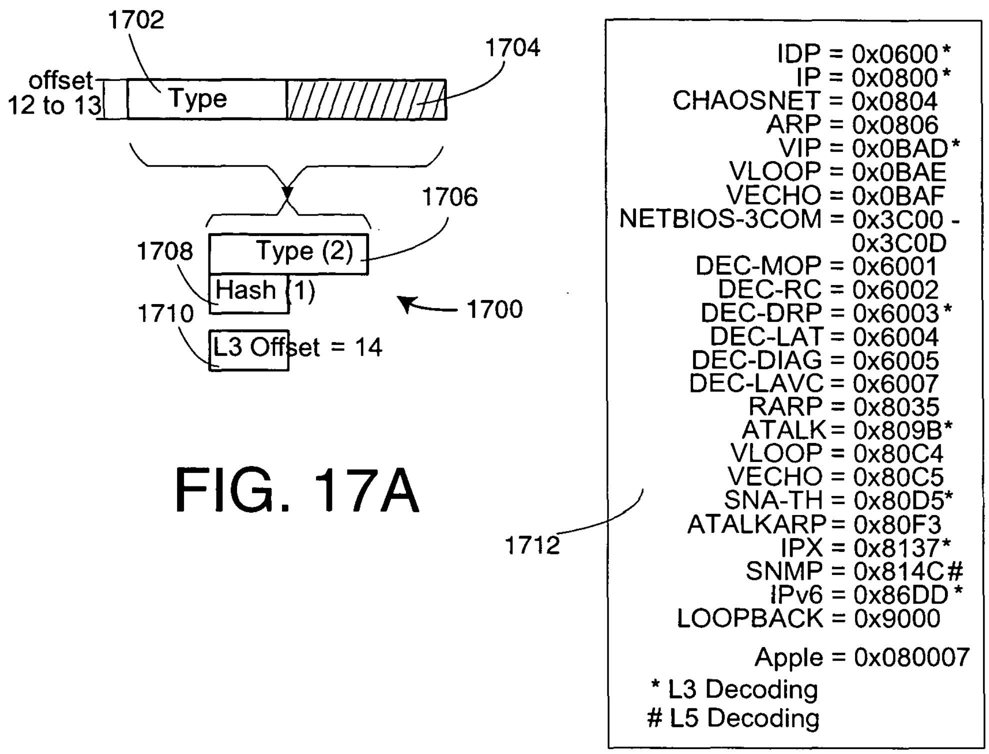


FIG. 17A

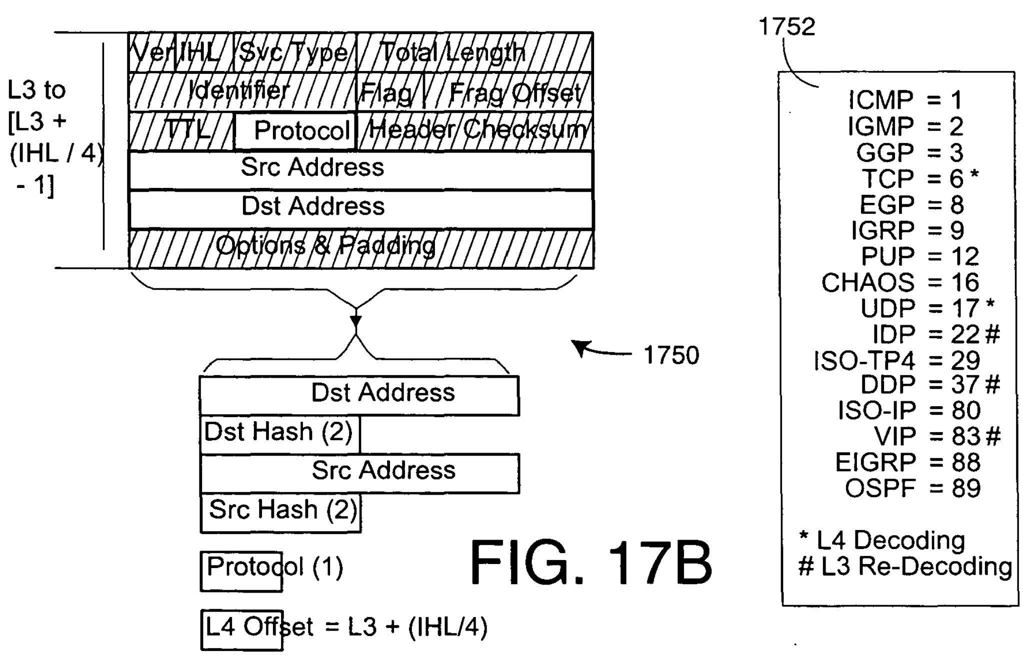


FIG. 17B

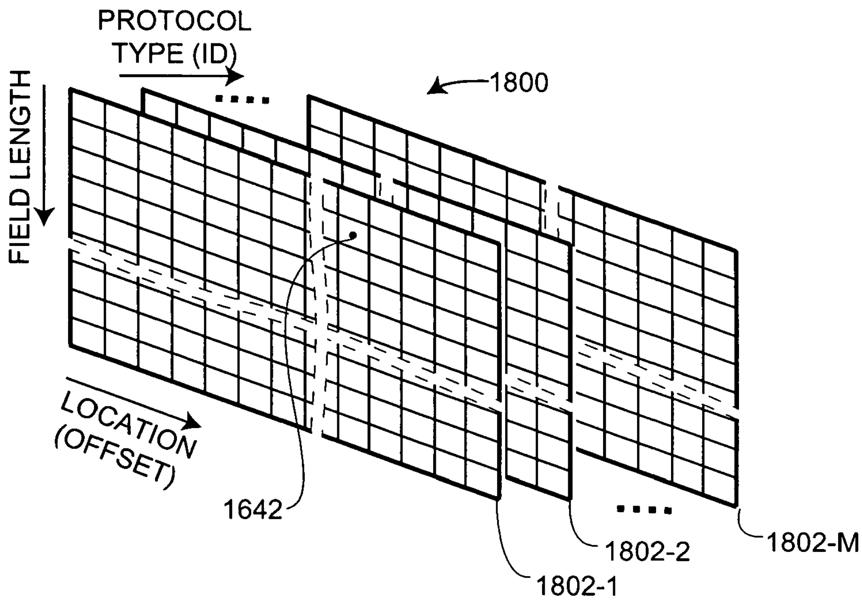


FIG. 18A

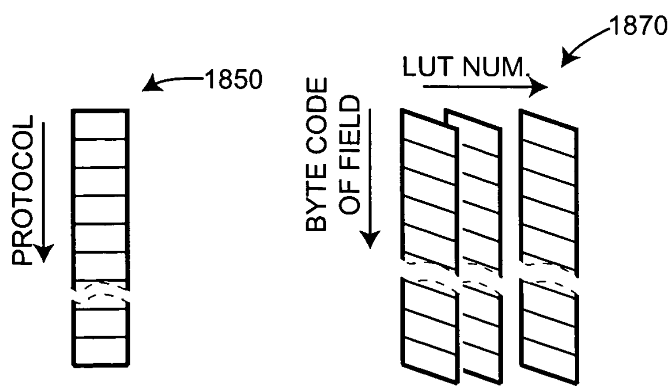


FIG. 18B

PATENT APPLICATION

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION	ATTORNEY DOCKET NO. APPT-001-1

As a below named inventor, I hereby declare that:

My residence/post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK

the specification of which is attached hereto unless the following box is checked:

(X) was filed on June 30, 2000 as US Application Serial No. 09/608237 or PCT International Application Number ____ and was amended on ________ (if applicable).

I hereby state that I have reviewed and understood the contents of the above-identified specification, including the claims, as amended by any amendment(s) referred to above. I acknowledge the duty to disclose all information which is material to patentability as defined in 37 CFR 1.56.

Foreign Application(s) and/or Claim of Foreign Priority

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor(s) certificate listed below and have also identified below any foreign application for patent or inventor(s) certificate having a filing date before that of the application on which priority is claimed:

COUNTRY	APPLICATION NUMBER	DATE FILED	PRIORITY CLAIMED UNDER 35
			YES: ____ NO: ____
			YES: ____ NO: ____

Provisional Application

I hereby claim the benefit under Title 35, United States Code Section 119(e) of any United States provisional application(s) listed below:

APPLICATION SERIAL NUMBER	FILING DATE
60/141,903	June 30, 1999

U.S. Priority Claim

I hereby claim the benefit under Title 35, United States Code, Section 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 the prior United States application in the manner provided by the first paragraph of Title 35, United States Code Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

APPLICATION SERIAL NUMBER	FILING DATE	STATUS(patented/pending/abandoned)

POWER OF ATTORNEY:

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) listed below to prosecute this application and transact all business in the Patent and Trademark Office connected therewith:

Dov Rosenfeld, Reg. No. 38,687

Send Correspondence to:	Direct Telephone Calls To:
Dov Rosenfeld 5507 College Avenue, Suite 2 Oakland, CA 94618	Dov Rosenfeld, Reg. No. 38,687 Tel: (510) 547-3378

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 Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Name of First Inventor: Russell S. Dietz **Citizenship:** USA

Residence: 6146 Ostenberg Drive, San Jose, CA 95120-2736

Post Office Address: Same

First Inventor's Signature

Date 10/3/00

ADDITIONAL INVENTOR SIGNATURES:

Name of Second Inventor: Joseph R. Maixner Citizenship: USA

Residence: 121 Driftwood Court, Aptos, CA 95003

Post Office Address: Same

Inventor's Signature Date

Name of Third Inventor: Andrew A. Koppenhaver Citizenship: USA

Residence: 10400 Kenmore Drive, Fairfax, VA 22030

Post Office Address: Same

Inventor's Signature Date

Name of Fourth Inventor: William H. Bares Citizenship: USA

Residence: 9005 Glenalden Drive, Germantown, TN 38139

Post Office Address: Same

Inventor's Signature Date

Name of Fifth Inventor: Haig A. Sarkissian Citizenship: USA

Residence: 8701 Mountain Top, San Antonio, Texas 78255

Post Office Address: Same

Inventor's Signature Date

Name of Sixth Inventor: James F. Torgerson Citizenship: USA

Residence: 227 157th Ave., NW, Andover, MN 55304

Post Office Address: Same

Inventor's Signature Date

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION — **ATTORNEY DOCKET NO. APPT-001-1**

As a below named inventor, I hereby declare that:

My residence/post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK

the specification of which is attached hereto unless the following box is checked:

(X) was filed on June 30, 2000 as US Application Serial No. 09/608237 or PCT International Application Number _____ and was amended on ________ (if applicable).

I hereby state that I have reviewed and understood the contents of the above-identified specification, including the claims, as amended by any amendment(s) referred to above. I acknowledge the duty to disclose all information which is material to patentability as defined in 37 CFR 1.56.

Foreign Application(s) and/or Claim of Foreign Priority

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor(s) certificate listed below and have also identified below any foreign application for patent or inventor(s) certificate having a filing date before that of the application on which priority is claimed:

COUNTRY	APPLICATION NUMBER	DATE FILED	PRIORITY CLAIMED UNDER 35
			YES: NO:
			YES: NO:

Provisional Application

I hereby claim the benefit under Title 35, United States Code Section 119(e) of any United States provisional application(s) listed below:

APPLICATION SERIAL NUMBER	FILING DATE
60/141,903	June 30, 1999

U.S. Priority Claim

I hereby claim the benefit under Title 35, United States Code, Section 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 the prior United States application in the manner provided by the first paragraph of Title 35, United States Code Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

APPLICATION SERIAL NUMBER	FILING DATE	STATUS(patented/pending/abandoned)

POWER OF ATTORNEY:

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) listed below to prosecute this application and transact all business in the Patent and Trademark Office connected therewith:

Dov Rosenfeld, Reg. No. 38,687

Send Correspondence to:	Direct Telephone Calls To:
Dov Rosenfeld	Dov Rosenfeld, Reg. No. 38,687
5507 College Avenue, Suite 2	Tel: (510) 547-3378
Oakland, CA 94618	

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 Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Name of First Inventor: Russell S. Dietz **Citizenship:** USA

Residence: 6146 Ostenberg Drive, San Jose, CA 95120-2736

Post Office Address: Same

First Inventor's Signature **Date**

ADDITIONAL INVENTOR SIGNATURES:

Name of Second Inventor: Joseph R. Maixner — Citizenship: USA

Residence: 121 Driftwood Court, Aptos, CA 95003

Post Office Address: Same

Inventor's Signature — Date 10/23/2000

Name of Third Inventor: Andrew A. Koppenhaver — Citizenship: USA

Residence: 10400 Kenmore Drive, Fairfax, VA 22030

Post Office Address: Same

Inventor's Signature — Date

Name of Fourth Inventor: William H. Bares — Citizenship: USA

Residence: 9005 Glenalden Drive, Germantown, TN 38139

Post Office Address: Same

Inventor's Signature — Date

Name of Fifth Inventor: Haig A. Sarkissian — Citizenship: USA

Residence: 8701 Mountain Top, San Antonio, Texas 78255

Post Office Address: Same

Inventor's Signature — Date

Name of Sixth Inventor: James F. Torgerson — Citizenship: USA

Residence: 227 157th Ave., NW, Andover, MN 55304

Post Office Address: Same

Inventor's Signature — Date

PATENT APPLICATION

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION — **ATTORNEY DOCKET NO. APPT-001-1**

As a below named inventor, I hereby declare that:

My residence/post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK

the specification of which is attached hereto unless the following box is checked:

(X) was filed on June 30, 2000 as US Application Serial No. 09/608237 or PCT International Application Number ____ and was amended on ________ (if applicable).

I hereby state that I have reviewed and understood the contents of the above-identified specification, including the claims, as amended by any amendment(s) referred to above. I acknowledge the duty to disclose all information which is material to patentability as defined in 37 CFR 1.56.

Foreign Application(s) and/or Claim of Foreign Priority

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor(s) certificate listed below and have also identified below any foreign application for patent or inventor(s) certificate having a filing date before that of the application on which priority is claimed:

COUNTRY	APPLICATION NUMBER	DATE FILED	PRIORITY CLAIMED UNDER 35
			YES: ____ NO: ____
			YES: ____ NO: ____

Provisional Application

I hereby claim the benefit under Title 35, United States Code Section 119(e) of any United States provisional application(s) listed below:

APPLICATION SERIAL NUMBER	FILING DATE
60/141,903	June 30, 1999

U.S. Priority Claim

I hereby claim the benefit under Title 35, United States Code, Section 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 the prior United States application in the manner provided by the first paragraph of Title 35, United States Code Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

APPLICATION SERIAL NUMBER	FILING DATE	STATUS(patented/pending/abandoned)

POWER OF ATTORNEY:

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) listed below to prosecute this application and transact all business in the Patent and Trademark Office connected therewith:

Dov Rosenfeld, Reg. No. 38,687

Send Correspondence to:	**Direct Telephone Calls To:**
Dov Rosenfeld 5507 College Avenue, Suite 2 Oakland, CA 94618	Dov Rosenfeld, Reg. No. 38,687 Tel: (510) 547-3378

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 Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Name of First Inventor: Russell S. Dietz **Citizenship:** USA

Residence: 6146 Ostenberg Drive, San Jose, CA 95120-2736

Post Office Address: Same

________________________ ________________________

First Inventor's Signature **Date**

ADDITIONAL INVENTOR SIGNATURES:

Name of Second Inventor: Joseph R. Maixner **Citizenship:** USA

Residence: 121 Driftwood Court, Aptos, CA 95003

Post Office Address: Same

Inventor's Signature **Date**

Name of Third Inventor: Andrew A. Koppenhaver **Citizenship:** USA

Residence: 9325 W. Hinsdale Place, Littleton, CO 80128

Post Office Address: Same

Inventor's Signature **Date** 10/10/2000

Name of Fourth Inventor: William H. Bares **Citizenship:** USA

Residence: 9005 Glenalden Drive, Germantown, TN 38139

Post Office Address: Same

Inventor's Signature **Date**

Name of Fifth Inventor: Haig A. Sarkissian **Citizenship:** USA

Residence: 8701 Mountain Top, San Antonio, Texas 78255

Post Office Address: Same

Inventor's Signature **Date**

Name of Sixth Inventor: James F. Torgerson **Citizenship:** USA

Residence: 227 157th Ave., NW, Andover, MN 55304

Post Office Address: Same

Inventor's Signature **Date**

PATENT APPLICATION

DECLARATION AND POWER OF ATT NEY FOR PATENT APPLICATION	ATT NEY DOCKET NO. APPT-001-1

As a below named inventor, I hereby declare that:

My residence/post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK

the specification of which is attached hereto unless the following box is checked:

(X) was filed on June 30, 2000 as US Application Serial No. 09/608237 or PCT International Application Number ____ and was amended on ________ (if applicable).

I hereby state that I have reviewed and understood the contents of the above-identified specification, including the claims, as amended by any amendment(s) referred to above. I acknowledge the duty to disclose all information which is material to patentability as defined in 37 CFR 1.56.

Foreign Application(s) and/or Claim of Foreign Priority

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor(s) certificate listed below and have also identified below any foreign application for patent or inventor(s) certificate having a filing date before that of the application on which priority is claimed:

COUNTRY	APPLICATION NUMBER	DATE FILED	PRIORITY CLAIMED UNDER 35
			YES: NO:
			YES: NO:

Provisional Application

I hereby claim the benefit under Title 35, United States Code Section 119(e) of any United States provisional application(s) listed below:

APPLICATION SERIAL NUMBER	FILING DATE
60/141,903	June 30, 1999

U.S. Priority Claim

I hereby claim the benefit under Title 35, United States Code, Section 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 the prior United States application in the manner provided by the first paragraph of Title 35, United States Code Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

APPLICATION SERIAL NUMBER	FILING DATE	STATUS(patented/pending/abandoned)

POWER OF ATTORNEY:

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) listed below to prosecute this application and transact all business in the Patent and Trademark Office connected therewith:

Dov Rosenfeld, Reg. No. 38,687

Send Correspondence to:	**Direct Telephone Calls To:**
Dov Rosenfeld 5507 College Avenue, Suite 2 Oakland, CA 94618	Dov Rosenfeld, Reg. No. 38,687 Tel: (510) 547-3378

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 Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Name of First Inventor: Russell S. Dietz **Citizenship:** USA

Residence: 6146 Ostenberg Drive, San Jose, CA 95120-2736

Post Office Address: Same

________________________ ________________

First Inventor's Signature **Date**

ADDITIONAL INVENTOR SIGNATURES:

Name of Second Inventor: Joseph R. Maixner **Citizenship:** USA

Residence: 121 Driftwood Court, Aptos, CA 95003

Post Office Address: Same

Inventor's Signature **Date**

Name of Third Inventor: Andrew A. Koppenhaver **Citizenship:** USA

Residence: 10400 Kenmore Drive, Fairfax, VA 22030

Post Office Address: Same

Inventor's Signature **Date**

Name of Fourth Inventor: William H. Bares **Citizenship:** USA

Residence: 9005 Glenalden Drive, Germantown, TN 38139

Post Office Address: Same

William H Bares 10/8/00

Inventor's Signature **Date**

Name of Fifth Inventor: Haig A. Sarkissian **Citizenship:** USA

Residence: 8701 Mountain Top, San Antonio, Texas 78255

Post Office Address: Same

Inventor's Signature **Date**

Name of Sixth Inventor: James F. Torgerson **Citizenship:** USA

Residence: 227 157th Ave., NW, Andover, MN 55304

Post Office Address: Same

Inventor's Signature **Date**

PATENT APPLICATION

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION	ATTORNEY DOCKET NO. APPT-001-1

As a below named inventor, I hereby declare that:

My residence/post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK

the specification of which is attached hereto unless the following box is checked:

(X) was filed on June 30, 2000 as US Application Serial No. 09/608237 or PCT International Application Number ____ and was amended on ________ (if applicable).

I hereby state that I have reviewed and understood the contents of the above-identified specification, including the claims, as amended by any amendment(s) referred to above. I acknowledge the duty to disclose all information which is material to patentability as defined in 37 CFR 1.56.

Foreign Application(s) and/or Claim of Foreign Priority

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor(s) certificate listed below and have also identified below any foreign application for patent or inventor(s) certificate having a filing date before that of the application on which priority is claimed:

COUNTRY	APPLICATION NUMBER	DATE FILED	PRIORITY CLAIMED UNDER 35
			YES: ____ NO: ____
			YES: ____ NO: ____

Provisional Application

I hereby claim the benefit under Title 35, United States Code Section 119(e) of any United States provisional application(s) listed below:

APPLICATION SERIAL NUMBER	FILING DATE
60/141,903	June 30, 1999

U.S. Priority Claim

I hereby claim the benefit under Title 35, United States Code, Section 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 the prior United States application in the manner provided by the first paragraph of Title 35, United States Code Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

APPLICATION SERIAL NUMBER	FILING DATE	STATUS(patented/pending/abandoned)

POWER OF ATTORNEY:

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) listed below to prosecute this application and transact all business in the Patent and Trademark Office connected therewith:

Dov Rosenfeld, Reg. No. 38,687

Send Correspondence to:	Direct Telephone Calls To:
Dov Rosenfeld 5507 College Avenue, Suite 2 Oakland, CA 94618	Dov Rosenfeld, Reg. No. 38,687 Tel: (510) 547-3378

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 Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Name of First Inventor: Russell S. Dietz **Citizenship:** USA

Residence: 6146 Ostenberg Drive, San Jose, CA 95120-2736

Post Office Address: Same

________________________ ________________________

First Inventor's Signature **Date**

ADDITIONAL INVENTOR SIGNATURES:

Name of Second Inventor: Joseph R. Maixner **Citizenship:** USA

Residence: 121 Driftwood Court, Aptos, CA 95003

Post Office Address: Same

Inventor's Signature **Date**

Name of Third Inventor: Andrew A. Koppenhaver **Citizenship:** USA

Residence: 10400 Kenmore Drive, Fairfax, VA 22030

Post Office Address: Same

Inventor's Signature **Date**

Name of Fourth Inventor: William H. Bares **Citizenship:** USA

Residence: 9005 Glenalden Drive, Germantown, TN 38139

Post Office Address: Same

Inventor's Signature **Date**

Name of Fifth Inventor: Haig A. Sarkissian **Citizenship:** USA

Residence: 8701 Mountain Top, San Antonio, Texas 78255

Post Office Address: Same

Haig A. Sarkissian Sept. 21, 2000

Inventor's Signature **Date**

Name of Sixth Inventor: James F. Torgerson **Citizenship:** USA

Residence: 227 157th Ave., NW, Andover, MN 55304

Post Office Address: Same

Inventor's Signature **Date**

PATENT APPLICATION

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION	**ATTORNEY DOCKET NO: APPT-001-1**

As a below named inventor, I hereby declare that:

My residence/post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK

the specification of which is attached hereto unless the following box is checked:

(X) was filed on June 30, 2000 as US Application Serial No. 09/608237 or PCT International Application Number ____ and was amended on ________ (if applicable).

I hereby state that I have reviewed and understood the contents of the above-identified specification, including the claims, as amended by any amendment(s) referred to above. I acknowledge the duty to disclose all information which is material to patentability as defined in 37 CFR 1.56.

Foreign Application(s) and/or Claim of Foreign Priority

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor(s) certificate listed below and have also identified below any foreign application for patent or inventor(s) certificate having a filing date before that of the application on which priority is claimed:

COUNTRY	APPLICATION NUMBER	DATE FILED	PRIORITY CLAIMED UNDER 35
			YES: ____ NO: ____
			YES: ____ NO: ____

Provisional Application

I hereby claim the benefit under Title 35, United States Code Section 119(e) of any United States provisional application(s) listed below:

APPLICATION SERIAL NUMBER	FILING DATE
60/141,903	June 30, 1999

U.S. Priority Claim

I hereby claim the benefit under Title 35, United States Code, Section 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 the prior United States application in the manner provided by the first paragraph of Title 35, United States Code Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

APPLICATION SERIAL NUMBER	FILING DATE	STATUS(patented/pending/abandoned)

POWER OF ATTORNEY:

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) listed below to prosecute this application and transact all business in the Patent and Trademark Office connected therewith:

Dov Rosenfeld, Reg. No. 38,687

Send Correspondence to:	**Direct Telephone Calls To:**
Dov Rosenfeld	Dov Rosenfeld, Reg. No. 38,687
5507 College Avenue, Suite 2	Tel: (510) 547-3378
Oakland, CA 94618	

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 Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Name of First Inventor: Russell S. Dietz **Citizenship:** USA

Residence: 6146 Ostenberg Drive, San Jose, CA 95120-2736

Post Office Address: Same

______________________ ______________________

First Inventor's Signature **Date**

Declaration and Power of Attorney (Continued)
Case No; «Case CaseNumber» APPT-001-1
Page 2

ADDITIONAL INVENTOR SIGNATURES:

Name of Second Inventor: Joseph R. Maixner Citizenship: USA

Residence: 121 Driftwood Court, Aptos, CA 95003

Post Office Address: Same

Inventor's Signature Date

Name of Third Inventor: Andrew A. Koppenhaver Citizenship: USA

Residence: 10400 Kenmore Drive, Fairfax, VA 22030

Post Office Address: Same

Inventor's Signature Date

Name of Fourth Inventor: William H. Bares Citizenship: USA

Residence: 9005 Glenalden Drive, Germantown, TN 38139

Post Office Address: Same

Inventor's Signature Date

Name of Fifth Inventor: Haig A. Sarkissian Citizenship: USA

Residence: 8701 Mountain Top, San Antonio, Texas 78255

Post Office Address: Same

Inventor's Signature Date

Name of Sixth Inventor: James F. Torgerson Citizenship: USA

Residence: 227 157th Ave., NW, Andover, MN 55304

Post Office Address: Same

James Torgerson 9/21/00

Inventor's Signature Date

Our Ref./Docket No: APPT-001-1-1 Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): Dietz, *et al.* Application No.: Filed: Title: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK	Group Art Unit: 2155 Examiner: Khanh Q. DINH

PRELIMINARY AMENDMENT

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Commissioner:

This is a preliminary amendment prior to any Office Action.

Any ***amendments to the specification*** begin on a new page immediately after these introductory remarks.

Any ***amendments to the claims*** begin on a new page immediately after such ***amendments to the specification***, if any.

Any ***amendments to the drawings*** begin on a new page immediately after such ***amendments to the claims***, if any.

The ***Remarks/arguments*** begin on a new page immediately after such ***amendments to the drawings***, if any.

If there are drawing amendments, an ***Appendix*** including amended drawings is attached following the ***Remarks/arguments***.

Certificate of Mailing under 37 CFR 1.10

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on.

Date: Oct ~~13~~ 14, 2003 Signed: [signature]
Name: Dov Rosenfeld, Reg. No. 38687

AMENDMENT(S) TO THE CLAIMS:

The following listing of claims will replace all prior versions, and listings, of claims on the application. All claims are set forth below with one of the following annotations.

- (Original): Claim filed with the application following the specification.
- (Currently amended): Claim being amended in the current amendment paper.
- (Cancelled): Claim cancelled or deleted from the application.
- (Withdrawn): Claim still in the application, but in a non-elected status.
- (New): Claim being added in the current amendment paper.
- (Previously presented): Claim not being currently amended, but which was amended or was new in a previous amendment paper.
- (Not entered): Claim presented in a previous amendment, but not entered or whose entry status unknown. No claim text is shown.

1.–10. (Cancelled).

11. (Original) A method of examining packets passing through a connection point on a computer network, each packets conforming to one or more protocols, the method comprising:

    (a) receiving a packet from a packet acquisition device;

    (b) performing one or more parsing/extraction operations on the packet to create a parser record comprising a function of selected portions of the packet;

    (c) looking up a flow-entry database comprising none or more flow-entries for previously encountered conversational flows, the looking up using at least some of the selected packet portions and determining if the packet is of an existing flow;

    (d) if the packet is of an existing flow, classifying the packet as belonging to the found existing flow; and

    (e) if the packet is of a new flow, storing a new flow-entry for the new flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry,

    wherein the parsing/extraction operations depend on one or more of the protocols to which the packet conforms.

12. (Original) A method according to claim 11, wherein each packet passing through the connection point is examined in real time.

13. (Original) A method according to claim 11, wherein classifying the packet as belonging to the found existing flow includes updating the flow-entry of the existing flow.

14. (Original) A method according to claim 13, wherein updating includes storing one or more statistical measures stored in the flow-entry of the existing flow.

15. (Original) A method according to claim 14, wherein the one or more statistical measures include measures selected from the set consisting of the total packet count for the flow, the time, and a differential time from the last entered time to the present time.

16. (Original) A method according to claim 11, wherein the function of the selected portions of the packet forms a signature that includes the selected packet portions and that can identify future packers, wherein the lookup operation uses the signature and wherein the identifying information stored in the new or updated flow-entry is a signature for identifying future packets.

17. (Original) A method according to claim 11, wherein at least one of the protocols of the packet uses source and destination addresses, and wherein the selected portions of the packet include the source and destination addresses.

18. (Original) A method according to claim 17, wherein the function of the selected portions for packets of the same flow is consistent independent of the direction of the packets.

19. (Original) A method according to claim 18, wherein the source and destination addresses are placed in an order determined by the order of numerical values of the addresses in the function of selected portions.

20. (Original) A method according to claim 19, wherein the numerically lower address is placed before the numerically higher address in the function of selected portions.

21. (Original) A method according to claim 11, wherein the looking up of the flow-entry database uses a hash of the selected packet portions.

22. (Original) A method according to claim 11, wherein the parsing/extraction operations are according to a database of parsing/extraction operations that includes information describing how to determine a set of one or more protocol dependent extraction operations from data in the packet that indicate a protocol used in the packet.

23. (Original) A method according to claim 11, wherein step (d) includes if the packet is of an existing flow, obtaining the last encountered state of the flow and performing any state operations specified for the state of the flow starting from the last encountered state of the flow; and wherein step (e) includes if the packet is of a new flow, performing any state operations required for the initial state of the new flow.

24. (Original) A method according to claim 23, wherein the state processing of each received packet of a flow furthers the identifying of the application program of the flow.

25. (Original) A method according to claim 23, wherein the state operations include updating the flow-entry, including storing identifying information for future packets to be identified with the flow-entry.

26. (Original) A method according to claim 25, wherein the state processing of each received packet of a flow furthers the identifying of the application program of the flow.

27. (Original) A method according to claim 23, wherein the state operations include searching the parser record for the existence of one or more reference strings.

28. (Original) A method according to claim 23, wherein the state operations are carried out by a programmable state processor according to a database of protocol dependent state operations.

29. (Original) A packet monitor for examining packets passing through a connection point on a computer network, each packets conforming to one or more protocols, the monitor comprising:

(a) a packet acquisition device coupled to the connection point and configured to receive packets passing through the connection point;

(b) an input buffer memory coupled to and configured to accept a packet from the packet acquisition device;

(c) a parser subsystem coupled to the input buffer memory and including a slicer, the parsing subsystem configured to extract selected portions of the accepted packet and to output a parser record containing the selected portions;

(d) a memory for storing a database comprising none or more flow-entries for previously encountered conversational flows, each flow-entry identified by identifying information stored in the flow-entry;

(e) a lookup engine coupled to the output of the parser subsystem and to the flow-entry memory and configured to lookup whether the particular packet whose parser record is output by the parser subsystem has a matching flow-entry, the looking up using at least some of the selected packet portions and determining if the packet is of an existing flow; and

(f) a flow insertion engine coupled to the flow-entry memory and to the lookup engine and configured to create a flow-entry in the flow-entry database, the flow-entry including identifying information for future packets to be identified with the new flow-entry, the lookup engine configured such that if the packet is of an existing flow, the monitor classifies the packet as belonging to the found existing flow; and if the packet is of a new flow, the flow insertion engine stores a new flow-entry for the new flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry,

wherein the operation of the parser subsystem depends on one or more of the protocols to which the packet conforms.

30. (Original) A monitor according to claim 29, wherein each packet passing through the connection point is accepted by the packet buffer memory and examined by the monitor in real time.

31. (Original) A monitor according to claim 29, wherein the lookup engine updates the flow-entry of an existing flow in the case that the lookup is successful.

32. (Original) A monitor according to claim 29, further including a mechanism for building a hash from the selected portions, wherein the hash is included in the input for a particular packet to the lookup engine, and wherein the hash is used by the lookup engine to search the flow-entry database.

33. (Original) A monitor according to claim 29, further including a memory containing a database of parsing/extraction operations, the parsing/extraction database memory coupled to the parser subsystem, wherein the parsing/extraction operations are according to one or more parsing/extraction operations looked up from the parsing/extraction database.

34. (Original) A monitor according to claim 33, wherein the database of parsing/extraction operations includes information describing how to determine a set of one or more protocol dependent extraction operations from data in the packet that indicate a protocol used in the packet.

35. (Original) A monitor according to claim 29, further including a flow-key-buffer (UFKB) coupled to the output of the parser subsystem and to the lookup engine and to the flow insertion engine, wherein the output of the parser monitor is coupled to the lookup engine via the UFKB, and wherein the flow insertion engine is coupled to the lookup engine via the UFKB.

36. (Original) A method according to claim 29, further including a state processor coupled to the lookup engine and to the flow-entry-database memory, and configured to perform any state operations specified for the state of the flow starting from the last encountered state of the flow in the case that the packet is from an existing flow, and to perform any state operations required for the initial state of the new flow in the case that the packet is from an existing flow.

37. (Original) A method according to claim 29, wherein the set of possible state operations that the state processor is configured to perform includes searching for one or more patterns in the packet portions.

38. (Original) A monitor according to claim 36, wherein the state processor is programmable, the monitor further including a state patterns/operations memory coupled to the state processor, the state operations memory configured to store a database of protocol dependent state patterns/operations.

39. (Original) A monitor according to claim 35, further including a state processor coupled to the UFKB and to the flow-entry-database memory, and configured to perform any state operations specified for the state of the flow starting from the last encountered state of the flow in the case that the packet is from an existing flow, and to perform any state operations required for the initial state of the new flow in the case that the packet is from an existing flow.

40. (Original) A monitor according to claim 36, wherein the state operations include updating the flow-entry, including identifying information for future packets to be identified with the flow-entry.

41. (Original) A packet monitor according to claim 29, further comprising:

a compiler processor coupled to the parsing/extraction operations memory, the compiler processor configured to run a compilation process that includes:

receiving commands in a high-level protocol description language that describe the protocols that may be used in packets encountered by the monitor and any children protocols thereof, and

translating the protocol description language commands into a plurality of parsing/extraction operations that are initialized into the parsing/extraction operations memory.

42. (Original) A packet monitor according to claim 38, further comprising:

a compiler processor coupled to the parsing/extraction operations memory, the compiler processor configured to run a compilation process that includes:

receiving commands in a high-level protocol description language that describe a correspondence between a set of one or more application programs and the state transition patterns/operations that occur as a result of particular conversational flow-sequences associated with an application programs, and

translating the protocol description language commands into a plurality of state patterns and state operations that are initialized into the state patterns/operations memory.

43. (Original) A packet monitor according to claim 29, further comprising:

a cache subsystem coupled to and between the lookup engine and the flow-entry database memory providing for fast access of a set of likely-to-be-accessed flow-entries from the flow-entry database.

44. (Original) A packet monitor according to claim 43, wherein the cache subsystem is an associative cache subsystem including one or more content addressable memory cells (CAMs).

45. (Original) A packet monitor according to claim 44, wherein the cache subsystem is also a least-recently-used cache memory such that a cache miss updates the least recently used cache entry.

46. (Original) A packet monitor according to claim 29, wherein each flow-entry stores one or more statistical measures about the flow, the monitor further comprising

a calculator for updating at least one of the statistical measures in the flow-entry of the accepted packet.

47. (Original) A packet monitor according to claim 46, wherein the one or more statistical measures include measures selected from the set consisting of the total packet count for the flow, the time, and a differential time from the last entered time to the present time.

48. (Original) A packet monitor according to claim 46, further including a statistical processor configured to determine one or more network usage metrics related to the flow from one or more of the statistical measures in a flow-entry.

49. (Original) A monitor according to claim 29, wherein:

flow-entry-database is organized into a plurality of bins that each contain N-number of flow-entries, and wherein said bins are accessed via a hash data value created by a parser subsystem based on the selected packet portions, wherein N is one or more.

50. (Original) A monitor according to claim 49, wherein the hash data value is used to spread a plurality of flow-entries across the flow-entry-database and allows fast lookup of a flow-entry and shallower buckets.

51. (Original) A monitor according to claim 36, wherein the state processor analyzes both new and existing flows in order to classify them by application and proceeds from state-to-state based on a set of predefined rules.

52. (Original) A monitor according to claim 29, wherein the lookup engine begins processing as soon as a parser record arrives from the parser subsystem.

53. (Original) A monitor according to claim 36, wherein the lookup engine provides for flow state entry checking to see if a flow key should be sent to the state processor, and that outputs a protocol identifier for the flow.

54. (Original) A method of examining packets passing through a connection point on a computer network, the method comprising:

(a) receiving a packet from a packet acquisition device;

(b) performing one or more parsing/extraction operations on the packet according to a database of parsing/extraction operations to create a parser record comprising a function of selected portions of the packet, the database of parsing/extraction operations including information on how to determine a set of one or more protocol dependent extraction operations from data in the packet that indicate a protocol is used in the packet;

(c) looking up a flow-entry database comprising none or more flow-entries for previously encountered conversational flows, the looking up using at least some of the selected packet portions, and determining if the packet is of an existing flow;

(d) if the packet is of an existing flow, obtaining the last encountered state of the flow and performing any state operations specified for the state of the flow starting from the last encountered state of the flow; and

(e) if the packet is of a new flow, performing any analysis required for the initial state of the new flow and storing a new flow-entry for the new flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry.

55. (Original) A method according to claim 54, wherein one of the state operations specified for at least one of the states includes updating the flow-entry, including identifying information for future packets to be identified with the flow-entry.

56. (Original) A method according to claim 54, wherein one of the state operations specified for at least one of the states includes searching the contents of the packet for at least one reference string.

57. (Original) A method according to claim 55, wherein one of the state operations specified for at least one of the states includes creating a new flow-entry for future packets to be identified with the flow, the new flow-entry including identifying information for future packets to be identified with the flow-entry.

58. (Original) A method according to claim 54, further comprising forming a signature from the selected packet portions, wherein the lookup operation uses the signature and wherein the identifying information stored in the new or updated flow-entry is a signature for identifying future packets.

59. (Original) A method according to claim 54, wherein the state operations are according to a database of protocol dependent state operations.

REMARKS

This is a continuation of U.S. Patent Application 09/608237. Claims 1-59 are the claims as filed. Claims 1-10 are the allowed claims of the parent Application No. 09/608237, and are being cancelled by this preliminary amendment. Claims 11-59 remain the claims of record after this amendment. Examination thereof is respectfully requested.

If the Examiner has any questions or comments that would advance the prosecution and allowance of this application, an email message to the undersigned at dov@inventek.com, or a telephone call to the undersigned at +1-510-547-3378 is requested.

Respectfully Submitted,

Oct 13, 2003
Date

Dov Rosenfeld, Reg. No. 38687

Address for correspondence:
Dov Rosenfeld
5507 College Avenue, Suite 2
Oakland, CA 94618
Tel. +1-510-547-3378
Fax: +1-510-291-2985
Email:dov@inventek.com.

~~MULTIPLE DEPENDENT~~ CLAIM FEE CALCULATION SHEET	SERIAL NO. 10684776	FILING DATE 10-14-03
	APPLICANT(S)	

CLAIMS

	AS FILED		AFTER 1ST AMENDMENT		AFTER 2ND AMENDMENT			*		*		*	
	IND	DEP	IND	DEP	IND	DEP		IND	DEP	IND	DEP	IND	DEP
1							51		|				
2							52		|				
3							53		|				
4							54	|					
5							55		|				
6							56		|				
7							57		|				
8							58		|				
9							59		|				
10							60						
11	|						61						
12		|					62						
13		|					63						
14		|					64						
15		|					65						
16		|					66						
17		|					67						
18		|					68						
19		|					69						
20		|					70						
21		|					71						
22		|					72						
23		|					73						
24		|					74						
25		|					75						
26		|					76						
27		|					77						
28		|					78						
29	|						79						
30		|					80						
31		|					81						
32		|					82						
33		|					83						
34		|					84						
35		|					85						
36		|					86						
37		|					87						
38		|					88						
39		|					89						
40		|					90						
41		|					91						
42		|					92						
43		|					93						
44		|					94						
45		|					95						
46		|					96						
47		|					97						
48		|					98						
49		|					99						
50		|					100						
TOTAL IND.							TOTAL IND.	3					
TOTAL DEP.							TOTAL DEP.	46					
TOTAL CLAIMS							TOTAL CLAIMS	49					

PTO-1360 (12/02)

*MAY BE USED FOR ADDITIONAL CLAIMS OR AMENDMENTS

U.S. DEPARTMENT of COMMERCE
Patent and Trademark Office

PATENT APPLICATION SERIAL NO. ____________________

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

10/20/2003 YPOLITE1 00000080 10684776

01 FC:1001 770.00 OP
02 FC:1202 522.00 OP

PTO-1556
(5/87)

*U.S. Government Printing Office: 2002 — 489-267/69033

PATENT APPLICATION FEE DETERMINATION RECORD
Effective October 1, 2003

Application or Docket Number
APPT-001-1-1

CLAIMS AS FILED - PART I

	(Column 1)	(Column 2)
TOTAL CLAIMS	49	
FOR	NUMBER FILED	NUMBER EXTRA
TOTAL CHARGEABLE CLAIMS	49 minus 20=	* 29
INDEPENDENT CLAIMS	3 minus 3 =	* Ø
MULTIPLE DEPENDENT CLAIM PRESENT		☐

* If the difference in column 1 is less than zero, enter "0" in column 2

SMALL ENTITY TYPE ☐ OR OTHER THAN SMALL ENTITY

RATE	FEE		RATE	FEE
BASIC FEE	385.00	OR	BASIC FEE	770.00
X$ 9=		OR	X$18=	522
X43=		OR	X86=	
+145=		OR	+290=	
TOTAL		OR	TOTAL	1292

CLAIMS AS AMENDED - PART II

SMALL ENTITY OR OTHER THAN SMALL ENTITY

AMENDMENT A

	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA
Total	*	Minus	**	=
Independent	*	Minus	***	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM				☐

RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
X$ 9=		OR	X$18=	
X43=		OR	X86=	
+145=		OR	+290=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

AMENDMENT B

	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA
Total	*	Minus	**	=
Independent	*	Minus	***	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM				☐

RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
X$ 9=		OR	X$18=	
X43=		OR	X86=	
+145=		OR	+290=	
TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

AMENDMENT C

	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA
Total	*	Minus	**	=
Independent	*	Minus	***	=
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM				☐

RATE	ADDITIONAL FEE		RATE	ADDITIONAL FEE
X$ 9=		OR	X$18=	
X43=		OR	X86=	
+145=		OR	+290=	
TOTAL ADDIT. FEE		OR	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.

FORM PTO-875 (Rev. 10/03) Patent and Trademark Office, U.S. DEPARTMENT OF COMMERCE

Our Docket/Ref. No.: APPT-001-1-1 Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): Serial No.: 10/684,776 Filed: October 14, 2003 Title: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK	Group Art Unit: 2155 Examiner:

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL: INFORMATION DISCLOSURE STATEMENT

Dear Commissioner:

Transmitted herewith are:

_X_ An Information Disclosure Statement for the above referenced patent application, together with PTO form 1449 and a copy of each reference cited in form 1449.

____ A payment for petition fees.

_X_ Return postcard.

_X_ The commissioner is hereby authorized to charge payment of any missing fee associated with this communication or credit any overpayment to Deposit Account 50-0292.

A DUPLICATE OF THIS TRANSMITTAL IS ATTACHED

Date: Dec 8, 2003

Respectfully submitted,

Dov Rosenfeld
Attorney/Agent for Applicant(s)
Reg. No. 38687

Correspondence Address:
Dov Rosenfeld
5507 College Avenue, Suite 2
Oakland, CA 94618
Telephone No.: +1-510-547-3378

Certificate of Mailing under 37 CFR 1.18

I hereby certify that this correspondence is being deposited with the United States Postal Service addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.
Date of Deposit: Dec 8, 2003 Signature:
Dov Rosenfeld, Reg. No. 38687

Our Docket/Ref. No.: APPT-001-1-1 Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): Serial No.: 10/684,776 Filed: October 14, 2003 Title: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK	Group Art Unit: 2155 Examiner:

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL: INFORMATION DISCLOSURE STATEMENT

Dear Commissioner:

Transmitted herewith are:

_X_ An Information Disclosure Statement for the above referenced patent application, together with PTO form 1449 and a copy of each reference cited in form 1449.

___ A payment for petition fees.

_X_ Return postcard.

_X_ The commissioner is hereby authorized to charge payment of any missing fee associated with this communication or credit any overpayment to Deposit Account 50-0292.

A DUPLICATE OF THIS TRANSMITTAL IS ATTACHED

Date: Dec 8, 2003

Respectfully submitted,

Dov Rosenfeld
Attorney/Agent for Applicant(s)
Reg. No. 38687

Correspondence Address:
Dov Rosenfeld
5507 College Avenue, Suite 2
Oakland, CA 94618
Telephone No.: +1-510-547-3378

Certificate of Mailing under 37 CFR 1.18

I hereby certify that this correspondence is being deposited with the United States Postal Service addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.
Date of Deposit: Dec 8, 2003 Signature:
Dov Rosenfeld, Reg. No. 38687

Our Docket/Ref. No.: APPT-001-1-1 DEC 17 2003 Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): Serial No.: 10/684,776 Filed: October 14, 2003 Title: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK	Group Art Unit: 2155 Examiner:

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

INFORMATION DISCLOSURE STATEMENT

Dear Commissioner:

This Information Disclosure Statement is submitted:

_X_ under 37 CFR 1.97(b), or
(Within three months of filing national application; or date of entry of international application; or before mailing date of first office action on the merits; whichever occurs last)

____ under 37 CFR 1.97(c) together with either a:
___ Certification under 37 CFR 1.97(e), or
___ a $180.00 fee under 37 CFR 1.17(p)
(After the CFR 1.97(b) time period, but before final action or notice of allowance, whichever occurs first)

____ under 37 CFR 1.97(d) together with a:
___ Certification under 37 CFR 1.97(e), and
___ a petition under 37 CFR 1.97(d)(2)(ii), and
___ a $130.00 petition fee set forth in 37 CFR 1.17(i)(1).
(Filed after final action or notice of allowance, whichever occurs first, but before payment of the issue fee)

_X_ Applicant(s) submit herewith Form PTO 1449-Information Disclosure Citation together with copies, of patents, publications or other information of which applicant(s) are aware, which applicant(s) believe(s) may be material to the examination of this application and for which there may be a duty to disclose in accordance with 37 CFR 1.56.

Certificate of Mailing under 37 CFR 1.18

I hereby certify that this correspondence is being deposited with the United States Postal Service addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450
Date of Deposit: Dec 8, 2003 Signature:
Dov Rosenfeld, Reg. No. 38687

____ (Certification under 37 C.F.R. 1.97 (e)) Each item of information contained in this information disclosure statement was first cited in a communication from a foreign patent office in a counterpart foreign application not more than three months prior to the filing of this information disclosure statement.

_X_ (Cited in a related case) Each item of information contained in this information disclosure statement was first cited in a communication from the U.S. Patent and Trademark Office in a related application. The present application is related to such other applications by claiming priority of the same U.S. Provisional patent application.

It is expressly requested that the cited information be made of record in the application and appear among the "references cited" on any patent to issue therefrom.

As provided for by 37 CFR 1.97(g) and (h), no inference should be made that the information and references cited are prior art merely because they are in this statement and no representation is being made that a search has been conducted or that this statement encompasses all the possible relevant information.

Respectfully submitted,

Date: Dec 8, 2003

Dov Rosenfeld
Attorney/Agent for Applicant(s)
Reg. No. 38687

Correspondence Address:
Dov Rosenfeld
5507 College Avenue, Suite 2
Oakland, CA 94618
Telephone No.: +1-510-547-3378

INFORMATION DISCLOSURE STATEMENT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.
APPT-001-1-1	10/684,776
APPLICANT	
FILING DATE	GROUP
14 Oct 2003	2155

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
	AA	5,680,585	10-1997	Bruell	703	26	
	AB	5,721,827	02-1998	Logan et al.	709	217	
	AC	6,272,151	08-2001	Gupta et al.	370	489	
	AD	6,430,409	08-2002	Rossmann	455	422.1	
	AE	6,516,337	02-2003	Tripp et al.	709	202	
	AF	6,519,568	02-2003	Harvey et al.	705	1	
	AG						
	AH						
	AI						
	AJ						
	AK						

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLI-CATION DATE	COUNTRY	CLASS	SUB-CLASS	TRANS-LATION YES	NO
	AM							
	AN							

OTHER DISCLOSURES (Including Author, Title, Date, Pertinent Pages, Place of Publication, Etc.)

	AR	
	AS	

EXAMINER	DATE CONSIDERED

*EXAMINER: initial if citation considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to Applicant.

al.FORM - 1449 SHEET 2 OF 3.

INFORMATION DISCLOSURE STATEMENT

(Use several sheets if necessary)

OIPE DEC 17 2003 PATENT & TRADEMARK OFFICE

ATTY. DOCKET NO.	SERIAL NO.
APPT-001-1-1	10/684,776
APPLICANT	
FILING DATE 14 Oct 2003	GROUP 2155

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
	BA	5,850,388	12-1998	Anderson et al.	370	252	
	BB	6,097,699	08-2000	Chen et al	370	231	
	BC	6,269,330	07-2001	Cidon et al.	704	43	
	BD	6,453,345	09-2002	Trcka et al.	709	224	
	BE	6,381,306	04-2002	Lawson et al.	379	32	
	BF	6,282,570	08-2001	Leung et al.	709	224	
	BG	5,761,429	06-1998	Thompson	709	224	
	BH	5,799,154	08-1998	Kuriyan	709	223	
	BI						
	BJ						
	BK						

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLI-CATION DATE	COUNTRY	CLASS	SUB-CLASS	TRANS-LATION YES	NO
	BM							
	BN							

OTHER DISCLOSURES (Including Author, Title, Date, Pertinent Pages, Place of Publication, Etc.)

	BR	Advanced Methods for Storage and Retrieval in Image; http://www.cs.tulane.edu/ww/Prototype/proposal.html; 1998
	BS	Measurement and Analysis of the Digital DECT Propagation Channel; IEEE; 1998

EXAMINER	DATE CONSIDERED

*EXAMINER: initial if citation considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to Applicant.

al.FORM - 1449 SHEET 3 OF 3.

INFORMATION DISCLOSURE STATEMENT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.
APPT-001-1-1	10/684,776
APPLICANT	
FILING DATE	GROUP
14 Oct 2003	2155

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
	CA	5,530,958	06-1996	Agarwal et al.	711	3	
	CB	4,458,310	07-1984	Chang, Shih-Jeh	711	119	
	CC	6,003,123	12-1999	Carter et al.	711	207	
	CD	5,530,834	06-1996	Colloff et al.	711	136	
	CE	5,749,087	05-1998	Hoover et al.	711	108	
	CF	3,949,369	04-1976	Churchill, Jr.	711	128	
	CG	4,559,618	12-1985	Houseman et al.	365	49	
	CH	4,910,668	03-1990	Okamoto et al.	711	207	
	CI	5,917,821	06-1999	Gobuyan et al.	370	392	
	CJ						
	CK						

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLI-CATION DATE	COUNTRY	CLASSIFICATION	TRANS-LATION YES	NO
	CM	JP 2003-44510A	02-2003	Japan	G06F017/30		
	CN						

OTHER DISCLOSURES (Including Author, Title, Date, Pertinent Pages, Place of Publication, Etc.)

	CR	
	CS	

EXAMINER	DATE CONSIDERED

*EXAMINER: initial if citation considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to Applicant.

Title: Advanced Methods for Storage and Retrieval in Image Databases

Prototype Lead: Dr. Cris Koutsougeras, (504)862-3369, ck@eecs.tulane.edu

Proposed Funding Source: ESDIS Prototyping

Type: Prototype

Category: Engineering

Primary Purpose: Technology Evolution

Key Requirements Addressed: IMS-0150, IMS-0160, and IMS-190

Key Risks Addressed: 089 and 105

Results Need Date: N/A

Objective

We will continue to develop image/granule analysis and retrieval methods based on queries over metadata descriptions of the images. We will be working closely with members of the MODIS science team and, for development and testing purposes, will base our research on MODIS products. In particular, we will use a land surface Level 3 product (MOD13 including NDVI) and an atmospheric product. The atmospheric product is likely to be the Level 2 Cloud Product (MOD06) but discussions are still underway with MODIS participants about this and about a possible oceanic product as well. The prototype that currently exists at Tulane University will be extended to include these products. The principle on which the Tulane prototype is based is that an abstraction of an image/granule in the form of metadata is immediately accessible, not the image/granule itself. Queries by researchers are performed over the metadata. Selected images/granules are then transmitted to the researcher on a delayed basis. Extending the prototype using these diverse MODIS products will enable us to not only directly benefit the MODIS science team but also to develop methods that will assist DAAC users in general.

On a broader front, we will be addressing the issue of a uniform interface for heterogeneous data (Level 3 requirement IMS-0150), providing different levels of user interaction support (IMS-0160), and investigating how to save knowledge between metadata searches (IMS-0190). The effort will be a step toward defining precisely the set of user services needed (Risk 089) and reducing the likelihood that significant data will be overlooked due to the large volumes of data managed by EOSDIS (Risk 105).

Approach

A prototype has been implemented as an end-to-end, client-server testbed (Fig. 1). This testbed, provides a close analog to the ECS Science Data Processing Segment (SDPS) subsystems. The prototype has at its core the ObjectStore database and most of the current contents are based on the NDVI product. The interface is interactive and web-based. Queries are performed over simple, atomic data as well as some structured data, e.g., image histograms. A set of metadata has been defined but is subject through this study to modification as we concentrate on new products and interact with the MODIS science team.

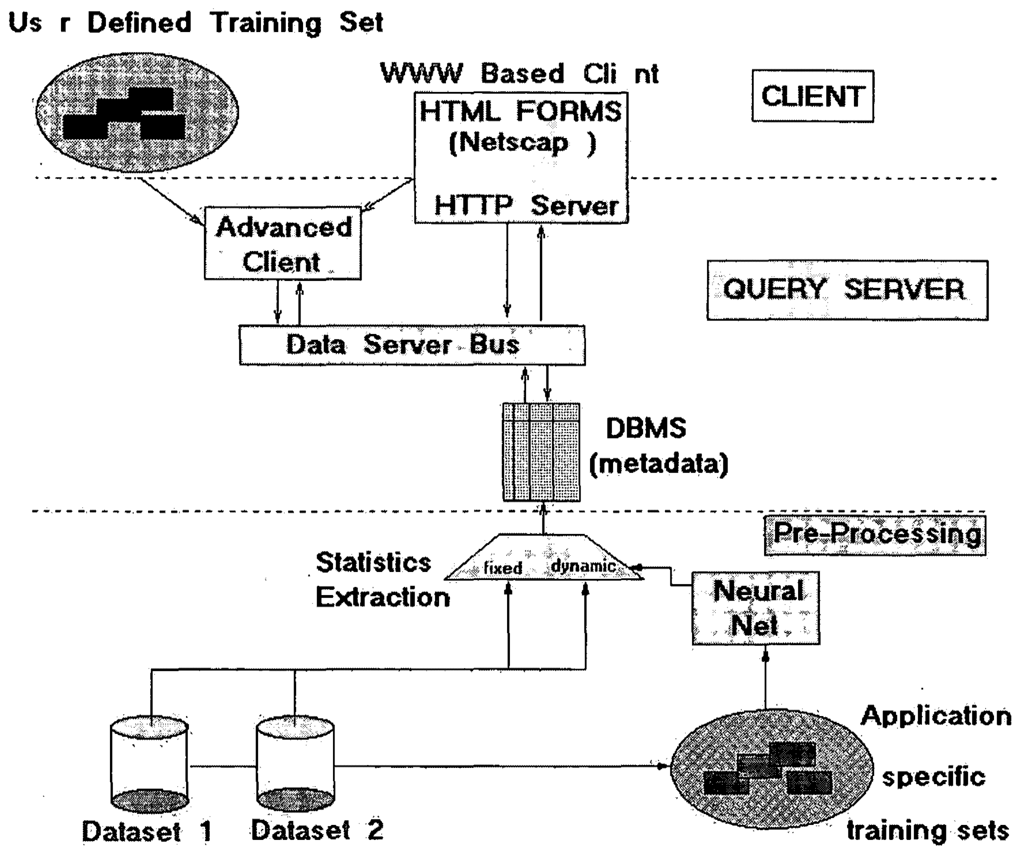


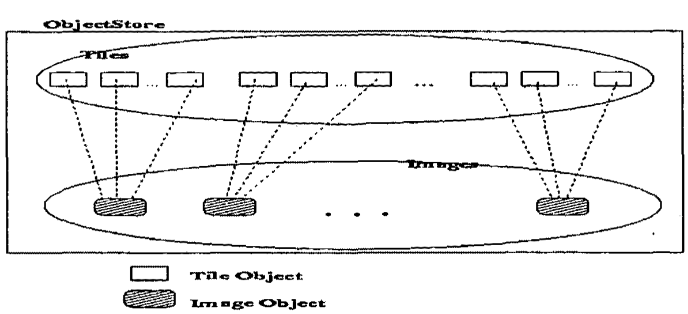
Fig. 1: Architecture of the Tulane EOS Query Prototype

Architecture of Present System. Because ObjectStore is an object-oriented database, the schema is discussed in terms of the object-oriented concepts of classes, data members, and methods. It is important to distinguish between geodata and metadata. Geodata, sometimes called granule level metadata, deals with the context of the image and includes information such as: date, flight, time, perspective (angle), instrument, latitude, data format, longitude, and mission.

Metadata deals with the content of the image and includes such things as the mean irradiance, cloud cover percentage, and texture measures.

The Database Classes. There are two classes employed -- image and tile. An object of class image contains an image and the geodata associated with it. An object of class tile contains the descriptive metadata and the identities of the image/subimage to which the object pertains. "Tile" is the word we use to designate the statistics associated with any one member of the quad tree frames of an image. The tile objects are collected into a container set called **tiles** and the image objects are collected into a container set called **images.** Images are stored in the database strictly for development purposes. In a deployable system, the images would be stored on another media in a data warehouse.

Fig. 2 depicts the relationship among the objects and their container sets. There is a many-to-one relationship between the tile objects and image objects. Each tile object contains the metadata for a portion of the image. Each tile object is stored separate from the image object to which it refers. This permits (1) uniform querying of metadata at the image and subimage levels (2) changing the tiling protocol or philosophy without having to change the database implementation.



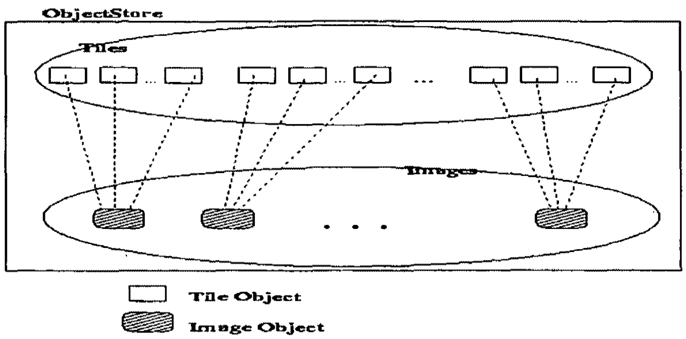


Fig. 2: Relationship Among Objects

Table 1 shows several of the two dozen metadata members of a tile object in the prototype implementation. The first three members, TileID, ImageId, and Channel, identify the tile and correlate it to an image object. The remaining data members are metadata values over which queries are performed.

Table 1: Tile Object Data Members

Metadata	Data Type	Description
TileId	String	Tile identity as described in Fig. 2
ImageId	Integer	Image of which tile is a subimage
Channel	Integer	Instrument band number
PixMean	Real	Spatial mean of all pixels
PixStdDev	Real	Standard deviation of pixel values
PixMode	Integer	Pixel value mode statistic
PixMin	Integer	Minimum pixel value
PixMax	Integer	Maximum pixel value
Histogram	Integer[264]	Number of pixels in each histogram bin
CloudPixels	Real	The fraction of the tile having cloud cover
NullPixels	Real	The fraction of pixels having null values

FourPhase	Real[20][20]	The phase angles of the Fourier transform
FourAmp	Real[20][20]	The amplitude of the Fourier transform
Wavelet	Real[20][20]	Wavelet (Haar) coefficients
Amplitude	Real[20]	Slope Real Fractal dimension

Sample Queries for the Proposed System. The purpose of metadata collection is to make accessible to the Earth science community the granules collected daily. To profile by example the kinds of uses that an EOS DAAC must be responsive to, we present two typical query scenarios.

Scenario 1 User Goal: Find areas experiencing recent and ongoing deforestation

Approach Using the MODIS NDVI 100km by 100km, look for both "high" standard deviation and "high" roughness. Omit tiles having a high percentage of cloudiness.

Discussion The tiles having the above characteristics are likely the ones representing the boundaries of the forest.

Scenario 2 User Goal: Study lee-wave cloud formation – clouds having a distinctive linear cloud pattern.

Approach Find (using any means) a set of tiles having the desired pattern and another set lacking it within the cloud product (MOD06). Cluster the training set using the Fourier signatures to define the texture of the desired tiles. Use the query system to search for images having a similar texture.

Discussion This is an example of an "advanced client" query. The researcher must put more effort into the initial search. Afterwards, the work can be reused by the researcher as well as by others.

Relationship of the Proposed Research to the Present Tulane Prototype. Presently, a researcher must be familiar with the nature of the metadata and their potential applications. This is because the researcher has to be able to express what (s)he is searching for and express the search requirements in terms of the metadata. The prototype web site performs simple queries over single-valued metadata variables such as

Retrieve images and subimages having a standard deviation of intensity less than x .

On structured data such as histograms, somewhat more advanced queries are possible such as submitting an exemplar histogram and retrieving the images/subimages that have similar histograms.

The next step is putting in place an interface that assists this translation in accordance with the spirit of Level 3 requirement IMS-0160. Although one could envision this interface as being so sophisticated that it is able to translate a sort of natural language query into a query involving the ground procedures, classes, and metadata of the database, it would first be necessary to establish the concept at a more modest scale.

Evaluation Method/Criteria

We will work in conjunction with MODIS science team members. These members will have remote access to the web site and will be able to provide specific requirements and provide feedback based on hands-on experience with the prototype.

Potential Impact

The prototype will address three significant IMS level 3 requirements: (1) uniform user interfaces, (2) interaction methods for different researchers with different skill levels, and (3) saving search knowledge between query sessions by researchers. It will address two key risks: (1) lack of well-defined user services and (2) overlooking significant information due to the volume of data managed. Additionally, there is a short-term impact that should not be overlooked. It will be of direct and immediate benefit to the MODIS science team members.

Milestones and Deliverables

Scope of Work. Using the current prototype as a basis, Tulane University will extend the system by emphasizing two (and possibly three) MODIS Level 2 and 3 products. The extensions include providing query support that requires different skill levels of researchers using the system, saving search knowledge between query sessions, and developing better approaches to characterizing the information at the subimage/subgranule level. The prototype made available to MODIS science team members may entail revising the metadata as now

defined based on their feedback.

The duration of the work outlined in this SOW is 12 months.

Summary of Tasks. The efforts during this period of performance will target further refinements to the existing work. More specifically, we target the issues of efficient storage (space problem) and efficient retrieval. We can build on the framework that has been created, refine it, and make it more user friendly.

Further development of the prototype will follow two directions: (1) Development of methods that use the available storage space more effectively (see Tasks 2 and 3); (2) Development of interface mechanisms that allow greater query flexibility for researchers having more expertise in data management (see Tasks 4 and 5).

Task 1: Project Management. Provide monthly progress reports to the Project Manager and Technical Lead. These 1-2 page reports will summarize the previous month's progress, plans for the upcoming month, and any identified issues or impediments to progress. These reports will also provide estimated dollar and labor contract expenditures for that month.

Task 2: Image Decomposition. Decomposition by quad tree, as described in the Approach section, is often preferred by scientists using land products. Frequently the phenomenon sought by such scientists is not distributed across the entire image. Almost always, it is contained in a subimage and the criteria used to find it will often be masked by the image as a whole. That is why the present prototype computes the metadata over each tile in a quad tree.

Yet, the quad tree has drawbacks. Atmospheric and oceanic scientists sometimes prefer other forms of decomposition. In a quad tree, the chief object in an image, say, a hurricane, is may be partitioned among different tiles. Clearly metadata particular to the storm will be washed out by the background that surrounds the parts. Other decomposition methods require the use of criteria based on image content. This might be pixel values or texture. However, decomposing the image into "blobs" that represent areas of uniformity is a plausible approach. Blobs fall short of complete image segmentation by region in that they need not fully partition the image and may overlap. They constitute an augmentation of the quad tree decomposition, not a replacement for it. Metadata should be computed for each blob in addition to each quad tree tile. A blob will more likely contain a coherent object in its entirety.

To illustrate, examine Fig. 3. Using ellipses as the basic geometry of a blob, the significant features on an image are enclosed. Shapes other than ellipses will be investigated. With respect to the database schema, minimal changes are necessary. A blob description will just become an additional tile in the illustration of Fig. 2. The metadata for a blob is the same as that for a tile.

Task 3: Data Space Conservation. Data concerning terabytes of mages must be immediately accessible. To conserve space, we will investigate ways to discover parts of images (partial images) that reoccur and thus can be seen as building blocks. These can be helpful in composing an image (or many images) but metadata descriptors of each of these components need to be stored only once. As a simplistic example, consider an image that contains large bodies of water over which there is a uniform distribution of pixel values. If these areas were represented as repetitions of a single tile, we would only need to store a detailed (sub)image description of this tile and then describe the large bodies of water as a (structured) collection of pointers to it, thus saving substantially in the space required to store the image metadata.

Fig. 3: Blob Decomposition of an Image

There are a couple of different possibilities for discovering building blocks. First, we can try to identify reoccurring tiles within an image. Second, we can try to discover tiles that are useful in many images. The process of discovery must be automated and run at ingest time. This problem has properties similar those of binpacking and thus an optimal solution will not be easy to find. However, evolutionary algorithm techniques are promising and may provide good practical solutions The issues to be addressed in this task are: (1) determining the geometry of the tiles, (2) tradeoffs in space savings in conjunction with the geometry of the tiles (this relates to the frequency of reoccurrence), and (3) evolutionary algorithm techniques for optimal decompositions.

Task 4: Advanced and Reconstructive Metadata. During the year just past, we examined and chose metadata that are statistically significant, that have high information content, and that researchers are already familiar with. We did not (and were not required to) incorporate mechanisms for using the more highly structured metadata (such as texture and transform coefficients) into queries. We did go beyond the requirements of the previous grant and develop means of using histogram information within queries.

In the coming year we will develop query mechanisms to complement all the metadata. Additionally, because we have defined and are collecting metadata from which the image can be reconstructed at reduced resolution, we will develop mechanisms by which the researcher can define her/his personal metadata and collect it from any set of images defined via less advanced queries. For example, consider the Fourier coefficients. They require much storage but contain much information. A researcher might reconstruct the image or tiles from an image using the Fourier coefficients then compute the eigenvalues of the pixel autocorrelation matrix. This is the sort of query for which we can provide the basic structure through this task.

Task 5: Query Reuse. The result of a query is a set of image and tile id's. The simplest approach to query reuse (and the one we will start with) is to associate with various sets a "cluster condition," a Boolean statement which each member of the set satisfies. This leads to the ability to reuse a set when the condition reoccurs. Next we will elaborate this in the form of an inverted file for which the key is a tile id from which the complete set of cluster conditions (seen thus far) satisfied by that tile can be accessed. This will achieve a powerful query reuse capability.

Deliverables. NASA will have the right to examine the software at any time. Provision to FTP software to NASA will be made on an informal basis. Three additional deliverables are proposed.

> **Item A: Decomposition approach presented as written document. This document will define the approaches used for decomposition (other than the quad tree) and the reasons for which one was selected. The issues of feasibility of performing the method upon ingest will be described together with experiences using the method. This is the culmination of Task 2.**
>
> **Item B: Operational end-to-end WWW-based prototype allowing direct query of fixed (statistical) metadata, utilization of advanced queries that incorporate structured metadata, and allow a researcher to utilize reconstructive metadata (e.g., Fourier coefficients) to define her/his personal metadata.**
>
> **Item C: Final report summarizing "lessons learned," focusing primarily on the architecture of the prototype, the design of the components that constitute the query interface, and changes that would be appropriate if another database product were utilized as the core of the system.**

Bear in mind that in additional to the deliverables, the MODIS science team will be able to gain direct benefit from the project beginning in late September and continuing throughout the prescribed period of performance.

Measurement and Analysis of the Digital DECT Propagation Channel*

Fulvio Babich[1], Giancarlo Lombardi[1], Steno Schiavon[2], Elvio Valentinuzzi[1]

1: Dipartimento di Elettrotecnica, Elettronica e Informatica,
Università di Trieste, Via A. Valerio 10, I-34127 Trieste, Italy
Phone: +39-40-676-3458; Fax: +39-40-676-3460
babich, lombardi, valent@stelvio.univ.trieste.it
http://ingsun1.univ.trieste/~{babich, lombardi}

2: TELITAL S.p.A., Viale Stazione di Prosecco 5/B, I-34010 Sgonico (TS), Italy,
Phone: +39-40-4192-244; Fax: +39-40-251257; steno.schiavon@rs1.telital.it

Abstract – In this paper, an experimental setup is presented, to measure bit error patterns over a DECT indoor radio channel. A prefixed bit sequence has been exchanged on the air between a mobile and a fixed part, using a DECT modem. DECT interferers were active in the environment during the experiments. Error patterns have been obtained from received sequences, aligning and comparing them with the transmitted ones. They have been stored in real time on a mass memory, by means of a data acquisition board, built for this purpose. Some results are shown, inherent to the Packet Error Distribution (PED) and to the burst and interburst length distributions, obtained from the acquired database. Finite State Markov Channel models have been determined, using measurement conditions, to reproduce and verify empiric results.

1 Introduction

Wireless communications are experiencing a considerable growth, due to their flexibility. The quick increase of the subscriber number requires to reconsider system architecture, in order to adapt it to the propagation environment, as carefully as possible, and, consequently, to obtain capacity gain. Given that urban and indoor wireless communications are the most requested from users, Cordless Telecommunication (CT) systems are being perfectioned, because they allow better coverage and capacity, with lower power expense, than traditional cellular systems. Among European standards, DECT is the most modern digital standard, providing a broad range of services.

Being the mobile propagation channel (with interference) the main reason of degradation of the transmitted signal, it may be useful to evaluate the channel correlation properties, to increase system capacity by taking into account channel memory [1]. In fact, the traditional techniques of coding and interleaving destroy error correlation at the receiver, but do limit system capacity. It is useful, then, to study the correlation properties of bit error sequences and to find models reproducing their behavior. Generally, bit error streams on a wireless channel are obtained, through simulation or as post-processing of experimental analog measurements [2, 3, 4].

In this paper, an experimental setup is presented, to measure and store bit error patterns over a DECT indoor radio channel. Measurements have been performed, building a database of bit error sequences in an indoor environment. The packet error distribution (PED) and the burst/interburst length distributions have been obtained from some streams of the database and compared with the distribution obtained from a Finite State Markov Channel model, proposed in [5] and whose parameters are estimated from measurement conditions.

The measurement setup is described in Section 2, while measurement execution is outlined in Section 3. Results about PED and burst/interburst length distributions are discussed in Section 4 and 5, respectively, while some conclusions are drawn in Section 6.

2 Measurement Setup

The measurements have been performed, using as transmitter and as receiver, two radio communication testers for DECT systems (CMD60, manufactured by R&S), respectively. The scheme of measurement apparatus is displayed in Figure 1. A continuous TTL

*This work has been partially supported by MURST "ex-quota 40%", Italy.

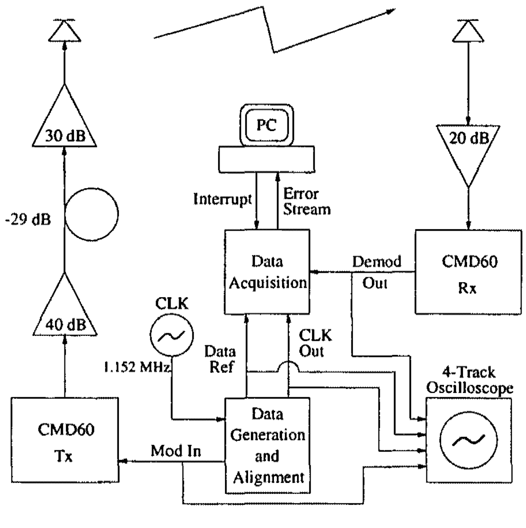


Figure 1: Scheme of the Measurement Setup.

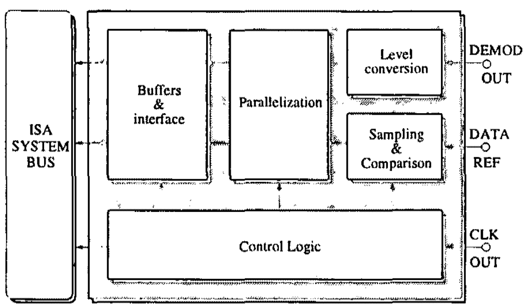


Figure 2: Scheme of the data acquisition board.

binary sequence, having DECT bit rate (1.152 Mb/s), is generated and supplied to the first tester, where it modulates a DECT carrier at 1897.344 MHz in the DECT GMSK format ($BT = 0.5$). This signal is amplified by 40 dB, transmitted over a 80 m coaxial cable (attenuation 29 dB) to the mobile end, where it is amplified again, by 30 dB, before being fed to a discone transmitting antenna.

The receiving part is made by another discone antenna, similar to previous, connected by a 5 m coaxial cable to a 20 dB low noise amplifier, that supplies the signal to the second tester, where the received binary sequence is obtained by a frequency discriminator. This sequence is aligned with the transmitted sequence, used as reference, because the delay of the measurement chain is about one and a half bit.

The comparison between transmitted and received sequence and the storage of the error sequence inside a PC Pentium 133 MHz are made, using a dedicated acquisition digital board, controlled via software using the interrupt management. The sampling and decision circuitry of the received signal are also on the board. It is composed by the blocks showed in Figure 2. The demodulated sequence (DEMOD OUT), having a Gaussian bit shape, is sampled and converted to TTL levels, to be compared with the reference sequence (DATA REF). The high rate of the data requires a parallel acquisition: shift registers, suitably joined to the serial data line, are used. The parallelized data are stored inside buffers, to be available on the ISA System Bus, carrying them to the PC. The control circuitry is for address and interrupt management.

3 Measurement Execution

The measurements have been executed in the laboratory and office wing of a PCS factory, near Trieste. The plan can be found in [6]. The basic structure of the floor is given by a very long hallway, terminated by a large laboratory and interrupted by metal fire-cut doors. Along the hallway, several medium sized room (in mean 5 × 5 m) and stairs are displaced. The furnishing is typical of offices and laboratories, with several metal cabinets and work benches. Further details are given in [6, 7].

The experiments have been carried on after office time, so that very few people was in the environment and stationarity can be assumed. On the other side, several DECT terminals were active and it has been observed that interference is the main reason of error during measurement execution. The receiving antenna is fixed on a 2.10 m dielectric pole, while the transmitting one is moving and kept at a 1.90 m height. The mobile antenna has been continuously displaced along straight and circular paths. Straight paths have a minimum length of 7 m (in the rooms) and a maximum length of 35 m (in the main hallway). The antenna was held on a vertical pole mounted on a trolley and moved by a person, with speed ~ 0.4 m/s. Circular paths have a ray of 1.5 m. The antenna is held on a horizontal arm, mounted on a vertical pole rotating around its axis by means of an electromagnetic engine. The angular speed impressed by the engine is 2π rad/min, corresponding to an antenna peripheric speed of 0.16 m/s.

Three positions of the receiving antenna have been chosen, two in rooms along the hallway and one in the lab terminating the hallway. The paths of the transmitter has been chosen, trying to cover many rooms of the floor. The measurement along every path has been carried on with transmitter constant power. The experiment at any path has been repeated varying the signal power at modulator output among the values -15, -25 and -35 dBm, respectively (about 40 dB have

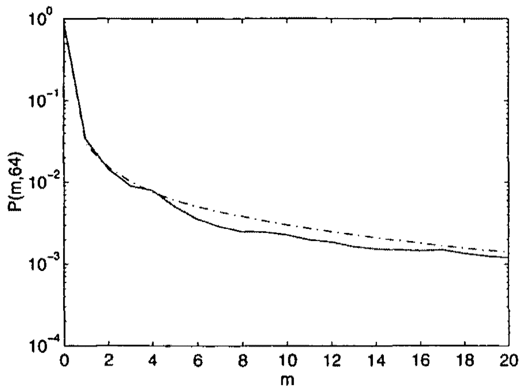


Figure 3: Packet Error Distribution $P(n, m)$ with $n = 64$. Solid line: Measure; Dashed Line: Theory.

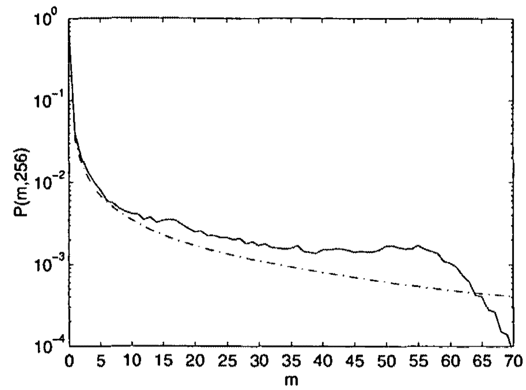


Figure 4: Packet Error Distribution $P(n, m)$ with $n = 256$. Solid line: Measure; Dashed Line: Theory.

to be added to these values to obtain radiated power). The final database contains 32 error streams obtained from straight paths and 33 error streams obtained from circular paths.

4 PED Evaluation

The database has been pre-processed to transform the bit error sequences in gap sequences, according to definitions in [8].

The first quantity evaluated from experimental data is Packet Error Distribution (PED). PED $P(n, m)$ is defined as the probability that a data block, made by n bits, contains n errors and is connected to Packet Error Rate PER by the equation:

$$\text{PER} = \sum_{m=t+1}^{n} P(n, m) = 1 - \sum_{m=0}^{t} P(n, m), \quad (1)$$

being t the number of errors in the block, that can be corrected.

Signal fading rate is very low: $f_D T_b = 8.7 \cdot 10^{-7}$, for circular paths, and $f_D T_b = 2.2 \cdot 10^{-6}$, for linear paths, where $T_b = 868$ ns is bit duration, $f_D = \frac{v}{\lambda}$ is channel Doppler spread, v is mobile speed and $\lambda = 15.7$ cm is the wavelength. However, errors are caused mainly by interference, originating from few base stations, transmitting signaling packets (96 bit long) in every slot ($T_I = 417\mu$s). To represent the instant signal-to-interference ratio (SIR), it seems reasonable to use the Finite State Markov Channel model (FSMC), proposed in [5], with fading rate $f_D T_I$. This means assuming that interference has approximately the same effect of fading with $f'_D = f_D \frac{T_I}{T_b}$. The model considers Rayleigh fading, quantized on L levels, by means of $L-1$ thresholds (referred to SIR [Section 3]) $\{A_k\}_{k=1}^{L-1}$. L fading states $\{S_k\}_{k=0}^{L-1}$ are obtained, so that $\{S_k : \text{SIR} \in [A_k \; A_{k+1})\}_{k=0}^{L-1}$, where $A_0 = 0$ and $A_L = +\infty$.

The FSMC model is completely characterized by the transition matrix $\mathbf{T} = \{t_{j,k}\}_{j,k=0}^{L-1}$ and by the crossover probabilities $\mathbf{e} = \{e_k\}_{k=0}^{L-1}$ (where e_k is the average error probability over the Binary Symmetric Channel, corresponding to the SIR in the state S_k). The transition probabilities are given by:

$$t_{k,k+1} \approx \frac{N_{k+1}}{R_t^{(k)}}, \quad k = 0, \ldots, L-2, \quad (2a)$$

$$t_{k,k-1} \approx \frac{N_k}{R_t^{(k)}}, \quad k = 1, \ldots, L-1, \quad (2b)$$

$$t_{i,j} \approx 0, \quad \forall i, j : |i - j| > 1, \quad (2c)$$

$$t_{k,k} = 1 - \sum_{l \neq k} t_{k,l}, \quad (2d)$$

where $R_t^{(k)} = \frac{p_k}{T_b}$, $N_k = \sqrt{\frac{2\pi A_k}{\varrho}} f'_D \exp\left(-\frac{A_k}{\varrho}\right)$, ϱ is the average SIR, $p_k = \exp\left(-\frac{A_k}{\varrho}\right) - \exp\left(-\frac{A_{k+1}}{\varrho}\right)$, $k = 0, \ldots, L-1$ are the steady state probabilities for each state S_k. The crossover probabilities for an incoherently demodulated 2-FSK, well approximating GMSK, are given by:

$$e_k = \frac{1}{p_k(2+\varrho)} \left\{ \exp\left[-A_k \left(\frac{1}{\varrho} + \frac{1}{2}\right)\right] - \exp\left[-A_{k+1} \left(\frac{1}{\varrho} + \frac{1}{2}\right)\right] \right\}, \quad (3)$$
$$k = 0, \ldots, L-1.$$

PED has been evaluated from this model with the algorithm, described in [8, 3].

Figs. 3 and 4 compare measured PED with PED obtained from FSMC model, for a particular circular path measurement, with -15 dBm power level at

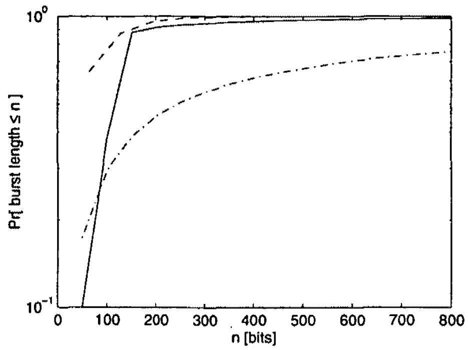


Figure 5: Burst length distribution. ——: Measure; -·-·: First order model at bit level; - - -: First order model at block level.

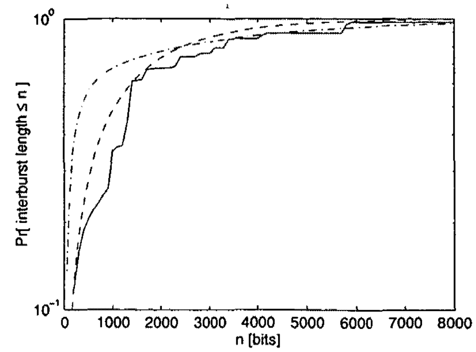


Figure 6: Interburst length distribution. ——: Measure; -·-·: First order model at bit level; - - -: First order model at block level.

modulator (SR1). In the former, packet length n is 64, while in the second it is 256. The mean bit error rate is $P_b = 1.46 \cdot 10^{-2}$, so that an average SIR $\varrho = 18.2$ dB is estimated from $P_b = \frac{1}{2+\varrho}$ [9]. Continuous line represents PED obtained from data, while dashed line represents PED obtained from FSMC model, evaluated using eqns. (2) and (3). The model has 12 quantization levels, using as thresholds $\{A_k = \varrho - 2(11 - k)\}_{k=1}^{11}$ dB.

It can be noticed that FSMC model follows well experimental results in Figure 3, while it shows some departure in Figure 4. Furthermore, it has been checked that PED behavior is quite insensitive to the value of $f_D T$ (if $f_D T \lesssim 10^{-3}$). It can be noticed that the effect of interference on PED is well represented by the FSMC model, defined by eqns. (2) and (3), used in presence of fading with AWGN [3]. This means that the effect of interference, also when interferers are temporally deterministic and in small number, can be considered Gaussian. The major departure of theory from measurement in Figure 4 puts on evidence the limits of the approximation.

5 Burst Distribution

The error stream has been also analyzed, considering the burst and interburst length distribution. A *burst* is defined as a group of bit, starting and ending with a wrong bit, such that the maximum separation between any couple of wrong bits is never higher than a fixed number N_G (*guard interval*) [10]. In this work, it is $N_G = 100$. The group of bits between two consecutive bursts is an *interburst*. Figs. 5 and 6 show with solid line the distribution of burst and interburst length, respectively, obtained from the same experimental data, used in Section 4. Inside each figure, the dash-dotted line represents the behavior obtained from the FSMC, therein described. The burst and interburst length distributions are approximated by the residence time distribution in the state set S_1–S_9 and S_{10}–S_{12}, considered as error and no-error states at bit level, respectively. The evaluation method is described in [11].

The dashed line represents the results obtained from a first order Markov model at block level, discussed in the following. Let $n = 64$ be the block length. Let G and B be the events of correct block (without errors) and of wrong block (containing wrong bits), respectively. Let a two-state FSMC model at the bit level be obtained, quantizing SIR with respect to a threshold F, to represent the process at bit level. A typical value, also used here, is $F = \varrho - 4$ (dB). Equations (2) give its transition matrix:

$$\mathbf{T}_{\text{bit}} = \begin{bmatrix} p_{bb} & p_{bg} \\ p_{gb} & p_{gg} \end{bmatrix},$$

and its stationary distribution $\mathbf{p}_{\text{bit}} = [p_b \; p_g]$, where g and b are the no-error and error states at the bit level, respectively.

A first order Markov chain describing the G-B process can be built from $\mathbf{T}_{\text{bit}}, \mathbf{p}_{\text{bit}}$ and its transition matrix is given by:

$$\mathbf{T}_{\text{block}} \simeq \begin{bmatrix} P_{BB} = 1 - P_{BG} & P_{BG} = p_g\,(p_{gg})^{n-1} \\ P_{GB} = 1 - P_{GG} & P_{GG} = (p_{gg})^n \end{bmatrix}$$

where $P_{BG} \simeq P(b_{m+1} = 0, \ldots, b_{m+n} = 0 | b_m = 1)p_b + P(b_{m+1} = 0, \ldots, b_{m+n} = 0 | b_m = 0)p_g$ and $P_{GG} \simeq P(b_{m+1} = 0, \ldots, b_{m+n} = 0 | b_m = 0)$ (given $n \gg 1$).

The burst and interburst length distributions are obtained from this model, as residence time in the B and G state, respectively.

The second model shows a closer resemblance with the experimental results, because the state definition matches better the burst and interburst definitions. Therefore, from the burst/interburst distribution point of view, the channel with interference is well represented even by an on-off Markov model, where the 'on' condition corresponds to SIR being above a suitable threshold F. Moreover, the first model exhibits larger deviation in the burst length case, because the approximation of a burst, as a sequence of totally wrong bits, is an oversimplification.

6 Conclusions and Future Work

In this paper, a measurement system has been described, for the acquisition of bit error streams on a DECT digital channel. A database of error patterns has been obtained in a laboratory and office environment, considering also the effect of some DECT interferers thereby placed. Some processing has been made on it, consisting in evaluation of Packet Error Distribution (PED) and of burst and interburst length distributions. The experimental results for PED match with the ones given by a FSMC model, typically adopted for fading channel with AWGN. This puts on evidence that the interference effect can be assumed Gaussian. Burst/interburst length distributions are correctly modeled even by a simple on-off model. Though, some deviations between models and experimental results have been also observed, due to the deterministic features of interference.

Future work will consist in examining more complex models, that are capable of giving more accurate results and that can be applied to further characterizing features of the channel, as the gap distribution. The aim is to find a class of models, capable to give a unitary description of digital wireless channel behavior, both at the bit level and at the packet level, so that parameters useful to system project are readily obtained from it.

Acknowledgments

The authors wish to thank the TELITAL S.p.A., for having supplied measurement instrumentation and having followed its execution with continuous and qualified technical support.

References

[1] A. Goldsmith and P. Varaiya, "Capacity, Mutual Information and Coding for Finite-State Markov Channels", *IEEE Trans. Inform. Theory*, vol. 42, no. 3, pp. 868 – 886, May 1996.

[2] A. J. Goldsmith, L. J. Greenstein, and G. J. Foschini, "Error Statistics of Real-Time Power Measurements in Cellular Channels with Multipath and Shadowing", *IEEE Trans. Veh. Technol.*, vol. 43, no. 3, pp. 439 – 446, Aug. 1994.

[3] H. Bischl and E. Lutz, "Packet Error Rate in the Non-Interleaved Rayleigh Channel", *IEEE Trans. Commun.*, vol. 43, no. 2/3/4, pp. 1375 – 1382, Feb./Mar./Apr. 1995.

[4] P. M. Crespo, R. Mann Pelz, J. P. Cosmas, and J. García-Frías, "Results of Channel Error Profiles for DECT", *IEEE Trans. Commun.*, vol. 44, no. 8, pp. 913 – 917, Aug. 1996.

[5] H. S. Wang and N. Moayeri, "Finite-State Markov Channel: A Useful Model for Radio Communication Channel", *IEEE Trans. Veh. Technol.*, vol. 44, no. 1, pp. 163 – 171, Feb. 1995.

[6] F. Babich, G. Lombardi, and E. Valentinuzzi, "Indoor Propagation Characteristics in DECT Band", in *Proceedings of IEEE VTC '96*, Atlanta, GA, Apr. 27 – May 2, 1996, pp. 574 – 578.

[7] F. Babich, G. Lombardi, L. Tomasi, and E. Valentinuzzi, "Indoor Propagation Measurements at DECT Frequencies", in *Proc. of IEEE MELECON '96*, Bari, Italy, May 13 – 16, 1996, pp. 1355 – 1359.

[8] L. N. Kanal and A. R. K. Sastry, "Models for Channels with Memory and Their Applications to Error Control", *Proc. IEEE*, vol. 66, no. 7, pp. 724 – 744, July 1978.

[9] J. G. Proakis, , *Digital Communications*, McGraw–Hill, New York, 1989.

[10] E. A. Newcombe and S. Pasupathy, "Error Rate Monitoring for Digital Communications", *Proc. IEEE*, vol. 70, no. 8, Aug. 1982.

[11] Attila Csenki, *Dependability for Systems with a Partitioned State Space*, vol. 90 of *Lecture Notes in Statistics*, Springer-Verlag, 1994.

PAT-NO: JP02003044510A

DOCUMENT-IDENTIFIER: JP 2003044510 A

TITLE: GATEWAY SYSTEM

PUBN-DATE: February 14, 2003

INVENTOR-INFORMATION:

NAME	COUNTRY
INOSHITA, AKIHITO	N/A
SUZUKI, HIROYOSHI	N/A
KUBOTA, HIROMI	N/A

ASSIGNEE-INFORMATION:

NAME	COUNTRY
MATSUSHITA ELECTRIC IND CO LTD	N/A

APPL-NO: JP2001225981

APPL-DATE: July 26, 2001

INT-CL (IPC): G06F017/30, G06F012/00 , G06F013/00

ABSTRACT:

PROBLEM TO BE SOLVED: To provide a gateway system that enables a network

terminal user to automatically surf valuable Web pages without any specified setting.

SOLUTION: An access monitor unit 25 of a gateway system 80 detects the URL for Webs a user frequently accesses and manages the URL with a URL management table 30. A surfing unit 40 of the gateway system automatically surfs the Webs having the URL and stores the Web data in a cache server 50. The gateway system generate a management table that includes not only the frequency of the accesses but also data for the elapsed time from the most recent accessed time to the present time and can automatically surf a Web site being judged as the high priority site based on the management table.

(19)日本国特許庁（ＪＰ）　(12)公開特許公報(Ａ)　(11)特許出願公開番号
特開2003－44510
(P2003－44510A)
(43)公開日 平成15年2月14日(2003.2.14)

(51)Int.Cl.⁷		識別記号	ＦＩ		テーマコード*(参考)	
Ｇ０６Ｆ	17/30	３４０	Ｇ０６Ｆ	17/30	３４０Ｂ	５Ｂ０７５
		１１０			１１０Ｆ	５Ｂ０８２
		４１９			４１９Ｂ	
	12/00	５４６		12/00	５４６Ｐ	
	13/00	５４０		13/00	５４０Ｂ	

審査請求 未請求 請求項の数12 ＯＬ (全 11 頁)

(21)出願番号 特願2001－225981(P2001－225981)

(22)出願日 平成13年7月26日(2001.7.26)

(71)出願人 000005821
松下電器産業株式会社
大阪府門真市大字門真1006番地

(72)発明者 井ノ下 明史
神奈川県横浜市港北区綱島東四丁目3番1号 松下通信工業株式会社内

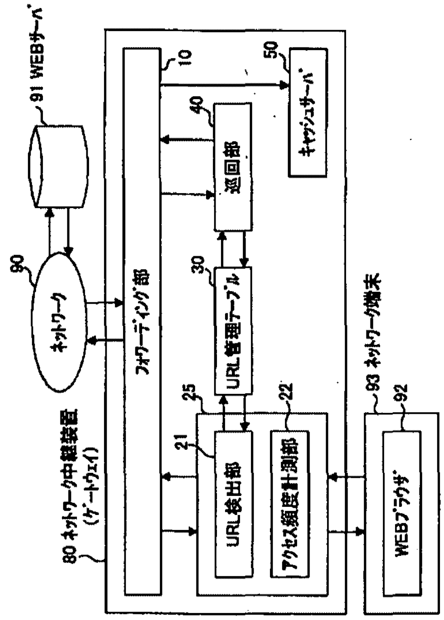
(72)発明者 鈴木 弘喜
神奈川県横浜市港北区綱島東四丁目3番1号 松下通信工業株式会社内

(74)代理人 100105050
弁理士 鷲田 公一

最終頁に続く

(54)【発明の名称】 ゲートウェイ装置

(57)【要約】
【課題】 ネットワーク端末のユーザが特別な設定を行わなくても、有効なウエブページを自動的に巡回できるようにすること。
【解決手段】 ゲートウエイ８０のアクセス監視部２５が、ユーザが頻繁にアクセスするウエブのＵＲＬを検出し、ＵＲＬ管理テーブル３０にて管理し、巡回部４０が、そのＵＲＬのウエブを自動的に巡回し、ウエブデータをキャッシュサーバ５０に蓄積する。アクセス頻度のみならず、直近のアクセス時点から経過した時間の情報を含む管理テーブルを作成し、これに基づいて優先度の高いＷＥＢサイトを判定して自動巡回を行うこともできる。



【特許請求の範囲】

【請求項1】 WEBブラウザを搭載したネットワーク端末とネットワーク上のWEBサーバとの通信を中継するゲートウェイ装置であって、IPパケットフォワーディング部と、前記ネットワーク端末から入力されたWEBサイトのURL検出部と、前記URLがWEBブラウザから入力された回数を計測するアクセス頻度計測部と、前記URLとそのアクセス回数を対応付けて記憶するURL管理テーブルと、前記URL管理テーブルを用いてネットワークを介してWEBサイトのデータを自動的に取得する巡回部と、前記巡回部が取得したWEBデータを記憶するキャッシュサーバを備え、前記ネットワーク端末からのアクセス頻度条件に応じて、自動でWEBデータを取得することを特徴とするゲートウェイ装置。

【請求項2】 前記URL管理テーブルは予め決められたタイムアウト時間を持つことで最後に更新したときから一定時間以上更新が無ければURL管理テーブルを更新しアクセス頻度を変更できることを特徴とする請求項1記載のゲートウェイ装置。

【請求項3】 前記URL管理テーブルは、ネットワーク端末からのアクセス頻度、更新日時順にアクセスする順番を自動、または手動で変更でき、前記巡回部は、ネットワーク端末からのアクセス頻度の高いURLから順に、予め決められた順位のURLまで巡回することを特徴とする請求項1記載のゲートウェイ装置。

【請求項4】 前記URL管理テーブルは、ネットワーク端末からのアクセス頻度、更新日時順にアクセスする順番を自動、または手動で変更でき、前記巡回部は、ネットワーク端末からのアクセス頻度の高いURLに対し、ある一定時間内にそのアクセス頻度の割合に応じた回数だけ巡回することを特徴とする請求項1記載のゲートウェイ装置。

【請求項5】 ゲートウェイに接続されるネットワーク端末が複数存在する場合において、ネットワーク端末を識別する識別部をさらに備え、前記URL管理テーブルおよび、前記キャッシュサーバを各ネットワーク端末ごとに管理することを特徴とする請求項1～請求項4のいずれかに記載のゲートウェイ装置。

【請求項6】 前記識別部はIPアドレスを用いて識別を行うことを特徴とする請求項5記載のゲートウェイ装置。

【請求項7】 前記識別部はMACアドレスを用いて識別を行うことを特徴とする請求項5記載のゲートウェイ装置。

【請求項8】 前記識別部はポート番号を用いて識別を行うことを特徴とする請求項5記載のゲートウェイ装置。

【請求項9】 前記巡回部により取得したHTMLのデータをCompactHTML（cHTML）に変換する変換部をさらに備え、前記キャッシュサーバに同一のWEBデータをHTMLとcHTMLの2種類のマークアップ言語で記憶しておくことを特徴とする、請求項1または請求項5記載のゲートウェイ装置。

【請求項10】 前記巡回部により取得したHTMLのデータをBMLに変換する変換部をさらに備え、前記キャッシュサーバに同一のWEBデータをHTMLとBMLの2種類のマークアップ言語で記憶しておくことを特徴とする請求項1または請求項5記載のゲートウェイ装置。

【請求項11】 ネットワーク端末がアクセスしたWEBサイトについて、URLおよびアクセス頻度，あるいは、URLとアクセス頻度と直近のアクセス時点から経過した時間の情報を含む管理テーブルを作成し、かつ作成された管理テーブルをアクセスの発生状況に応じて随時更新し、前記管理テーブルに含まれる情報に基づいて優先度の高いWEBサイトを判定して自動巡回を行い、その結果として得られた前記優先度の高いWEBサイトのデータを蓄積することを特徴とするWEB自動巡回方法。

【請求項12】 ゲートウェイ装置としてのコンピュータを、

ネットワーク端末がアクセスしたWEBサイトについて、URLおよびアクセス頻度，あるいは、URLとアクセス頻度と直近のアクセス時点から経過した時間の情報を含む管理テーブルを作成すると共に、作成された管理テーブルをアクセスの発生状況に応じて随時更新する手段と、

前記管理テーブルに含まれる情報に基づいて優先度の高いWEBサイトを判定して自動巡回を行い、その結果として得られた前記優先度の高いWEBサイトのデータを蓄積する手段として機能させるためのプログラム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、ゲートウェイ装置に関し、特に、WWW(World Wide Web)等のハイパーテキストのデータをネットワークを経由してサーバから取得するハイパーテキスト自動取得機能をもつゲートウエイに関する。

【0002】ゲートウエイは、異なるシステムや異なるネットワーク間を接続するための中継機能をもつ装置であり、近年、注目されるものに、家庭におけるネットワーク端末を制御するホームネットワークゲートウエイなどがある。

【0003】

【従来の技術】近年、複数のサーバコンピュータおよび複数のクライアントコンピュータがネットワークで結ばれ、各サーバコンピュータにハイパーテキスト構造のマルチメディアデータが記憶されており、各クライアントコンピュータにおいて、ブラウザソフトウェアによっ

て、このようなマルチメディアデータを閲覧することが可能なシステムが広く普及している。このようなシステムの例としては、例えば、インターネットにおけるWWW (World Wide Web) と呼ばれるシステムなどが挙げられる。マルチメディアデータを含む文書は、例えばHTML (hyper text markup language) と呼ばれる記述言語によって記述されており、テキスト文書、静止画、動画、音楽データ、およびJava (登録商標) アプレットなどのアプリケーションプログラムなどを含むことが可能になっている。このような文書 (以下、HTMLページと称する) およびマルチメディアデータには、それぞれURL (uniform resource locator) と呼ばれる固有のアドレスが割り当てられている。ユーザは、ブラウザ上においてURLを指定することによって、所望のHTMLページあるいはマルチメディアデータにアクセスすることができる。ネットワーク端末が、インターネット等のネットワーク上に分散されたWWW等のハイパーテキストのデータを取得する場合には、クライアント装置 (ネットワーク端末) に搭載されるユーザインターフェース·ツール「ブラウザ」を使い、ネットワーク上における目的のサーバ名とファイル名とを指定すると、上記ブラウザによって、指定されたファイルとそのファイルをメインとしたページを構成する他のファイルとが自動的に取得されて、ビジュアルに組み合わせて当該ページが表示される。

【0004】ここで、上記指定のファイルから互いに関連するページ間を移動して表示させるためには、1つのページから関連するページへのリンク等を1つずつ指定して順に表示する必要がある。

【0005】また、扱うファイル群は、ネットワーク上に散在するため、上記WWW等のハイパーテキストのデータを取得する場合には、実際に要求を出してから取得された総てのページの表示が終了するまではかなりの時間を要する。そのために短時間に指定のファイルに関連する全ページの取得/表示が終了する機能が望まれている。

【0006】そこで、このような要求を満たす機能として自動巡回機能がある。この自動巡回機能を有するクライアント装置では、取得した指定ファイルが存在するページのデータと関連ページのデータとを記憶できるようになっている。

【0007】以下、上記自動巡回機能を具体的な例を上げて説明する。

【0008】①従来例1:ソフトウェア「フリーローダ」(FreeLoader, Inc.)では、パソコン上で起動するソフトウェアであり、WWW上の指定ページを自動的に取得して記憶装置に保存するソフトウェアである。同様なソフトに「波乗野郎」(B.U.G., Inc.) 等がある。指定した時間に自動的に起動させ、予め設定したページを取得できるようになっている。

【0009】②従来例2:特願平8−299664では、予め設定された規則及びパラメータに従って、順次自動的にリンク先を辿っていく自動ウェブ巡回部を備える情報機器が提案されている。

【0010】

【発明が解決しようとする課題】ところで、上述した自動巡回機能を有するソフトウェアやや自動巡回機能を有する情報機器においては、関連度の高いページを取得したり、指定された項目のページのみを取得したり、指定されたリンク数内でページを取得するなど、ユーザが必要な情報を設定することによって取得ページ数を絞ったり、指定した時刻に起動するなど多機能を実現している。

【0011】ところが、必ずユーザがコンソールから必要事項を設定しなければいけないため、コンソールを立ち上げるという操作が必要になる。

【0012】また、設定項目の入力の際にはユーザにある程度のネットワークに関する知識が必要となるため、初心者などコンソールの操作になれないユーザには設定すら行えないという問題がある。また、自動巡回の機能を実現するためにはパソコン等のネットワーク端末を常に起動しておかなければいけないという問題がある。

【0013】そこで、本発明の目的は、ユーザが予め設定を行わなくても有効なページを効率よく自動的に取得できるWEBデータ (ウエブデータ) の取得機能をもった、ネットワーク中継装置 (ゲートウェイ) を提供することにある。

【0014】

【課題を解決するための手段】本発明のゲートウエイ装置の一態様では、WEBブラウザを搭載したネットワーク端末とネットワーク上のWEBサーバを通信可能にするゲートウェイにおいて、IPパケットフォワーディング部と、前記ネットワーク端末から入力されたWEBサイトのURL検出部と、前記URLがWEBブラウザから入力された回数を計測するアクセス頻度計測部と、前記URLとそのアクセス回数を対応付けて記憶するURL管理テーブルと、前記URL管理テーブルを用いてネットワークを介してWEBサイトのデータを自動的に取得する巡回部と、前記巡回部が取得したWEBデータを記憶するキャッシュサーバを備え、ネットワーク端末からのアクセス頻度条件に応じて、自動でWEBデータを取得する。

【0015】本発明によれば、WEBブラウザに入力されたURLと、WEBブラウザから入力された回数をURL管理テーブルで管理し、予め決められた条件でゲートウェイが自動的にWEBデータを巡回し、ゲートウェイに内蔵されたキャッシュサーバに記憶しておくことで、特にユーザからの設定を必要としないでユーザの好みに合わせたWEBデータを常に最新データに更新しておくことが出来る。また、ネットワーク端末を起動しな

くても、WEBデータの自動巡回が出来る。

【0016】また、本発明は前記URL管理テーブルが予め決められたタイムアウト時間を持つことで最後に更新したときから一定時間以上更新が無ければURL管理テーブルを更新しアクセス頻度を変更できることで、アクセスが無い場合は登録されたアクセス頻度をクリアすることができるため、過去にアクセスしたURLに影響されない。

【0017】また、本発明の他の態様では、前記URL管理テーブルを、ネットワーク端末からのアクセス頻度、更新日時順にアクセスする順番を自動、または手動で変更でき、前記巡回部は、ネットワーク端末からのアクセス頻度の高いURLから順に、予め決められた順位のURLまで巡回することで、アクセス頻度の高いURLに限ってWEBサイトのデータを更新することができる。

【0018】また、本発明の他の態様では、前記URL管理テーブルを、ネットワーク端末からのアクセス頻度、更新日時順にアクセスする順番を自動、または手動で変更でき、前記巡回部は、ネットワーク端末からのアクセス頻度の高いURLに対し、ある一定時間内にそのアクセス頻度の割合に応じた回数だけ巡回することで、アクセス頻度の高いURLは更新頻度も高くすることができる。

【0019】また、本発明の他の態様では、ゲートウェイに接続されるネットワーク端末が複数存在する場合において、ネットワーク端末を識別する識別部をさらに備え、前記URL管理テーブルおよび、前記キャッシュサーバを各ネットワーク端末ごとに管理することで、ネットワーク端末ごとにアクセス頻度が高いWEBサイトを自動巡回することが出来る。

【0020】また、本発明の他の態様では、識別部はIPアドレス、MACアドレス、ポート番号を用いて識別され、ネットワーク端末ごとにアクセス頻度の高いWEBサイトを自動巡回することが出来る。

【0021】また、本発明の他の態様では、前記巡回部により取得したHTMLのデータをCompactHTML（cHTML）に変換する変換部をさらに備え、前記キャッシュサーバに同一のWEBデータをHTMLとcHTMLの2種類のマークアップ言語で記憶しておくことで、HTML対応のWEBブラウザで閲覧したURLを元に自動巡回して取得したデータをcHTML対応WEBブラウザを搭載したネットワーク端末で閲覧することが出来る。

【0022】また、本発明の他の態様では、前記巡回部により取得したHTMLのデータをBMLに変換する変換部をさらに備え、前記キャッシュサーバに同一のWEBデータをHTMLとBMLの2種類のマークアップ言語で記憶しておくことで、HTML対応のWEBブラウザで閲覧したURLを元に自動巡回して取得したデータをBML対応WEBブラウザを搭載したテレビなどのネットワーク端末で閲覧することが出来る。

【0023】

【発明の実施の形態】以下、本発明の実施の形態について図面を参照して説明する。

【0024】（実施の形態1）図1は、本発明の自動巡回機能をもつゲートウェイの一例のブロック図であり、図2〜図4は、図1のゲートウェイ80の動作手順の例を示す図である。また、図5は、図1のゲートウエイにおけるURL管理テーブルの内容の一例を示す図であり、図6は、URL管理テーブルの他の例（タイマを考慮した例）を示し、図7は、URL管理テーブルの他の例（アクセス頻度の順位を考慮した例）を示す図である。

【0025】図1において、参照符号10はゲートウェイがもつIPパケットのフォワーディング部、21はユーザがアクセスしているWEBサイトのURL検出部、22はWEBサイトへのアクセス頻度計測部、参照符号25はアクセス監視部、30は前記URL検出部により検出したURLと前記アクセス頻度計測部により計測したアクセス頻度と関連付けて記憶するURL管理テーブル、40は複数のWEBサイトのデータを自動で取得する巡回部、50はアクセスしたデータを記憶するキャッシュサーバ、90はネットワーク、91は前記ネットワーク上にあるWEBサーバ、92はネットワーク端末に搭載されるWEBブラウザである。

【0026】図2、図3、図4を使って本発明の一実施の形態における自動巡回方法の動作を説明する。

【0027】図2は、ネットワーク端末からのWEBデータ閲覧要求に対応するWEBデータが、ゲートウェイ内のキャッシュサーバ50になかった場合の動作の仕組みである。

【0028】ネットワーク端末からWEBデータの閲覧要求がゲートウェイに対してあがったとき、ゲートウェイはネットワーク端末がアクセスしようとしているURLをURL検出部21で検出するとともに、そのアクセス頻度をアクセス頻度計測部22で計測し、URL管理テーブル30に登録する。

【0029】ゲートウェイは要求されたURLに対応するWEBデータがキャッシュサーバ50に無い場合、ネットワーク90を介してWEBサーバ91にアクセスし、最新のWEBデータを取得し、キャッシュサーバ50に記憶すると同時に、ネットワーク端末上のWEBブラウザ92にWEBデータを表示させる。

【0030】次に、図3はネットワーク端末からのWEBデータ閲覧要求に対応するWEBデータが、ゲートウェイ内のキャッシュサーバ50にあった場合の動作の仕組みである。ネットワーク端末からWEBデータの閲覧要求がゲートウェイに対してあがったとき、ネットワーク端末がアクセスしようとしているURLに対応するW

ＥＢデータがキャッシュサーバ５０にあれば、キャッシュサーバからＷＥＢデータがダウンロードされ、ネットワーク端末上のＷＥＢブラウザ９２に表示される。

【００３１】次に、図4はゲートウェイが自動巡回を行う場合の動作の仕組みである。ゲートウェイはＵＲＬ管理テーブル３０を参照し、アクセス頻度条件に従ってネットワーク９０上のＷＥＢサーバ９１にアクセスし、ＷＥＢデータを取得し、キャッシュサーバ５０に記憶する。このとき、ＵＲＬ管理テーブル３０は参照されるのみで、更新されることは無い。

【００３２】図5はゲートウェイが管理するＵＲＬ管理テーブル３０の一例である。

【００３３】図6はＵＲＬ管理テーブル３０にタイマを設け、特定のＵＲＬに対して例えば７２時間以内といったある一定期間アクセスがなければそのＵＲＬをＵＲＬ管理テーブル３０から自動的に削除する。そうすることで、ユーザの好みが変わっても過去に頻繁にアクセスしたＷＥＢサイトのデータをいつまでも自動巡回することは無くなる。

【００３４】図7はＵＲＬ管理テーブル３０にアクセス頻度の順位をつけ、例えば、アクセス頻度上位５位までといったある一定の順位までのＷＥＢサイトを自動巡回することができる。そうすることで、アクセス頻度の高いＷＥＢサイトに限って自動巡回するため、必要以上にネットワークにアクセスしなくなる。

【００３５】（実施の形態２）図8は、本実施の形態にかかるゲートウエイの構成を示す図である。本実施の形態の基本的な構成は、図１のゲートウエイと同じであるが、アクセス監視部２５において、さらに識別部２０をもつことに特徴がある。

【００３６】識別部２０は、ゲートウエイに複数のネットワーク端末が接続される場合に、各ネットワーク端末を識別する働きをする。

【００３７】したがって、図8のゲートウェイでは、ゲートウエイに接続されるネットワーク端末が複数存在する場合でも、ネットワーク端末ごとにＵＲＬ管理テーブル３０内でアクセス頻度が管理され、ネットワーク端末単位での自動巡回を行うことが可能となる。

【００３８】例えば、ネットワーク端末９３上のＷＥＢブラウザ９２を操作してユーザがＷＥＢサイトへのアクセスを行っているとき、ネットワーク端末９３用に割り当てられたＵＲＬ管理テーブル３０の領域にアクセス先のＵＲＬとアクセス頻度が記憶される。

【００３９】巡回部４０はネットワーク端末ごとにアクセス頻度上位のＵＲＬを自動巡回し、キャッシュサーバ５０にＷＥＢデータを保存する。

【００４０】ＷＥＢデータはキャッシュサーバ５０内でもネットワーク端末ごとに管理され、次回ユーザがＷＥＢブラウザ９２よりアクセスを試みた際、ポート識別部によりどのネットワーク端末からのアクセスかを判断し、キャッシュサーバ５０のデータをダウンロードし、ＷＥＢサイトの閲覧が出来る。端末の識別にはＩＰアドレス、または、Ｅｔｈｅｒｎｅｔ（登録商標）のMACアドレス、または、ポート番号が用いられる。また、ネットワーク端末間でのＵＲＬ管理テーブル、および、キャッシュサーバの参照を不可能とすれば、個人情報の保護ができる。

【００４１】図9はネットワーク端末ごとに管理されるＵＲＬ管理テーブルの一例である。図示されるように、各端末毎に区別されて、ＵＲＬが管理されている。

【００４２】（実施の形態３）図１０は、本実施の形態のゲートウエイの構成を示す図である。図１０のゲートウエイの特徴は、ＨＴＭＬを第二のマークアップ言語に変換する変換部６０を備えていることである。

【００４３】ネットワーク端末装置が自動巡回を行う際、ネットワーク端末からのアクセスによりＵＲＬとアクセス頻度を計測しＵＲＬ管理テーブル３０に登録される。

【００４４】巡回部４０はＵＲＬ管理テーブル３０からアクセス頻度の高いＵＲＬを自動巡回し、キャッシュサーバ５０にＷＥＢデータを記憶する。そのとき、ＨＴＭＬのＷＥＢデータと、ＨＴＭＬを第二のマークアップ言語に変換したＷＥＢデータの２種類をキャッシュサーバに記憶しておく。

【００４５】第二のマークアップ言語としては、ＣｏｍｐａｃｔＨＴＭＬおよび、ＢＭＬなどがある。

【００４６】次回ユーザからのアクセスがあったとき、ユーザの閲覧ツールが通常のＷＥＢブラウザ９２か、ｃＨＴＭＬ対応ＷＥＢブラウザか、ＢＭＬ対応のＷＥＢブラウザかを判断しキャッシュサーバ５０より、閲覧ツールにあわせて閲覧可能なＷＥＢデータをダウンロードして閲覧することが出来る。

【００４７】例えば、パソコンなどの頻繁にアクセスするＷＥＢサイトのデータを自動巡回し、巡回して記憶したＷＥＢサイトのデータをｉモード対応携帯電話にダウンロードして通勤時間中にＷＥＢデータを閲覧することが出来る。

【００４８】以上より明らかなように、本発明によれば、自動巡回機能を備えたゲートウェイによれば、ユーザからの入力操作を必要とせず、ユーザの好んでアクセスしているＷＥＢサイトのデータを自動巡回するという効果が得られる。以上説明した本発明のＷＥＢサイトの自動巡回方法の基本的な手順は、図１１に示すようになる。すなわち、ネットワーク端末がアクセスしたＷＥＢサイトについて、ＵＲＬおよびアクセス頻度，あるいは、ＵＲＬとアクセス頻度と直近のアクセス時点から経過した時間の情報を含む管理テーブルを作成し、かつ作成された管理テーブルをアクセスの発生状況に応じて随時更新し（ステップ１００）、管理テーブルに含まれる情報に基づいて優先度の高いＷＥＢサイトを判定して自

動巡回を行い(ステップ110)、その結果として得られた前記優先度の高いWEBサイトのデータを蓄積する(ステップ120)。

【0049】また、本発明によれば、各URLがタイマーを持ち、予め決められた時間内にユーザからのアクセスが無い場合は登録されたアクセス頻度をクリアすることで、過去に頻繁にアクセスしていたURLの影響を受けず自動巡回が出来るという効果が得られる。

【0050】また、本発明によれば、ユーザのアクセス頻度の高い順にある決められた順位までのWEBサイトのデータを自動で取得することで、ユーザがアクセスするWEBサイトが多数存在しても常にアクセス頻度上位のWEBサイトを選んで効率よく自動巡回するという効果が得られる。

【0051】また、本発明によれば、ユーザのアクセス頻度の割合に応じて自動巡回する回数を変えることで、頻繁にアクセスするWEBサイトの情報は更新される頻度も高くなるという効果が得られる。

【0052】また、本発明によれば、ネットワーク端末を識別する識別手段を備えることで、ゲートウェイに接続されるネットワーク端末が複数存在する場合でも、各ネットワーク端末ごとにアクセス頻度が高いWEBサイトを自動巡回することが出来るため、個人のプライバシーが守られるという効果が得られる。

【0053】また、HTMLのデータをcHTMLに変換する変換部を備えることで、HTMLで収集したWEBサイトのデータを、iモード対応の携帯電話などのcHTML対応WEBブラウザを搭載したネットワーク端末で閲覧できるという効果が得られる。

【0054】また、HTMLのデータをBMLに変換する変換部を備えることで、HTMLで収集したWEBサイトのデータを、BML対応WEBブラウザを搭載したテレビなどのネットワーク端末で閲覧できるという効果が得られる。

【0055】

【発明の効果】以上説明したように本発明によれば、特別な設定をしなくても、ユーザが頻繁にアクセスを繰り返しているようなウエブサイトを自動的に巡回し、必要なデータを蓄積することができ、ユーザの利便性が向上する。

【図面の簡単な説明】

【図1】本発明の実施の形態1にかかるゲートウエイの構成を示すブロック図

【図2】ゲートウエイの動作手順の一例を示す図

【図3】ゲートウエイの動作手順の他の例を示す図

【図4】ゲートウエイの動作手順の他の例を示す図

【図5】URL管理テーブルの一例を示す図

【図6】URL管理テーブルの他の例を示す図

【図7】URL管理テーブルの他の例を示す図

【図8】本発明の実施の形態2にかかるゲートウェイの構成を示すブロック図

【図9】図8のゲートウエイにおいて採用されるURL管理テーブルの内容を示す図

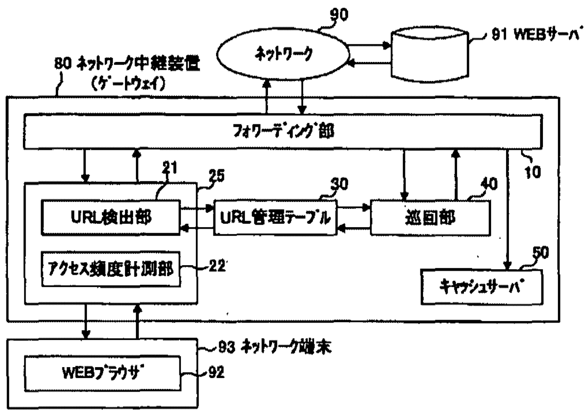
【図10】本発明の実施の形態3にかかるゲートウエイの構成を示すブロック図

【図11】本発明にかかるWEBサイトの自動巡回方法の基本的な手順を示すフロー図

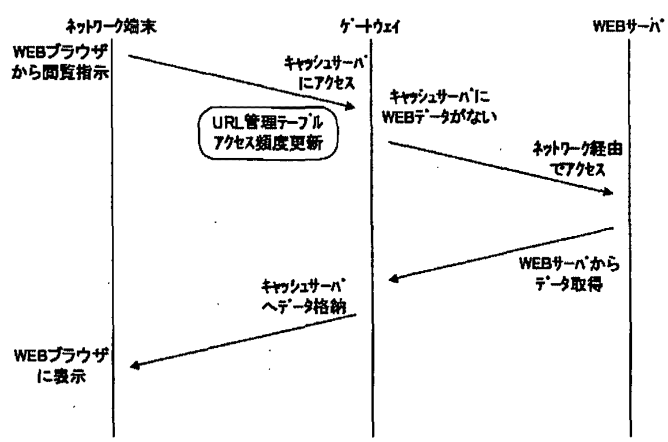
【符号の説明】

10 フォワーディング部
20 識別部
21 URL検出部
22 アクセス頻度計測部
30 URL管理テーブル
40 巡回部
50 キャッシュサーバ
60 変換部
90 ネットワーク
91 WEBサーバ
92 WEBブラウザ

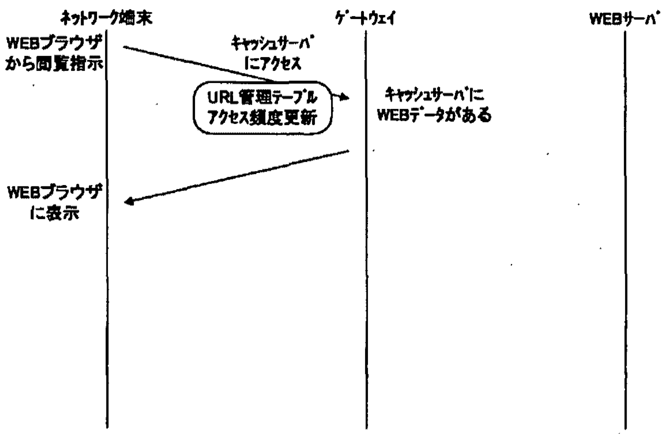
【図1】


90
ネットワーク
91 WEBサーバ
80 ネットワーク中継装置
(ゲートウェイ)
フォワーディング部
10
21
25
30
40
URL検出部
URL管理テーブル
巡回部
アクセス頻度計測部
22
50
キャッシュサーバ
93 ネットワーク端末
WEBブラウザ
92

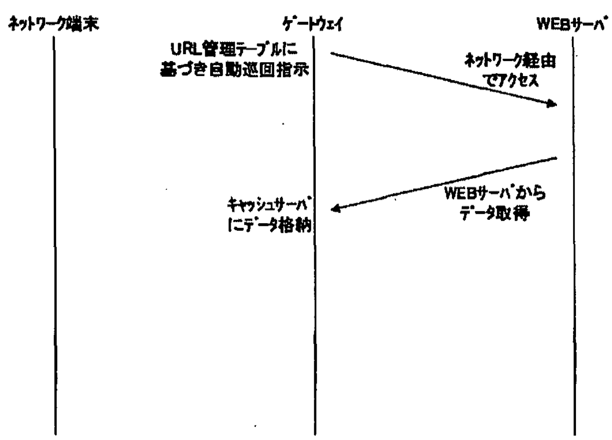
【図2】


ネットワーク端末
ゲートウェイ
WEBサーバ
WEBブラウザから閲覧指示
キャッシュサーバにアクセス
URL管理テーブルアクセス頻度更新
キャッシュサーバにWEBデータがない
ネットワーク経由でアクセス
WEBサーバからデータ取得
キャッシュサーバへデータ格納
WEBブラウザに表示

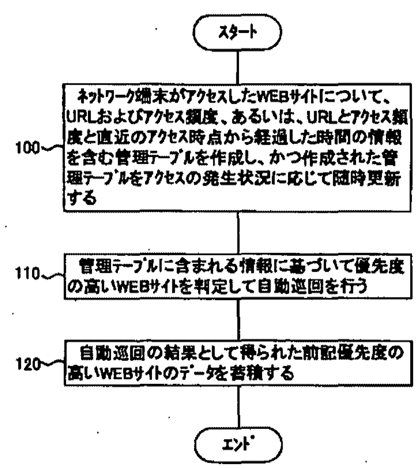
【図3】


ネットワーク端末
ゲートウェイ
WEBサーバ
WEBブラウザから閲覧指示
キャッシュサーバにアクセス
URL管理テーブルアクセス頻度更新
キャッシュサーバにWEBデータがある
WEBブラウザに表示

【図4】


ネットワーク端末
ゲートウェイ
WEBサーバ
URL管理テーブルに基づき自動巡回指示
ネットワーク経由でアクセス
WEBサーバからデータ取得
キャッシュサーバにデータ格納

【図11】


スタート
100
ネットワーク端末がアクセスしたWEBサイトについて、URLおよびアクセス頻度、あるいは、URLとアクセス頻度と直近のアクセス時点から経過した時間の情報を含む管理テーブルを作成し、かつ作成された管理テーブルをアクセスの発生状況に応じて随時更新する
110
管理テーブルに含まれる情報に基づいて優先度の高いWEBサイトを判定して自動巡回を行う
120
自動巡回の結果として得られた前記優先度の高いWEBサイトのデータを蓄積する
エンド

【図5】

URL	アクセス頻度
http://www.nsl.mci.mei.co.jp/	110
http://www.asahi.com/	90
http://www.goo.ne.jp/	70
http://www.yahoo.co.jp/	55
http://www.lycos.co.jp/	53
http://jp.excite.com/	48
http://japan.infoseek.com/	46
http://www.mainichi.co.jp/	40
⋮	⋮
http://www.yomiuri.co.jp/	10

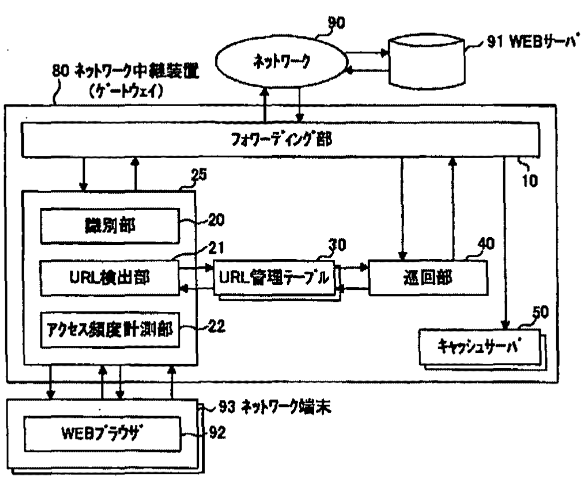
【図6】

URL	アクセス頻度(回)	タイマ(h)
http://www.nsl.mci.mei.co.jp/	110	72
http://www.asahi.com/	90	50
http://www.goo.ne.jp/	70	20
http://www.yahoo.co.jp/	55	40
http://www.lycos.co.jp/	53	45
http://jp.excite.com/	48	01
http://japan.infoseek.com/	46	04
http://www.mainichi.co.jp/	40	20
⋮	⋮	⋮
http://www.yomiuri.co.jp/	10	60

【図7】

順位	URL	アクセス頻度
1	http://www.nsl.mci.mei.co.jp/	110
2	http://www.asahi.com/	90
3	http://www.goo.ne.jp/	70
4	http://www.yahoo.co.jp/	55
5	http://www.lycos.co.jp/	53
	http://jp.excite.com/	48
	http://japan.infoseek.com/	46
	http://www.mainichi.co.jp/	40
	⋮	⋮
	http://www.yomiuri.co.jp/	10

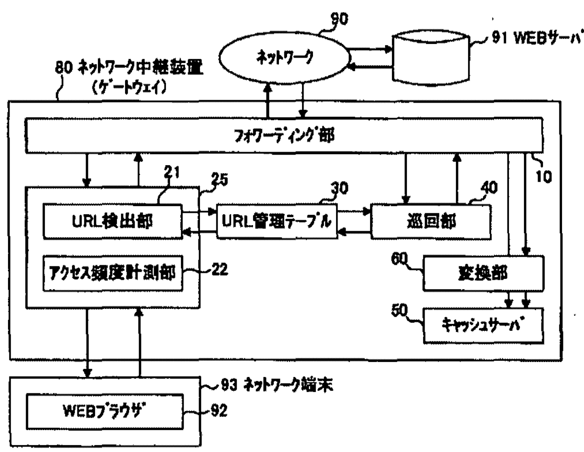
【図8】



【図9】

端末識別	URL	アクセス頻度(回)
端末1	http://www.nsl.mci.mei.co.jp/	110
端末1	http://www.asahi.com/	90
端末1	http://www.goo.ne.jp/	29
端末1	http://www.yahoo.co.jp/	11
端末2	http://www.lycos.co.jp/	53
端末2	http://jp.excite.com/	48
端末2	http://japan.infoseek.com/	46
端末2	http://www.mainichi.co.jp/	11
⋮	⋮	⋮
端末N	http://www.yomiuri.co.jp/	10

【図10】



フロントページの続き

(72)発明者 窪田 浩実
神奈川県横浜市港北区綱島東四丁目3番1号 松下通信工業株式会社内

Fターム(参考) 5B075 KK07 ND36 NR20 PR04 UU40
5B082 FA03 FA12 GC04

Our Docket/Ref. No.: APPT-001-1-1 Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): Dietz et al. Serial No.: 10/684,776 Filed: October 14, 2003 Title: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK	Group Art Unit: 2141 Examiner:

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL: INFORMATION DISCLOSURE STATEMENT

Dear Commissioner:

Transmitted herewith are:

_X_ An Information Disclosure Statement for the above referenced patent application, together with PTO form 1449 and a copy of each reference cited in form 1449.

____ A payment for petition fees.

_X_ Return postcard.

_X_ The commissioner is hereby authorized to charge payment of any missing fee associated with this communication or credit any overpayment to Deposit Account 50-0292.

A DUPLICATE OF THIS TRANSMITTAL IS ATTACHED

Date: March 4, 2004

Respectfully submitted,

Dov Rosenfeld
Attorney/Agent for Applicant(s)
Reg. No. 38687

Correspondence Address:
Dov Rosenfeld
5507 College Avenue, Suite 2
Oakland, CA 94618
Telephone No.: +1-510-54

Certificate of Mailing under 37 CFR 1.18

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Date of Deposit: March 4, 2004 Signature: Amy Drury

Our Docket/Ref. No.: APPT-001-1-1 Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): Dietz et al. Serial No.: 10/684,776 Filed: October 14, 2003 Title: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK	Group Art Unit: 2141 Examiner:

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

INFORMATION DISCLOSURE STATEMENT

Dear Commissioner:

This Information Disclosure Statement is submitted:

X under 37 CFR 1.97(b), or
(Within three months of filing national application; or date of entry of international application; or before mailing date of first office action on the merits; whichever occurs last)

___ under 37 CFR 1.97(c) together with either a:
___ Certification under 37 CFR 1.97(e), or
___ a $180.00 fee under 37 CFR 1.17(p)
(After the CFR 1.97(b) time period, but before final action or notice of allowance, whichever occurs first)

___ under 37 CFR 1.97(d) together with a:
___ Certification under 37 CFR 1.97(e), and
___ a petition under 37 CFR 1.97(d)(2)(ii), and
___ a $130.00 petition fee set forth in 37 CFR 1.17(i)(1).
(Filed after final action or notice of allowance, whichever occurs first, but before payment of the issue fee)

X Applicant(s) submit herewith Form PTO 1449-Information Disclosure Citation together with copies, of patents, publications or other information of which applicant(s) are aware, which applicant(s) believe(s) may be material to the examination of this application and for which there may be a duty to disclose in accordance with 37 CFR 1.56.

Certificate of Mailing under 37 CFR 1.18

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.
Date of Deposit: March 4, 2004 Signature: [signature]
Amy Drury

X (Cited in a related case) Each item of information contained in this information disclosure statement was first cited in a communication from the U.S. Patent and Trademark Office in a related application. The present application is related to such other applications by claiming priority of the same U.S. Provisional patent application.

It is expressly requested that the cited information be made of record in the application and appear among the "references cited" on any patent to issue therefrom.

As provided for by 37 CFR 1.97(g) and (h), no inference should be made that the information and references cited are prior art merely because they are in this statement and no representation is being made that a search has been conducted or that this statement encompasses all the possible relevant information.

Date: March 4, 2004

Respectfully submitted,

Dov Rosenfeld
Attorney/Agent for Applicant(s)
Reg. No. 38687

Correspondence Address:
Dov Rosenfeld
5507 College Avenue, Suite 2
Oakland, CA 94618
Telephone No.: +1-510-547-3378

INFORMATION DISCLOSURE STATEMENT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.
APPT-001-1-1	10/684,776
APPLICANT	
Dietz et al.	
FILING DATE	GROUP
14 Oct 2003	2141

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
	AA	6,625,657 B1	Sep. 23, 2003	Bullard	709	237	Mar. 25, 1999
	AB	6,330,226 B1	Dec. 11, 2001	Chapman et al.	370	232	Jan. 27, 1998
	AC	6,651,099 B1	Nov. 18, 2003	Dietz et al.	709	224	Jun. 30, 2000
	AD	6,424,624 B1	Jul. 23, 2002	Galand et al.	370	231	Oct. 7, 1998
	AE	6,279,113 B1	Aug. 21, 2001	Vaidya	713	201	Jun. 4, 1998
	AF	6,363,056 B1	Mar. 26, 2002	Beigi et al.	370	252	Jul. 15, 1998
	AG	6,115,393	Sep. 5, 2000	Engel et al.	370	469	Jul. 21, 1995
	AH	4,972,453	Nov. 20, 1990	Daniel, III et al.	379	10	Feb. 28, 1989
	AI	5,535,338	Jul. 9, 1996	Krause et al.	395	200.20	May 30, 1995
	AJ	5,802,054	Sep. 1, 1998	Bellenger	370	401	Aug. 16, 1996
	AK	5,720,032	Feb. 17, 1998	Picazo, Jr. et al.	395	200.2	Jan. 28, 1997

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLI-CATION DATE	COUNTRY	CLASS	SUB-CLASS	TRANS-LATION YES	NO
	AL							
	AM							

OTHER DISCLOSURES (Including Author, Title, Date, Pertinent Pages, Place of Publication, Etc.)

	AN	R. Periakaruppam and E. Nemeth. "GTrace-A Graphical Traceroute Tool." 1999 Usenix LISA. Available on www.caida.org, URL: http://www.caida.org/outreach/papers/1999/GTrace/GTrace.pdf
	AO	W. Stallings. "Packet Filtering in the SNMP Remote Monitor." November 1994. Available on www.ddj.com, URL: http://www.ddj.com/documents/s=1013/ddj9411h/9411h.htm

EXAMINER	DATE CONSIDERED

*EXAMINER: initial if citation considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to Applicant.

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

GTrace – A Graphical Traceroute Tool

Ram Periakaruppan, Evi Nemeth
University of Colorado at Boulder
Cooperative Association for Internet Data Analysis (CAIDA)
{ramanath, evi}@cs.colorado.edu

Abstract

Traceroute [Jacobson88], originally written by Van Jacobson in 1988, has become a classic tool for determining the routes that packets take from a source host to a destination host. It does not provide any information regarding the physical location of each node along the route, which makes it difficult to effectively identify geographically circuitous unicast routing. Indeed, there are examples of paths between hosts just a few miles apart that cross the entire United States and back, phenomena not immediately evident from the textual output of *traceroute*. While such path information may not be of much interest to many end users, it can provide valuable insight to system administrators, network engineers, operators and analysts. We present a tool that depicts geographically the IP path information that *traceroute* provides, drawing the nodes on a world map according to their latitude/longitude coordinates.

1. Introduction

Today's Internet has evolved into a large and complex aggregation of network hardware scattered across the globe, with resources accessed transparently with respect to their location, be it in the next room or on another continent. As the Internet becomes increasingly commercialized among many different corporate administrative entities, it is more difficult to ascertain the geographical routes that packets actually travel across the network. Knowledge of these geographical paths can provide useful insight to system administrators, network engineers, operators and analysts.

It is challenging to obtain the location for a given node of a path since there is no existing database that accurately maps hostnames or IP addresses to physical locations. Although RFC 1876 [RFC1876] defined a DNS resource record to carry such location information (the LOC record) for hosts, networks and subnets, very few sites maintain LOC records. Hence there is no straightforward way to determine the physical location of hosts.

GTrace is a graphical front end to *traceroute* that uses a number of heuristics to determine the location of a node. Often the name of a node in the path contains geographical information such as a city name/abbreviation or airport code. GTrace operates on the assumption that these codes and names indicate the physical location of the node. The locations obtained are connected together on a world map to show the geographical path that packets take from the source to destination host. GTrace also tries to verify the validity of each location obtained, eliminating ones that are incorrect.

The following sections review the *traceroute* tool and describe the design and implementation of GTrace. We also show example output from GTrace.

2. Traceroute

Traceroute is a tool that discovers the route an IP datagram takes through the Internet from a source host to a destination host. It works by exploiting the TTL (Time To Live) field of the IP Header. Each router that handles an IP datagram decrements the TTL field. When the TTL reaches zero, a router must discard the packet and send an error message to the originator of the datagram.

Traceroute uses this feature, initially sending a datagram with the TTL set to one. The first router along the path, upon receiving the datagram decrements the TTL, discards the datagram and sends back an ICMP error message. *Traceroute* records this first IP address (source address of the error message packet) and then sends the next datagram with the TTL set to two. This process continues until the datagram finally reaches the target host, or until the maximum TTL threshold is reached.

3. Design and Implementation of GTrace

Recognizing that it is not possible to obtain precise physical location information for all existing IP addresses, our main design criteria for GTrace was that it be sufficiently flexible to support the addition of new databases and heuristics. We chose to implement GTrace in Java, for both its portability and its new Swing [Swing] user interface toolkit. GTrace operates in two phases. In the first phase GTrace executes *traceroute* to the destination host and tries to determine locations for each node along the path. During the second phase, GTrace verifies whether the locations obtained in the previous phase are reasonably correct.

GTrace is composed of the following seven key components: Graphical User Interface, Dispatcher Thread, Hop Threads, Lookup Client, NetGeo Server, Lookup Server and Location Verifier. Fig. 1 illustrates the overall architecture of the tool. The function of each component is described below.

3.1 Graphical User Interface

The Main Thread handles all features of the Graphical User Interface and is responsible for spawning the dispatcher thread when a destination host is specified. Fig. 2 shows a snapshot of GTrace on startup. The GUI has two sections, with a map on the top and traditional

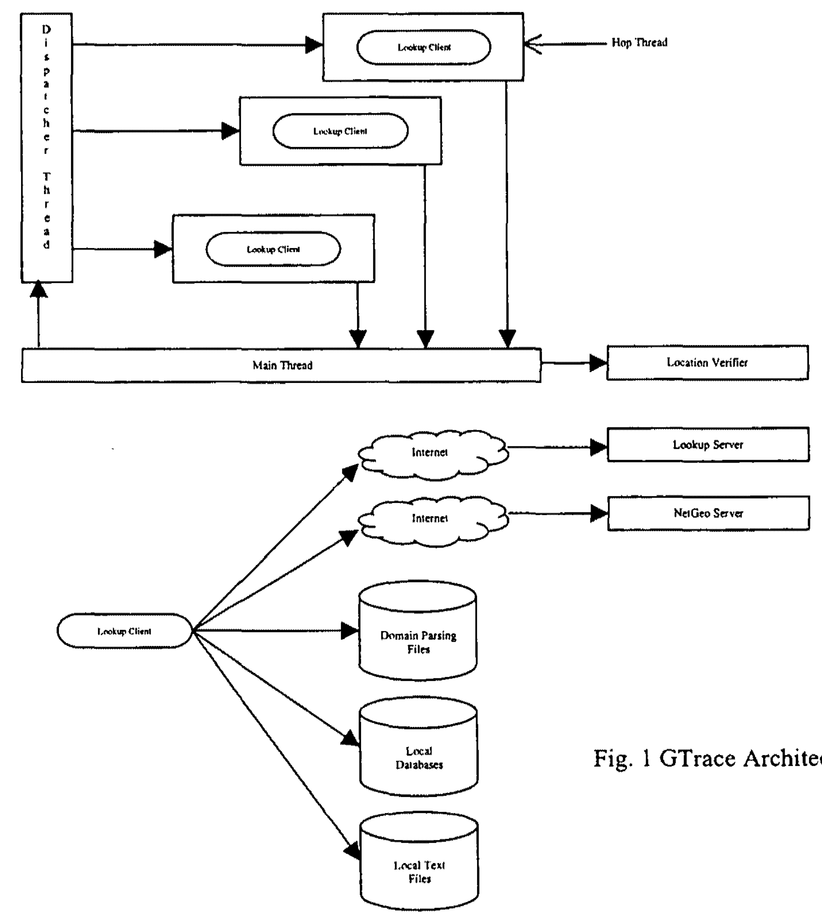


Fig. 1 GTrace Architecture

traceroute output below. The tool supports zooming in or out of particular regions of the maps. Twenty-three maps are available courtesy of VisualRoute [VisualRoute] and users can also add their own. We later provide an example that highlights some of the features of the GUI.

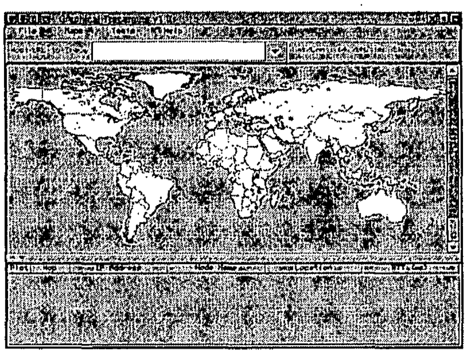


Fig. 2 GTrace's startup screen

3.2 Dispatcher Thread

The function of the dispatcher thread is to execute *traceroute* to the destination host. It then reads the output of *traceroute*, creating a new thread for each line of output. These threads are referred to as *hop threads*. The dispatcher thread can also read *traceroute* output from a file, which allows users to visualize traceroutes performed using third-party *traceroute* servers.

3.3 Hop Threads

Each hop thread parses its line of *traceroute* output and immediately notifies the main thread so that it can update the display with relevant *traceroute* fields for the corresponding hop. It then creates an instance of the Lookup Client, which tries to determine the location of the node and return the resulting information to the main thread before exiting.

3.4 Lookup Client

The Lookup Client tries to determine the location of a node by using a set of search heuristics. Many of the nodes in a typical *traceroute* path are in the ".net" domain. Often the names of these nodes have some geographical hint in them. The Lookup Client uses customized domain parsing files that specify rules for extracting these geographic hints. We have such files for several ".net" domains that use internally consistent naming conventions within their domain.

However this technique does not solve the problem of locating nodes that do not have embedded geographical hints. GTrace also utilizes databases from CAIDA [DBCAIDA] and NDG Software [DBNDG] that map hostnames and IP addresses to latitude/longitude coordinates. For nodes with no information in these databases, the Lookup Client uses the domain's registered address (unfortunately often only the headquarters for a geographically distributed infrastructure) obtained through a *whois* lookup to determine the location. Nodes for which the Lookup Client is unable to determine a location are listed in the text portion, but skipped in the geographical display.

The search algorithm is described below. We try each heuristic in turn, stopping as soon as one yields a location. The Lookup Client also makes a note of the search step that produced the location, providing this information to the user as well as the Location Verifier.

Search Algorithm:

1. Check the cache to see if the location for the IP address has already been determined from a previous trace.

2. Check if the host has a DNS LOC record. If not, reduce the hostname to the next higher level domain (i.e., remove the first component of the name) and check again for a LOC record. Continue until we have reached the last meaningful component of the name (for example foo.com in xxx.foo.com or bar.com.au in xxx.yyy.bar.com.au). Note that if a site has a LOC record for the whole domain, but machines are located outside the scope of that LOC record, GTrace would end up using incorrect data. If the Location Verifier detects such a situation, GTrace will notify the user and optionally can be configured to notify GTrace's author, who will contact the DNS administrator at the corresponding site to correct their LOC records.

3. Search for a complete match of the hostname/IP address in the databases and files specified in the GTrace configuration file.

4. If the hostname has a corresponding domain parsing file, use the rules defined in the file to extract geographical hints and proceed as indicated in the file.

5. Reduce the hostname to the next higher level domain as in step 2 and search for a match as in step 3. The process is repeated until we have reached the last meaningful component of the name.

6. Query the NetGeo [NetGeo] server with the IP address. NetGeo determines the location based on *whois* registrant information.

7. If still no match occurs and the last two letters of the hostname end in a two-letter country code, map it to the geographic center of that country.

The search algorithm is ordered in decreasing level of location reliability. Locations obtained from steps 2 and 3 are taken as authoritative, while those from step 4 onward are considered a guess. Cache entries will indicate whether the location was authoritatively determined or was a guess; this status determines the color of the lines connecting the nodes on the map.

The Lookup Client does not determine locations for IP addresses that fall in the ranges 10.0.0.0 - 10.255.255.255, 172.16.0.0 - 172.31.255.255 or 192.168.0.0 - 192.168.255.255, as these blocks are reserved for private internet use [RFC 1918]. Unfortunately some addresses in these blocks do occur in traces since some ISPs use this address space for internal router interfaces. These nodes are shown in the text portion of the display with the location marked as private internet use.

The Lookup Client queries the Lookup Server if one is defined in the GTrace configuration file and if location information has not been obtained through step 1, 2 or 3 of the search algorithm. GTrace compares the reply from the Lookup Server with any obtained previously from local lookups, with preference given to the location obtained through a lower numbered search step. Based on the GTrace configuration file, the Lookup Client also uses databases, text files and domain parsing files as follows.

Databases

The Lookup Client may need to perform lookups in many databases before determining a location. GTrace's database support is provided by the BerkeleyDB [BerkeleyDB] embedded database system, which supports a Java API that the Lookup Client uses to query the databases. The database interface allows multiple thread reads on the same database at the same time. Locking is not an issue, since Lookup Clients only read, do not write.

The following five databases are packaged with the GTrace distribution.

Machine.db [DBCAIDA]	Maps machine names to their latitude/longitude values.
Organization.db [DBCAIDA]	Maps organizations to their latitude/longitude values.
Hosts.db [DBNDG]	Maps IP addresses to their latitude/longitude values.
Cities.db [DBCAIDA]	Maps cities around the world to their latitude /longitude values.
Airport.db [AirportCodes]	Maps airport codes to their latitude/longitude values.

One can add a new database in BerkeleyDB format to GTrace with *GTraceCreateDB* and by adding an entry to the GTrace configuration file. The contents of the database ie., whether it maps hostnames, IP addresses, or both to latitude/longitude values, also have to be indicated in the configuration file. The user can also add records to existing databases using *GTraceAddRec*. *GTraceCreateDB* and *GTraceAddRec* are Java classes packaged with the GTrace distribution.

Text Files

Users may also specify new locations for nodes in text files, though it is more efficient to create a database for large data sets. New files have to be listed in the GTrace configuration file in order for the search algorithm to have access to them.

Domain Parsing files

Files describing properties of each domain are used to ferret out geographical hints embedded in hostnames. These files define parsing rules using Perl5 compatible regular expressions. GTrace uses the regular expression library from ORO Inc. [OROMatcher] for parsing. New files can be added and existing ones modified without requiring any changes to GTrace.

For example, ALTER.NET (a domain name used by UUNET, a part of MCI/WorldCom) names some of their router interfaces with three letter airport codes as shown below:

193.ATM8-0-0.GW2.EWR1.ALTER.NET
(EWR -> Newark, NJ)

190.ATM8-0-0.GW3.BOS1.ALTER.NET
(BOS -> Boston, MA)

198.ATM6-0.XR2.SCL1.ALTER.NET
(Exception)

199.ATM6-0.XR1.ATL1.ALTER.NET
(ATL -> Atlanta, GA)

Fig. 3 shows an example of a GTrace domain parsing file that would work for ALTER.NET hosts. The file first defines the regular expressions, followed by any domain specific exceptions. The exceptions are strings that match the result of the regular expressions. The user may identify the exception's location either by city or by latitude/longitude value using the format shown below:

```
exception=city,state,country
          city,country
          L: latitude, longitude
```

In the former case, the user should also use *GTraceQueryDB* to ensure that the cities database has a latitude/longitude entry for the city specified. The first line in Fig. 3 defines a substitution operation, which when matched against 193.ATM8-0-0.GW2.EWR1.ALTER.NET, would return 'EWR". The contents following the last " / " of the first line indicate what to do with a successful match, namely in this case to instruct the program to first check for a match in the data specified in the current file and then for a match in the airport database.

```
s/.*?\.([^\.]+)\d\.ALTER\.NET/$1/this,airport.db
scl=santaclara, ca, us
tco=tysonscorner, va, us
nol=neworleans, la, us
```

Fig. 3 Example of a domain parsing file for ALTER.NET.

The reason for checking the domain parsing file first is that sometimes the naming scheme for a given domain is not consistent. For example, a search for SCL obtained from 198.ATM6-0.XR2.SCL1.ALTER.NET in the airport database would return a location for Santiago de Chile. In the case of ALTER.NET, they also use three letter codes that are not airport codes but abbreviations for US cities (Fig. 3 illustrates three such abbreviations.) Note that if this exception list were not present and SCL did get mapped to Chile, the Location Verifier would likely have eliminated it using the Round Trip Time (RTT) heuristic described later, which would have recognized the RTT as much too small to get a packet to Chile and back.

Sometimes ISPs name their hosts with more than one geographical hint in them. For example VERIO.NET names some of their hosts in the following format: den0.sjc0.verio.net, which typically suggests source and destination of the interface. If there is no rule on whether the convention is to use the source or destination label first in the hostname, the rule could be defined to extract both and GTrace could use the Location Verifier's heuristics to guess.

The advantage of this technique is that one can describe an entire domain as a set of rules without needing database entries for every host in the domain. The limitation of the technique is that it will fail for domains that do not use internally consistent naming schemes.

3.5 NetGeo Server

The original design of the Lookup Client performed and parsed results of *whois* lookups directly, which required storage of a prohibitively large number of mappings of world locations to latitude/longitude values. Distributing such a large database with GTrace was not ideal. CAIDA's NetGeo [NetGeo] tool, with its ability to determine geographical locations based on the data available in *whois* records, provided a vital resource.

NetGeo is a database and collection of Perl scripts used to map IP addresses to geographical locations. Given an IP address, NetGeo will first search its own local database. If a record for the target address is found in the database, NetGeo will return the requested location information, e.g., latitude and longitude. If NetGeo finds no matching record in its database, it will perform one or more *whois* lookups until it finds a *whois* record for the appropriate network. The NetGeo Perl scripts will then parse the *whois* record and extract location information, which NetGeo both returns to the client and stores in its local database for future use.

The NetGeo database contains tables for mapping world location names (city, state/province/district, country) or US zip codes to latitude/longitude values. Most *whois* records provide enough address information for NetGeo to be able to associate some latitude/longitude value with the IP address. Occasionally the *whois* record only suggests a country or state, in which case NetGeo returns a generic latitude/longitude for that country or state. In preliminary testing, NetGeo has been able to parse addresses and find (albeit sometimes imprecise) latitude/longitude information for 89% of 17,000 RIPE *whois* records, 76% of 700 APNIC *whois* records and for more than 95% of 30,000 ARIN *whois* records.

3.6 Lookup Server

The Lookup Server handles requests from Lookup Clients and tries to determine the location of a host or IP address by executing steps 3, 4 and 5 of the search algorithm. This information is sent back to the client, which then decides whether to use the location information or not depending on the locations it might have received from other Lookup Servers or lookups it performed locally. The Lookup Client selects the location that was obtained from the lowest numbered search step.

The Lookup Server can also be requested by the Lookup Client to execute step 2 of the search algorithm. This is because not all versions of *nslookup* support queries for LOC records. GTrace tests the version of *nslookup* on the machine it is running on to determine if such a request is necessary.

3.7 Location Verifier

The Main Thread invokes the Location Verifier once all the hop threads have died and the trace is complete. The task of the Location Verifier is to check whether the locations obtained for nodes along the path are reasonable. The verifier does not determine new locations for nodes, it only indicates to the user why an existing location might be wrong and where the node could possibly be located.

The verifier algorithm is based on the fact that IP packets can not travel faster than the speed of light. Light travels across different mediums at different speeds: 3.0×10^8 m/s in vacuum, 2.3×10^8 m/s in copper and 2.0×10^8 m/s in fiber [Peterson]. GTrace uses the speed of light in copper for all of its calculations.

For each successive pair of hops that have locations, the verifier algorithm uses the deltas of the round-trip times (RTT) returned by *traceroute* to rule out locations that are physically not possible. *Traceroute* measures RTT rather than one way latency, as this would require control over both end nodes and delays are often not symmetric. Also, one must be cautious with the RTT values since they incorporate several components of delay. The RTT between two nodes has four components: the speed-of-light propagation delay, the amount of time it takes to transmit the unit of data, queuing delays inside the network and the processing time at the destination node to generate the ICMP time exceeded message. *Traceroute* typically sends 40-byte UDP datagrams, so it is safe to assume negligible transmit time. Ideally, for the verifier algorithm one would like the RTT to represent only the propagation delay, but this is not the case due to variable queuing and processing delays, hence it is not possible to set the upper bound on the RTT to a hop. Accordingly the verifier algorithm uses the minimum RTT returned by *traceroute,* as this would represent the best approximation of the propagation delay. Things are further complicated by the fact that the RTT delta between hops k and $k+1$ can be biased because the return path the ICMP packet takes from hop k can be totally different from the return path it takes from hop $k+1$. The Location Verifier tries to re-determine RTT values for hops it thinks are biased using *ping*.

By default, *traceroute* sends three datagrams each time it increments the TTL to search for the next hop. Changing the value of the q parameter in the GTrace configuration file will modify this behavior. The larger the value of q, the more accurate the estimate of the propagation delay, but large values of q also slow down GTrace as *traceroute* has to send q packets for each hop.

Knowing the geographical distance between two nodes, GTrace can calculate the time-of-flight RTT (the propagation delay at the speed-of-light in copper), compare it against *traceroute's* value and flag a problem if the RTT is smaller than physically possible. In such a case either the location of the source or of the destination or both is incorrect. The details of the verification algorithm are as follows:

Verifier Algorithm:

1. Ideally, the RTT to hop k in a path should always be less than the RTT to hop $k+1$ or $k+2$... But this is not always true due to queuing delays, asymmetric paths and other delays. We allow a 1ms fudge factor to cover such discrepancies. Thus the RTTs between hops k and $k+1$ should be such that $RTT(k) \leq RTT(k+1) + 1ms$. If this condition does not hold true then the RTT to each of the out-of-order hops preceding hop k is estimated again with *ping*, i.e. till the first hop j preceding k such that $RTT(j) \leq RTT(k+1) + 1ms$. If the RTT estimates obtained using *ping* still do not satisfy the condition $RTT(k) \leq RTT(k+1) + 1ms$, then hop k is not used in the later stages of the verifier algorithm.

2. Cluster the *traceroute* path into regions having similar RTT values. This is based on the assumption that nodes with similar RTTs will tend to be in the same geographic region.

3. For each region identified in the previous step, calculate the time-of-flight RTT for pairs of hops that have locations. If the RTT delta reported by *traceroute* for that pair of hops is smaller than the time-of-flight RTT, flag the pair of hops so that it is corrected in step 5.

4. Repeat step 3 for hops falling on the edges of adjacent regions.

5. Try to "correct" unreasonable location values that were identified in steps 3 and 4 using the reliability of the search step that produced the location match. Adjacent nodes between regions are corrected first because they represent larger and probably more inaccurate locations. Correcting the nodes identified in step 3 follows this. By correct, we mean trying different alternatives for the incorrect location based on the cluster in which it falls, flagging it to the user and not plotting it in the display.

Example:

Consider the trace shown in Fig. 4, where locations are expressed as city names for ease of illustration. The *Search Step* column indicates which step of the search algorithm produced the location for that hop. Step 1 of the verifier algorithm would mark hop 13 as unusable since its RTT is greater than its subsequent hops. In this case it is probably due to the return path from hop 13 being longer than that from hop 14. Next, step 2 of the algorithm would cluster the *traceroute* path into the following regions: 1-4, 5, 6-8, 9-10, 11-12 and 14-16. Step 3 would flag that there is a problem between hops 7 and 8 since it is not possible for a packet to travel from San Francisco to New Jersey in less than a millisecond. Likewise, step 4 would flag a problem between hops 10 and 11. Step 5 would first try to correct hops 10 and 11 since they fall in different regions. Seeing that the location for hop 11 was obtained through step 3 of the search algorithm and hop 10 was from a higher step, the Location Verifier would change hop 10's location to that of hop 11's, in this example to Washington and rerun the algorithm from step 3. This process is repeated until all locations from one hop to the next are physically realistic. In the end the Location Verifier would have indicated to the user that hop 8 is incorrect and is most probably located somewhere near San Francisco. Hops 9 and 10 are also incorrect and may be in Washington with their interfaces labeled San Francisco to

Hop	Node Name	IP Address	Search Step	Location	RTT (ms)
1	pinot-fe2-0-0	(192.172.226.65)	6	San Diego	0.917ms
2	medusa.sdsc.edu	(198.17.46.10)	3	San Diego	0.881ms
3	sdsc-gw.san-bb1.cerf.net	(192.12.207.9)	4	San Diego	1.944 ms
4	pos0-0-155M.san-bb6.cerf.net	(134.24.29.130)	4	San Diego	4.640 ms
5	atm6-0-1-622M.lax-bb4.cerf.net	(134.24.29.142)	4	Los Angeles	9.598 ms
6	pos6-0-622M.sfo-bb3.cerf.net	(134.24.29.233)	4	San Francisco	15.317 ms
7	pos10-0-0-155M.sfo-bb1.cerf.net	(134.24.32.86)	4	San Francisco	16.813 ms
8	192.205.31.29	(192.205.31.29)	6	New Jersey	16.917 ms
9	att-gw.sf.cw.net	(192.205.31.78)	4	San Francisco	81.281 ms
10	corerouter2.SanFrancisco.cw.net	(204.70.9.132)	4	San Francisco	81.254 ms
11	core1.Washington.cw.net	(204.70.4.129)	3	Washington	89.727 ms
12	mix1-fddi-0.Washington.cw.net	(204.70.2.14)	4	Washington	89.708 ms
13	**vsnlpoone.Washington.cw.net**	**(204.189.152.134)**	**4**	**Poone**	**706.301 ms**
14	202.54.6.17	(202.54.6.17)	6	Madras	697.946 ms
15	202.54.6.254	(202.54.6.254)	6	Madras	702.893 ms
16	giasmda.vsnl.net.in	(202.54.6.161)	4	Madras	704.856 ms

Fig. 4 A sample *traceroute* output produced by the first phase of GTrace.

identify the other end of that link.

4. Configuration Files

The configuration options in GTrace are quite flexible. How it functions and executes the search algorithm depends on the contents of two configuration files: *GTrace.conf* and *GTraceMaps.conf*

4.1 GTrace.conf

GTrace.conf specifies the location of the commands GTrace uses and lists databases, text files, Lookup Servers if any, to use in the search algorithm. Fig. 5 shows an example configuration file. This file is automatically generated by the configure scripts while installing GTrace.

4.2 GTraceMaps.conf

The *GTraceMaps.conf* configuration file specifies attributes of the maps that GTrace uses in displays. Users can add their own maps as part of or independent from the existing world hierarchy. Independent maps allow users to describe their own intranet topology and then use GTrace as a graphical debugging tool within their network.

```
#GTrace configuration file

#Paths
TRACEROUTE=/usr/sbin/traceroute –q 3
WHOIS=/usr/bin/whois
PING= /usr/sbin/ping
NSLOOKUP=/usr/sbin/nslookup
DOMAINFILES=/home/ram/gtrace/data
DATABASES=/home/ram/gtrace/db

#Names of databases and text files to be used
#for location lookups. Order is important, list
#them in the order they should be searched.
CITIES=cities.db
AIRPORTS=airport.db

HOSTSLOC=Machine.db,hostnames/ipaddr;
            Hosts.db,ipaddr;
            Organization.db,hostnames/ipaddr;

TEXTFILES=England.txt,hostnames/ipaddr;


#Location of Lookup Servers if any
LOOKUPSRVS=
```

Fig. 5 Sample *GTrace.conf* file

5. GTrace Features

Fig. 6 shows an example of a trace that was executed from University of Colorado, Boulder to CAIDA in San Diego. On the display, the colors of the lines on the map indicate the

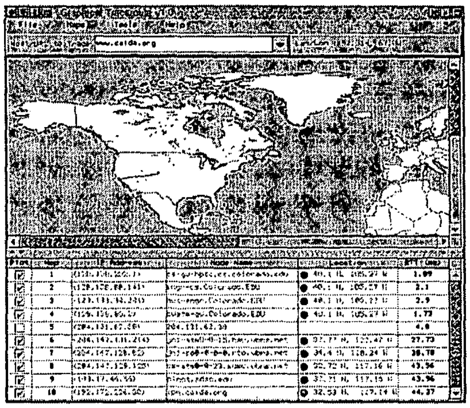


Fig. 6 Example of a trace produced by GTrace

reliability of the location obtained for the endpoints. The colors are decided based on the following criteria:

Green	Both endpoints are authoritative locations.
Yellow	One endpoint is authoritative and the other is a guess whose location is not a country center, state center or obtained from a *whois* record.
Blue	Both endpoints are guesses and the locations of both the endpoints are not a country center, state center or obtained from a *whois* record.
Red	One endpoint is a location that is a country center, state center or obtained from a *whois* record.

The table in the lower section of the display consists of six columns. The first column provides the user with a checkbox that is enabled for each location plotted on the map. The user can disable a checkbox and the corresponding location will be skipped. Locations that are flagged as unreasonable by the Location Verifier are not plotted by default.

The second, third and fourth columns display the hop number, IP address and host name respectively. Clicking on columns three and four will bring up *whois* information for the node.

Column five provides the latitudes and longitudes obtained for each hop. Clicking on this column will provide an explanation of how the location was determined and whether the Location Verifier detected any problems. A small colored ball in front of the latitude and longitude value indicates which search step produced the location. The colors and the search step they represent are given below:

Green	Step 2 LOC record.
Yellow	Step 3 Complete match
Blue	Step 4 Domain parsing file
Cyan	Step 5 Hostname reduction match
Red	Step 6 *whois* record
Gray	Step 7 Country code

The last column shows the smallest of the round trip times returned by *traceroute*. The color of the value indicates how many packets timed out: black implies that no packets timed out, blue implies that one packet timed out, and a value in red indicates that two or more packets timed out.

6. Using GTrace in the Local Environment

System Administrators often use *traceroute* as a debugging tool to identify problems in their network. GTrace provides a visual representation that can facilitate understanding and debugging of their network. It can be used to discover routing loops as well as for deciding routes. For example in a large campus if a path from host A to host B (located in the same building) goes across campus and back, the routing could be fixed to avoid such inefficient paths. GTrace can also be useful from an end user perspective. Students can use the tool to work out the topology of their campus network.

7. Conclusion

GTrace is a handy tool for identifying network topology and routing problems as well as gaining more macroscopic insight into the Internet infrastructure. While GTrace uses several heuristics to determine locations and its approach does not guarantee accuracy, it is robust and extensible. New databases, new Lookup Servers and learned insights into ISP's naming conventions can easily be added to GTrace. We hope that users and system administrators will find GTrace useful and contribute their own domain parsing files, or even run their own Lookup Servers for community use.

The practical success of GTrace lies in the rules defined for the ".net" domains, since these comprise the majority of hops in many traceroutes. Looking up a ".net" name in the *whois* database is only useful for small localized ISPs. Relying on *whois* heuristics would result in backbone providers' ".net" nodes to all uselessly map to a single corporate headquarters for that provider.

The accuracy of this tool would be much improved if the Internet community maintained LOC records in the DNS. Unfortunately since LOC records are optional, non-trivial in effort to support and without any clear payoff to ISPs, pervasive use of them will probably never occur and geographic visualization of arbitrary Internet infrastructure will continue to require heuristics to determine physical location of nodes.

8. Acknowledgments

We would like to thank kc claffy at CAIDA for suggesting the idea to develop this tool. We would also like to mention a special word of thanks to the following people and institutions: VisualRoute for permission to use their maps and labels, Sleepycat Software for the BerkeleyDB Package, Jim Donohoe for developing NetGeo and to the entire research team at CAIDA who helped with many aspects during the development of GTrace.

Several students (Colorado: Robert Cooksey, Brent Halsey, Jamey Wood, Jeremy Bargen and UCSD: Jim Anderson) wrote graphical *traceroute* tools as class projects in Evi Nemeth's Network System's class. Many good ideas from these students' projects were incorporated into GTrace.

9. Availability and Support

GTrace-1.0 is the current release and it can be downloaded from the GTrace home page at *http://www.caida.org/Tools/GTrace*. The source code comes with the GTrace distribution. Further information on using the tool or how you can contribute domain parsing files can be found on the GTrace home page.

10. Author Information

Ram Periakaruppan is pursuing his Master's degree in Computer Science at the University of Colorado, Boulder. He can be reached at <*ramanath@cs.colorado.edu*>.

Evi Nemeth has been a computer science faculty member at the University of Colorado for years. Currently she is on leave doing the IEC (Internet Engineering Curriculum) project at CAIDA (Cooperative Association for Internet Data Analysis) on the UCSD campus and working furiously to make the publisher's deadline for the third edition of the UNIX System Administration Handbook. She can be reached at <*evi@cs.colorado.edu*>.

References

[AirportCodes] Listing of Airport Codes, http://www.mapping.com/airportcodes.html

[BerkeleyDB] BerkeleyDB Package Distribution, http://www.sleepycat.com

[DBCAIDA] Database files compiled by CAIDA, http://www.caida.org/NetGeo/NetGeo/

[DBNDG] Database file compiled by NDG Software, http://www.dtek.chalmers.se/~d3augus t/xt/dl/

[Jacobson88] Van Jacobson, Traceroute source code and documentation. Available from: ftp://ftp.ee.lbl.gov/traceroute.tar.Z.

[NetGeo] The Internet Geographic Database, http://www.caida.org/Tools/NetGeo

[OROMatcher] OROMatcher - Regular Expression Package for Java, http://www.savarese.org

[Peterson] Peterson, Larry L., & Davie, Bruce S., Computer Networks - A Systems Approach, Morgan Kaufmann, (1996).

[RFC1876] RFC 1876, Davis, C., Vixie, P., Goodwin, T., and Dickinson I., A means for Expressing Location Information in the Domain Name System, January (1996).

[RFC1918] RFC 1918, Rekhter, Y., Moskowitz, B., Karrenberg, D., Groot, G. J., Lear E., Address Allocation for Private Internets, February (1996).

[Swing] Java Foundation Classes – Swing http://java.sun.com/products/jfc/

[VisualRoute] Maps from VisualRoute, http://www.visualroute.com

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

Dr. Dobb's Software Tools for the Professional Programmer

Packet Filtering in the SNMP Remote Monitor

Controlling remote monitors on a LAN

William Stallings

William is president of Comp-Comm Consulting of Brewster, MA. This article is based on his recent book, SNMP, SNMPv2, and CMIP: The Practical Guide to Network Management Standards (Addison-Wesley, 1993). He can be reached at stallings@acm.org.

The Simple Network Management Protocol (SNMP) architecture was designed for managing complex, multivendor internetworks. To achieve this, a few *managers* and numerous *agents* scattered throughout the network must communicate. Each agent uses its own management-information database (MIB) of managed objects to observe or manipulate the local data available to a manager.

The remote-network monitoring (RMON) MIB, defined as part of the SNMP framework, provides a tool that an SNMP or SNMPv2 manager can use to control a remote monitor of a local-area network. The RMON specification is primarily a definition of a data structure containing management information. The effect, however, is to define standard network-monitoring functions and interfaces for communicating between SNMP-based management consoles and remote monitors. In general terms, the RMON capability provides an efficient way of monitoring LAN behavior, while reducing the burden on both other agents and management stations; see Figure 1.

The accompanying text box entitled, "Abstract Syntax Notation One (ASN.1)" gives details on defining the communication formats between agents and managers.

The key to using RMON is the ability to define "channels"--subsets of the stream of packets on a LAN. By combining various filters, a channel can be configured to observe a variety of packets. For example, a monitor can be configured to count all packets of a certain size or all packets with a given source address.

To use RMON effectively, the person responsible for configuring the remote monitor must understand the underlying filter and channel logic used in setting it up. In this article, I'll examine this filter and channel logic.

The RMON MIB contains variables that can be used to configure a monitor to observe selected packets on a particular LAN. The basic building blocks are a data filter and a status filter. The data filter allows the monitor to screen observed packets based on whether or not a portion of the packet matches a certain bit pattern. The status filter allows the monitor to screen observed packets on the basis of their status (valid, CRC error, and so on). These filters can be combined using logical AND and OR operations to form a complex test to be applied to incoming packets. The stream of packets that pass the test is referred to as a "channel," and a count of such packets is maintained. The channel can be configured to generate an alert if a packet passes through the channel when it is in an enabled state. Finally, the packets passing through a channel can be captured in a buffer. The logic defined for a single channel is quite complex. This gives the user enormous flexibility in defining the stream of packets to be counted.

Filter Logic

At the lowest level of the filter logic, a single data or status filter defines characteristics of a packet. First, consider the logic for defining packet characteristics using the variables *input* (the incoming portion of a packet to be filtered), *filterPktData* (the bit pattern to be tested for), *filterpktDataMask* (the relevant bits to be tested for), and *filterPktDataNotMask* (which indicates whether to test for a match or a mismatch). For the purposes of this discussion, the logical operators AND, OR, NOT, XOR, EQUAL, and NOT-EQUAL are represented by the symbols o, +, --, _, =, and _, respectively.

Suppose that initially, you simply want to test the input against a bit pattern for a match. This could be used to screen for packets with a specific source address, for example. In the expression in Example 1(a), you would take the bit-wise exclusive-OR of *input* and *filterPktData*. The result has a 1 bit only in those positions where *input* and *filterPktData* differ. Thus, if the result is all 0s, there's an exact match. Alternatively, you may wish to test for a mismatch. For example, suppose a LAN consists of a number of workstations and a server. A mismatch test could be used to screen for all packets that did not have the server as a source. The test for a mismatch would be just the opposite of the test for a match; see Example 1(b). A 1 bit in the result indicates a mismatch.

The preceding tests assume that all bits in the input are relevant. There may, however, be some "don't-care" bits irrelevant to the filter. For example, you may wish to test for packets with any multicast destination address. Typically, a multicast address is indicated by one bit in the address field; the remaining bits are irrelevant to such a test. The variable *filterPktDataMask* is introduced to account for "don't-care" bits. This variable has a 1 bit in each relevant position and 0 bits in irrelevant positions. The tests can be modified; see Example 1(c).

The XOR operation produces a result that has a 1 bit in every position where there is a mismatch. The AND operation produces a result with a 1 bit in every relevant position where there is a mismatch. If all of the resulting bits are 0, then there is an exact match on the relevant bits; if any of the resulting bits is 1, there is a mismatch on the relevant bits.

Finally, you may wish to test for an input that matches in certain relevant bit positions and mismatches in others. For example, you could screen for all packets that had a particular host as a destination (exact match of the DA field) and did not come from the server (mismatch on the SA field). To enable these more complex tests to be performed, use *filterPktDataNotMask*, where:

- The 0 bits in *filterPktDataNotMask* indicate the positions where an exact match is required between the relevant bits of *input* and *filterPktData* (all bits match).
- The 1 bits in *filterPktDataNotMask* indicate the positions where a mismatch is required between the relevant bits of *input* and *filterPktData* (at least one bit does not match).

For convenience, assume the definition in Example 2(a). Incorporating *filterPktDataNotMask* into the test for a match gives Example 2(b).

The test for a mismatch is slightly more complex. If all of the bits of *filterPktDataNotMask* are 0 bits, then no mismatch test is needed. By the same token, if all bits of *filterPktDataNotMask* are 1 bits, then no match test is needed. However, in this case, *filterPktDataNotMask* is all 0s, and the match test automatically passes *relevant_bits_differento*0=0. Therefore, the test for mismatch is as in Example 2(c).

The logic for the filter test is summarized in Figure 2. If an incoming packet is to be tested for a bit pattern in a portion of the packet, located at a distance *filterPktDataOffset* from the start of the packet, the following tests will be performed:

- Test #1: As a first test (not shown in the figure), the packet must be long enough so that at least as many bits in the packet follow the offset as there are bits in *filterPktData*. If not, the packet fails this filter.
- Test #2: Each bit set to 0 in *filterPktDataNotMask* indicates a bit position in which the relevant bits of the packet portion should match *filterPktData*. If there is a match in every desired bit position, then the test passes; otherwise the test fails.
- Test #3: Each bit set to 1 in *filterPktDataNotMask* indicates a bit position in which the relevant bits of the packet portion should not match *filterPktData*. In this case, the test is passed if there is a mismatch in at least one desired bit position.

A packet passes this filter if and only if it passes all three tests.

Why use the filter test? Consider that you might want to accept all Ethernet packets that have a destination address of "A5"h but do not have a source address of "BB"h. The first 48 bits of the Ethernet packet constitute the destination address and the next 48 bits, the source address. Example 3 implements the test. The variable *filterPktDataOffset* indicates that the pattern matching should start with the first bit of the packet; *filter PktData* indicates that the pattern of interest consists of "A5"h in the first 48 bits and "BB"h in the second 48 bits; *filter PktDataMask* indicates that the first 96 bits are relevant; and *filterPktDataNotMask* indicates that the test is for a match on the first 48 bits and a mismatch on the second 48 bits.

The logic for the status filter has the same structure as that for the data filter; see Figure 2. For the status filter, the reported status of the packet is converted into a bit pattern. Each error-status condition has a unique integer value, corresponding to a bit position in the status-bit pattern. To generate the bit pattern, each error value is raised to a power of 2 and the results are added. If there are no error conditions, the status-bit pattern is all 0s. An Ethernet interface, for example, has the error values defined in Table 1. Therefore, an Ethernet fragment would have the status value of 6(21+22).

Channel Definition

A channel is defined by a set of filters. For each observed packet and each channel, the packet is passed through each of the filters defined for that channel. The way these filters are combined to determine whether a packet is accepted for a channel depends on the value of an object associated with the channel (*channelAcceptType*), which has the syntax *INTEGER {acceptMatched(1), acceptFailed(2)}*.

If the value of this object is *acceptMatched(1)*, packets will be accepted for this channel if they pass both the packet-data and packet-status matches of at least one associated filter. If the value of this object is *acceptFailed(2)*, packets will be accepted to this channel only if they fail either the packet-data match or the packet-status match of every associated filter.

Figure 3 illustrates the logic by which filters are combined for a channel whose accept type is *acceptMatched*. A filter is passed if both the data filter and the status filter are passed; otherwise, that filter has failed. If you define a pass as a logical 1 and a fail as a logical 0, then the result for a single filter is the AND of the data filter and status filter for that filter. The overall result for a channel is then the OR of all the filters. Thus, a packet is accepted for a channel if it passes at least one associated filter pair for that channel.

If the accept type for a channel is *acceptFailed*, then the complement of the function just described is used. That is, a packet is accepted for a channel only if it fails every filter pair for that channel. This would be represented in Figure 3 by placing a NOT gate after the OR gate.

Channel Operation

The value of *channelAcceptType* and the set of filters for a channel determine whether a given packet is accepted for a channel or not. If the packet is accepted, then the counter *channelMatches* is incremented. Several additional controls are associated with the channel: *channelDataControl*, which determines whether the channel is on or off; *channelEventStatus*, which indicates whether the channel is enabled to generate an event when a packet is matched; and *channelEventIndex*, which specifies an associated event.

When *channelDataControl* has the value off, then, for this channel, no events may be generated as the result of packet acceptance, and no packets may be buffered. If *channelDataControl* has the value on, then these related actions are possible.

Figure 4 summarizes the channel logic. If *channelDataControl* is on, then an event will be generated if: 1. an event is defined for this channel in *channelEventIndex*; and 2. *channelEventStatus* has the value *eventReady* or *eventAlwaysReady*. If the event status is *eventReady*, then each time an event is generated, the event status is changed to *eventFired*. It then takes a positive action on the part of the management station to reenable the channel. This mechanism can therefore be used to control the flow of events from a channel to a management station. If the management station is not concerned about flow control, it may set the event status to *eventAlwaysReady*, where it will remain until explicitly changed.

Summary

The packet-filtering facility of RMON provides a powerful tool for the remote monitoring of LANs. It enables a monitor to be configured to count and buffer packets that pass or fail an elaborate series of tests. This facility is the key to successful remote-network monitoring.

Abstract Syntax Notation One (ASN.1)

Steve Witten

Steve, a software engineer for Hewlett-Packard, specializes in network testing and measurement. You can contact him at stevewi@hpspd.spd.hp.com.

SNMP protocol and MIB are formally defined using an abstract syntax. This allowed SNMP's authors to define data and data structures without regard to differences in machine representations. This abstract syntax is an OSI language called "abstract syntax notation one" (ASN.1). It is used for defining the formats of the packets exchanged by the agent and manager in the SNMP protocol and is also the means for defining the managed objects.

ASN.1 is a formal language defined in terms of a grammar. The language itself is defined in ISO #8824. The management framework defined by the SNMP protocol, the SMI, and the MIB use only a subset of ASN.1's capabilities. While the general principles of abstract syntax are good, many of the bells and whistles lead to unnecessary complexity. This minimalist approach is taken to facilitate the simplicity of agents.

Listings One through Three show an MIB, using a fictitious enterprise called SNMP Motors. Listing One is an ASN.1 module that contains global information for all MIB modules. Listing Two, another ASN.1 module, contains the definitions of specific MIB objects. Finally, Listing Three illustrates manageable objects.

Once data structures can be described in a machine-independent fashion, there must be an unambiguous way of transmitting those structures over the network. This is the job of the transfer-syntax notation. Obviously, you could have several transfer-syntax notations for an abstract syntax, but only one abstract-syntax/transfer-syntax pair has been defined in OSI. The basic encoding rule (BER) embodies the transfer syntax. The BER is simply a recursive algorithm that can produce a compact octet encoding for any ASN.1 value.

At the top level, the BER describes how to encode a single ASN.1 type. This may be a simple type such as an *Integer*, or an arbitrarily complex type. The key to applying the BER is understanding that the most complex ASN.1 type is nothing more than several simpler ASN.1 types. Continuing the decomposition, an ASN.1 simple type (such as an *Integer*) is encoded.

Using the BER, each ASN.1 type is encoded as three fields: a tag field, which indicates the ASN.1 type; a length field, which indicates the size of the ASN.1 value encoding which follows; and a value field, which is the ASN.1 value encoding.

Each field is of variable length. Because ASN.1 may be used to define arbitrarily complex types, the BER must be able to support arbitrarily complex encodings.

It is important to note how the BER views an octet. Each octet consists of eight bits. BER numbers the high-order (most significant) bit as bit 8 and the low-order (least significant) bit as bit 1. It's critical that this view be applied consistently because different machine architectures use different ordering rules.

Figure 1 RMON description.

Figure 2 Logic for the filter test.

Figure 3 Logic by which filters are combined for a channel whose accept type is acceptMatched.

Figure 4: Logic for channel filter.

```
procedure packet_data_match;
begin
  if (result = 1 and channelAcceptType = acceptMatched) or
      (result = 0 and channelAcceptType = acceptFailed)
  then begin
      channelMatches := channelMatches + 1;
      if channelDataControl = on
      then begin
              if (channelEventStatus _ eventFired) and
                 (channelEventIndex _ 0) then generate_event;
              if (channelEventStatus = eventReady) then
                  channelEventStatus := eventFired
              end;
       end;
```

```
end;
```

Example 1: Testing the input against a bit pattern for a match.

```
(a) (input XOR filterPktData) = 0 --> match

(b) (input XOR filterPktData) (does not equal) 0 --> mismatch

(c) ((input XOR filterPktData) (and) filterPktDataMask) =
      0 --> match on relevant bits
    ((input XOR filterPktData) (and) filterPktDataMask) (does not equal)
      0 --> mismatch on relevant bits
```

Table 1: Ethernet-interface error values.

Bit	Error
0	Packet is longer than 1518 octets.
1	Packet is shorter than 64 octets.
2	Packet experienced a CRC or alignment error.

Example 2: Assuming the definition in (a), incorporating filterPktDataNotMask into the test for a match, you end up with (b). Test for a mismatch is shown in (c).

```
(a) relevant_bits_different =
      (input XOR filterPktData) (and) filterPktDataMask

(b) (relevant_bits_different (and) filterPktDataNotMask') =
      0 --> successful match

(c) ((relevant_bits_different (and) filterPktDataNotMask) (does not equal) 0) +
      (filterPktDataNotMask = 0) --> successful mismatch
```

Example 3: Launching a filter test.

```
filterPktDataOffset    = 0
filterPktData          = "00 00 00 00 00 A5 00 00 00 00 00 BB"h
filterPktDataMask      = "FF FF FF FF FF FF FF FF FF FF FF FF"h
filterPktDataNotMask   = "00 00 00 00 00 00 FF FF FF FF FF FF"h
```

Listing One

```
SNMP-motors-MIB DEFINITIONS ::= BEGIN
IMPORTS
    enterprises
        FROM RFC1155-SMI;
SNMP-motors OBJECT IDENTIFIER ::= { enterprises 9999 }
expr        OBJECT IDENTIFIER ::= { SNMP-motors 2 }
END
```

Listing Two

```
SNMP-motors-car-MIB DEFINITIONS ::= BEGIN
IMPORTS
    SNMP-motors
        FROM SNMP-motors-MIB;
IMPORTS
```

```
    OBJECT TYPE, ObjectName, NetworkAddress,
    IpAddress, Counter, Gauge, TimeTicks, Opaque
        FROM RFC1155-SMI;
car OBJECT IDENTIFIER ::= { SNMP-motors 3 }
-- this is a comment
-- Implementation of the car group is mandatory
-- for all SNMP-motors cars.
-- ( the rest of the SNMP-motors-car-MIB module )
END
```

Listing Three

```
carName OBJECT TYPE
    SYNTAX DisplayString (SIZE (0..64))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "A textual name of the car."
    ::= { car 1 }
carLength OBJECT TYPE
    SYNTAX INTEGER (0..100)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The length of the car in feet."
    ::= { car 2 }
carPassengers OBJECT TYPE
    SYNTAX INTEGER (0..4)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The number of passengers in the car."
    ::= { car 3 }
carPassengerTable OBJECT TYPE
    SYNTAX SEQUENCE OF CarPassengerEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A table describing each passenger."
    ::= { car 4 }
carPassengerEntry OBJECT TYPE
    SYNTAX SEQUENCE OF CarPassengerEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "A entry table describing each passenger."
    ::= { carPassengerTable 1 }
CarPassengerEntry ::= SEQUENCE {
    carPindex
        INTEGER,
    carPname
        DisplayString,
    carPstatus
        INTEGER
}
carPindex OBJECT TYPE
    SYNTAX INTEGER (1..4)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Index for each passenger which ranges from
                  1 to the value of carPassengers."
    ::= { carPassengerEntry 1 }
carPname OBJECT TYPE
```

```
    SYNTAX DisplayString (SIZE (0..64))
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "The name of the passenger."
    ::= { carPassengerEntry 2 }
carPstatus OBJECT TYPE
    SYNTAX INTEGER { other(1),driver(2) }
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        "The status of the passenger."
    ::= { carPassengerEntry 3 }
```

Freeform Search

Database:
- US Pre-Grant Publication Full-Text Database
- US Patents Full-Text Database
- US OCR Full-Text Database
- EPO Abstracts Database
- JPO Abstracts Database
- Derwent World Patents Index
- IBM Technical Disclosure Bulletins

Term: L3 same packet

Display: 10 **Documents in Display Format:** TI **Starting with Number** 1

Generate: ○ **Hit List** ◉ **Hit Count** ○ **Side by Side** ○ **Image**

Search Clear Interrupt

Search History

DATE: Friday, October 01, 2004 Printable Copy Create Case

Set Name side by side	**Query**	**Hit Count**	**Set Name** result set
DB=USPT; PLUR=YES; OP=ADJ			
L4	L3 same packet	20	L4
L3	l1 same L2	46	L3
L2	new adj1 flow	1021	L2
L1	(present or exist$3) adj2 flow	5136	L1

END OF SEARCH HISTORY

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/684,776	10/14/2003	Russell S. Dietz	APPT-001-1-1	3352

21921 7590 10/05/2004

DOV ROSENFELD
5507 COLLEGE AVE
SUITE 2
OAKLAND, CA 94618

EXAMINER	
MEKY, MOUSTAFA M	
ART UNIT	**PAPER NUMBER**
2157	

DATE MAILED: 10/05/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

Office Action Summary	Application No.	Applicant(s)	
	10/684,776	DIETZ ET AL.	
	Examiner	**Art Unit**	
	Moustafa M Meky	2157	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *14 October 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *11-59* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *11-59* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *12/17/03 & 3/8/04*.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

1. Claims 11-59 are presenting for examination.

2. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless -
>
> (e) the invention was described in a patent granted on an application for patent by another filed in the United States before the invention thereof by the applicant for patent, or on an international application by another who has fulfilled the requirements of paragraphs (1), (2), and (4) of section 3718 of this title before the invention thereof by the applicant for patent.

The changes made to 35 U.S.C. 102(e) by the American Inventors Protection Act of 1999 (AIPA) do not apply to the examination of this application as the application being examined was not (1) filed on or after November 29, 2000, or (2) voluntarily published under 35 U.S.C. 122(b). Therefore, this application is examined under 35 U.S.C. 102(e) prior to the amendment by the AIPA (pre-AIPA 35 U.S.C. 102(e)).

3. Claims 11-59 are rejected under 35 U.S.C. 102(e) as being anticipated by Muller et al. (US Pat. No. 6,483,804).

4. As to claims 11-12, Muller shows in Fig 1A, a method of examining packets through a connection point (the point connects the network to the NIC of the circuit 100). Muller discloses the following steps:

* receiving a packet from a packet acquisition device (NIC), see col 6, lines 26-29, lines 54-60, col 8, lines 33-35;

* performing one or more parsing/extraction operations to create a record comprising a function of selected portions of the packet, see col 7, lines 31-44, col 8, lines 50-67, col 9, lines 1-5;

* looking up a flow-entry database 110 to determine if the packet is of an existing flow, see col 9, lines 18-24, col 11, lines 32-45 ;

* if the packet is of an existing flow, classifying the packet as belonging to the found existing flow, see col 11, lines 46-52; and

* if the packet is of a new flow, storing a new flow-entry in the flow-entry database 110, see col 11, lines 46-52.

5. As to claims 13-15, Muller teaches updating the flow-entry of the existing flow including measures selected from the set consisting of the total packet count, see col 7, lines 36-45, col 8, lines 50-54, lines 64-66.

6. As to claim 16, Muller shows that the function of the selected portions of the packet forms a signature (flow key), see col 8, lines 64-67, col 9, lines 1-5, col 11, lines 35-37.

7. As to claims 17-20, Muller shows at least one of the protocols uses source and destination addresses, see col 7, lines 31-40.

8. As to claim 21, Muller shows the looking up of the flow-entry database 110 uses a hash of the selected packet portions, see col 9, lines 18-22.

9. As to claim 22, Muller shows determining a set of one or more protocol from data in the packet, see col 10, lines 63-67, col 11, lines 27-30.

10. As to claim 23, Muller shows obtaining the last encountered state of the existing flow and performing any state operations required for a new flow, see col 9, lines 15-28.

11. As to claim 24, Muller shows identifying of the application program of the flow, see col 8, lines 60-61, col 12, lines 45-47.

12. As to claim 25, Muller shows storing identifying information for future packets, see col 9, lines 26-28.

13. As to claim 26, Muller shows identifying the application program of the flow, see col 8, lines 60-61, col 12, lines 45-47.

14. As to claim 27, Muller shows searching the parser record for the existence of one or more reference strings, see col 9, lines 32-36.

15. As to claim 28, Muller shows the state operations are carried by state processor , see col 9, lines 42-47, col 10, lines 61-63

16. As to claim 29-59, the claims are similar in scope to claims 11-28, and they are rejected under the same rationale.

Therefore, it can be seen from paragraphs 4-16 that Muller anticipates claims 11-59.

17. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

18. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Moustafa M. Meky whose telephone number is (703) 305-9697. The examiner can normally be reached on week days from 8:30 am to 4:30 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Ario Etienne, can be reached on (703) 308-7562. The fax phone number for this Group is (703) 308-9052.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the Group receptionist whose telephone number is (703) 305-9600. The fax number for the After-Final correspondence/amendment is (703) 746-7238. The fax number for official correspondence/amendment is (703) 746-7239. The fax number for Non-official draft correspondence/amendment is (703) 746-7240.

M.M.M

October 01, 2004

Moustafa M. Meky

MOUSTAFA M. MEKY
PRIMARY EXAMINER

Et al.FORM - 1449 SHEET 1 OF 3.

INFORMATION DISCLOSURE STATEMENT (Use several sheets if necessary)	ATTY. DOCKET NO. APPT-001-1-1	SERIAL NO. 10/684,776
	APPLICANT	
	FILING DATE 14 Oct 2003	GROUP ~~2155~~ 2157

DEC 17 2003 JC109 U.S. PATENT & TRADEMARK OFFICE

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
MMM	AA	5,680,585	10-1997	Bruell	703	26	
MMM	AB	5,721,827	02-1998	Logan et al.	709	217	
MMM	AC	6,272,151	08-2001	Gupta et al.	370	489	
MMM	AD	6,430,409	08-2002	Rossmann	455	422.1	
MMM	AE	6,516,337	02-2003	Tripp et al.	709	202	
MMM	AF	6,519,568	02-2003	Harvey et al.	705	1	
	AG						
	AH						
	AI						
	AJ						
	AK						

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLI-CATION DATE	COUNTRY	CLASS	SUB-CLASS	TRANS-LATION YES	NO
	AM							
	AN							

OTHER DISCLOSURES (Including Author, Title, Date, Pertinent Pages, Place of Publication, Etc.)

	AR	
	AS	

EXAMINER M. Meky	DATE CONSIDERED 9/30/2004

*EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to Applicant.

Best Available Copy

INFORMATION DISCLOSURE STATEMENT (Use several sheets if necessary)	ATTY. DOCKET NO. APPT-001-1-1	SERIAL NO. 10/684,776
	APPLICANT	
	FILING DATE 14 Oct 2003	GROUP ~~2155~~ 2157

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
MMM	BA	5,850,388	12-1998	Anderson et al.	370	252	
MMM	BB	6,097,699	08-2000	Chen et al	370	231	
MMM	BC	6,269,330	07-2001	Cidon et al.	704	43	
MMM	BD	6,453,345	09-2002	Trcka et al.	709	224	
MMM	BE	6,381,306	04-2002	Lawson et al.	379	32	
MMM	BF	6,282,570	08-2001	Leung et al.	709	224	
MMM	BG	5,761,429	06-1998	Thompson	709	224	
MMM	BH	5,799,154	08-1998	Kuriyan	709	223	
	BI						
	BJ						
	BK						

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLI-CATION DATE	COUNTRY	CLASS	SUB-CLASS	TRANS-LATION YES	NO
	BM							
	BN							

OTHER DISCLOSURES (Including Author, Title, Date, Pertinent Pages, Place of Publication, Etc.)

MMM	BR	Advanced Methods for Storage and Retrieval in Image; http://www.cs.tulane.edu/ww/Prototype/proposal.html; 1998
MMM	BS	Measurement and Analysis of the Digital DECT Propagation Channel; IEEE; 1998

EXAMINER	DATE CONSIDERED
M. Nehy	9-30-2004

*EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to Applicant.

al.FORM - 1449 SHEET 3 OF 3.

DEC 17 2003

INFORMATION DISCLOSURE STATEMENT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.
APPT-001-1-1	10/684,776
APPLICANT	
FILING DATE	GROUP
14 Oct 2003	~~2155~~ 2157

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
MMM	CA	5,530,958	06-1996	Agarwal et al.	711	3	
MMM	CB	4,458,310	07-1984	Chang, Shih-Jeh	711	119	
MMM	CC	6,003,123	12-1999	Carter et al.	711	207	
MMM	CD	5,530,834	06-1996	Colloff et al.	711	136	
MMM	CE	5,749,087	05-1998	Hoover et al.	711	108	
MMM	CF	3,949,369	04-1976	Churchill, Jr.	711	128	
MMM	CG	4,559,618	12-1985	Houseman et al.	365	49	
MMM	CH	4,910,668	03-1990	Okamoto et al.	711	207	
MMM	CI	5,917,821	06-1999	Gobuyan et al.	370	392	
	CJ						
	CK						

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLI-CATION DATE	COUNTRY	CLASSIFICATION	TRANS-LATION YES	NO
MMM	CM	JP 2003-44510A	02-2003	Japan	~~G06F017/30~~		
	CN						

OTHER DISCLOSURES (Including Author, Title, Date, Pertinent Pages, Place of Publication, Etc.)

	CR	
	CS	

EXAMINER	DATE CONSIDERED
M. Melz	9-30-2004

*EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to Applicant.

Best Available Copy

INFORMATION DISCLOSURE STATEMENT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.
APPT-001-1-1	10/684,776
APPLICANT: Dietz et al.	
FILING DATE: 14 Oct 2003	GROUP: ~~2141~~ 2157

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
MMM	AA	6,625,657 B1	Sep. 23, 2003	Bullard	709	237	Mar. 25, 1999
MMM	AB	6,330,226 B1	Dec. 11, 2001	Chapman et al.	370	232	Jan. 27, 1998
MMM	AC	6,651,099 B1	Nov. 18, 2003	Dietz et al.	709	224	Jun. 30, 2000
MMM	AD	6,424,624 B1	Jul. 23, 2002	Galand et al.	370	231	Oct. 7, 1998
MMM	AE	6,279,113 B1	Aug. 21, 2001	Vaidya	713	201	Jun. 4, 1998
MMM	AF	6,363,056 B1	Mar. 26, 2002	Beigi et al.	370	252	Jul. 15, 1998
MMM	AG	6,115,393	Sep. 5, 2000	Engel et al.	370	469	Jul. 21, 1995
MMM	AH	4,972,453	Nov. 20, 1990	Daniel, III et al.	379	10	Feb. 28, 1989
MMM	AI	5,535,338	Jul. 9, 1996	Krause et al.	395	200.20	May 30, 1995
MMM	AJ	5,802,054	Sep. 1, 1998	Bellenger	370	401	Aug. 16, 1996
MMM	AK	5,720,032	Feb. 17, 1998	Picazo, Jr. et al.	395	200.2	Jan. 28, 1997

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLI-CATION DATE	COUNTRY	CLASS	SUB-CLASS	TRANS-LATION YES	NO
	AL							
	AM							

OTHER DISCLOSURES (Including Author, Title, Date, Pertinent Pages, Place of Publication, Etc.)

MMM	AN	R. Periakaruppam and E. Nemeth. "GTrace-A Graphical Traceroute Tool." 1999 Usenix LISA. Available on www.caida.org, URL: http://www.caida.org/outreach/papers/1999/GTrace/GTrace.pdf
MMM	AO	W. Stallings. "Packet Filtering in the SNMP Remote Monitor." November 1994. Available on www.ddj.com, URL: http://www.ddj.com/documents/s=1013/ddj9411h/9411h.htm

EXAMINER	DATE CONSIDERED
M. Meky	9-30-2004

*EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to Applicant.

Notice of References Cited	Application/Control No. 10/684,776	Applicant(s)/Patent Under Reexamination DIETZ ET AL.	
	Examiner Moustafa M Meky	Art Unit 2157	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-6,483,804	11-2002	Muller et al.	370/230
	B	US-6,570,875	05-2003	Hegde	370/389
	C	US-6,452,915	09-2002	Jorgensen	370/238
	D	US-6,466,985	10-2002	Goyal et al.	709/238
	E	US-6,453,360	09-2002	Muller et al.	709/250
	F	US-6,243,667	06-2001	Kerr et al.	703/27
	G	US-6,118,760	09-2000	Zaumen et al.	370/229
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

BIBDATASHEET

Bib Data Sheet

CONFIRMATION NO. 3352

SERIAL NUMBER	FILING DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
10/684,776	10/14/2003 RULE	709	2157	APPT-001-1-1

APPLICANTS

Russell S. Dietz, San Jose, CA;

Joseph R. Maixner, Aptos, CA;
Andrew A. Koppenhaver, Littleton, CO;William H. Bares, Germantown, TN;
Haig A. Sarkissian, San Antonio, TX;
James F. Torgerson, Andover, MN;

** CONTINUING DATA *************************
This application is a CON of 09/608,237 06/30/2000 PAT 6,651,099
which claims benefit of 60/141,903 06/30/1999
Yes, MMM

** FOREIGN APPLICATIONS ********************
None MMM

IF REQUIRED, FOREIGN FILING LICENSE GRANTED
** 01/16/2004

	STATE OR COUNTRY	SHEETS DRAWING	TOTAL CLAIMS	INDEPENDENT CLAIMS
Foreign Priority claimed ☐ yes ☒ no 35 USC 119 (a-d) conditions met ☐ yes ☒ no ☐ Met after Allowance Verified and Acknowledged MMM Examiner's Signature Initials	CA	18	49	3

ADDRESS
21921
DOV ROSENFELD
5507 COLLEGE AVE
SUITE 2
OAKLAND , CA
94618

TITLE
Method and apparatus for monitoring traffic in a network

FILING FEE RECEIVED	FEES: Authority has been given in Paper No. _________ to charge/credit DEPOSIT ACCOUNT No. _________ for following:	☐ All Fees ☐ 1.16 Fees (Filing) ☐ 1.17 Fees (Processing Ext. of time) ☐ 1.18 Fees (Issue)

Index of Claims

Application No.	Applicant(s)
10/684,776	DIETZ ET AL.
Examiner	**Art Unit**
Moustafa M Meky	2157

√	Rejected	–	(Through numeral) Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

Claim Final	Claim Original	Date 10/1/2003	Claim Final	Claim Original	Date	Claim Final	Claim Original	Date
	1			51	√		101	
	2			52	√		102	
	3			53	√		103	
	4			54	√		104	
	5			55	√		105	
	6			56	√		106	
	7			57	√		107	
	8			58	√		108	
	9			59	√		109	
	10			60			110	
	11	√		61			111	
	12	√		62			112	
	13	√		63			113	
	14	√		64			114	
	15	√		65			115	
	16	√		66			116	
	17	√		67			117	
	18	√		68			118	
	19	√		69			119	
	20	√		70			120	
	21	√		71			121	
	22	√		72			122	
	23	√		73			123	
	24	√		74			124	
	25	√		75			125	
	26	√		76			126	
	27	√		77			127	
	28	√		78			128	
	29	√		79			129	
	30	√		80			130	
	31	√		81			131	
	32	√		82			132	
	33	√		83			133	
	34	√		84			134	
	35	√		85			135	
	36	√		86			136	
	37	√		87			137	
	38	√		88			138	
	39	√		89			139	
	40	√		90			140	
	41	√		91			141	
	42	√		92			142	
	43	√		93			143	
	44	√		94			144	
	45	√		95			145	
	46	√		96			146	
	47	√		97			147	
	48	√		98			148	
	49	√		99			149	
	50	√		100			150	

Search Notes	Application No. 10/684,776	Applicant(s) DIETZ ET AL.
	Examiner Moustafa M Meky	**Art Unit** 2157

SEARCHED			
Class	Subclass	Date	Examiner
709	200, 201, 220, 223, 224,231	10/1/2004	MMM
709	232, 236	10/1/2004	MMM
709	238-240	10/1/2004	MMM
709	246	10/1/2004	MMM
370	389,392	10/1/2004	MMM

INTERFERENCE SEARCHED			
Class	Subclass	Date	Examiner

SEARCH NOTES (INCLUDING SEARCH STRATEGY)		
	DATE	EXMR
WEST SEARC	10/1/2004	MMM

U.S. Patent and Trademark Office

Part of Paper No. 2

OIPE JC168 MAR 0 4 2005 PATENT & TRADEMARK OFFICE

IFW $ CE

Our Ref./Docket No: APPT-001-1-1 Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): Dietz, *et al.*	Group Art Unit: 2157
Application No.: 10/684,776	Examiner: Moustafa M. Meky
Filed: October 14, 2003	
Title: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK	

RESPONSE TO OFFICE ACTION UNDER 37 CFR 1.111

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Commissioner:

This is a response to the Office Action of October 5, 2004.

Any ***amendments to the specification*** begin on a new page immediately after these introductory remarks. Any ***amendments to the claims*** begin on a new page immediately after such ***amendments to the specification***, if any. Any ***amendments to the drawings*** begin on a new page immediately after such ***amendments to the claims***, if any.

The ***Remarks/arguments*** begin on a new page immediately after such ***amendments to the drawings***, if any.

A Declaration by inventor Russell S. Dietz, and a set of Exhibits are attached following the ***Remarks/arguments***.

Certificate of Mailing under 37 CFR 1.8	
I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on.	
Date: Mar. 2, 2005	Signed: [signature] Name: Amy Drury

REMARKS

Claims 11–59 are the claims of record of the application. Claims 11–59 have been rejected.

In paragraph 3 of the Office Action, claims 11–59 have been rejected under 35 USC 102(e) as anticipated by Muller et al. (U.S. Patent 6,483,804).

The reference date for U.S. Patent 6,483,804 is 1 March 1999. The independent claims of the present invention were reduced to practice prior to this reference date. A declaration by the first inventor Russell S. Dietz under 37 CFR 1.131 swearing behind U.S. Patent 6,483,804 is attached, together with several Exhibits to such declaration. The declaration and exhibit shows that prior to the reference date of March 1, 1999, the inventor conceived of the invention of independent claims 11, 29, and 54 of the present invention. Furthermore, the declaration and exhibit shows that prior to the reference date of March 1, 1999, the inventor reduced to practice the invention of independent claims 11, 29, and 54 of the present invention. The invention of these claims functioned for its intended purpose by running the apparatus on a computer, and a program implementing the method on test data that was part of a node of a network.

Thus, the rejection of independent claims 11, 29, and 54 under 35 USC 102(e) is overcome. Withdrawal of the rejection and allowance of independent claims 11, 29, and 54 are respectfully requested.

Furthermore, the remaining claims 12–28, 30–53, and 55–59 are all dependent on these independent claims 11, 29, and 54. Thus, these claims are also allowable. Withdrawal of the rejection and allowance of claims 12–28, 30–53, and 55–59 are respectfully requested.

For these reasons, and in view of the above amendment, this application is now considered to be in condition for allowance and such action is earnestly solicited.

The Applicants believe all of Examiner's rejections have been overcome with respect to all remaining claims (as amended), and that the remaining claims are allowable. Action to that end is respectfully requested. If the Examiner has any questions or comments that would advance the prosecution and allowance of this application, an email message to the undersigned at dov@inventek.com, or a telephone call to the undersigned at +1-510-547-3378 is requested.

Respectfully Submitted,

Mar. 2, 2005
Date

Dov Rosenfeld, Reg. No. 38687

Address for correspondence:
Dov Rosenfeld
5507 College Avenue, Suite 2,
Oakland, CA 94618
Tel. 510-547-3378; Fax: +1-510-291-2985; Email:dov@inventek.com

Our Ref./Docket No: APPT-001-1-1 Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

OIPE JC18 MAR 04 2005 PATENT & TRADEMARK OFFICE

Applicant(s): Dietz, *et al.* Application No.: 10/684,776 Filed: October 14, 2003 Title: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK	Group Art Unit: 2157 Examiner: Moustafa M. Meky

TRANSMITTAL: RESPONSE TO OFFICE ACTION

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Commissioner:

Transmitted herewith is a response to an office action for the above referenced application. Included with the response are:

X A Declaration under 37 CFR 1.131 with Exhibits;

This application has:

____ a small entity status. If a claim for such status has not earlier been made, consider this as a claim for small entity status.

____ No additional fee is required.

03/10/2005 AWONDAF1 00000092 10684776
01 FC:1252 450.00 OP

Certificate of Mailing under 37 CFR 1.8

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on.

Date: Mar. 2, 2005 Signed: [signature]
Name: Amy Drury

_____ Applicant(s) believe(s) that no Extension of Time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition for an extension of time.

_X_ Applicant(s) hereby petition(s) for an Extension of Time under 37 CFR 1.136(a) of:

_____	one months ($120)	_X_	two months ($450)
_____	three months ($1020)	_____	four months ($1590)

If an additional extension of time is required, please consider this as a petition therefor.

_X_ A credit card payment form for the required fee(s) is attached.

_X_ The Commissioner is hereby authorized to charge payment of the following fees associated with this communication or credit any overpayment to Deposit Account No. 50-0292 (A DUPLICATE OF THIS TRANSMITTAL IS ATTACHED):

_X_ Any missing filing fees required under 37 CFR 1.16 for presentation of additional claims.

_X_ Any missing extension or petition fees required under 37 CFR 1.17.

Respectfully Submitted,

Mar. 2, 2005
Date

Dov Rosenfeld, Reg. No. 38687

Address for correspondence:
Dov Rosenfeld
5507 College Avenue, Suite 2,
Oakland, CA 94618
Tel. 510-547-3378; Fax: +1-510-291-2985

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

OIPE JC183 MAR 04 2005 PATENT & TRADEMARK OFFICE

Application Number	10/684,776
Filing Date	14 Oct 2003
First Named Inventor	Dietz, Russell S.
Group Art Unit	2157
Examiner Name	Moustafa M. Meky
Attorney Docket Number	APPT-001-1-1

ENCLOSURES ***(check all that apply)***

- [ ] Fee Transmittal Form
- [x] Fee Attached
- [x] Amendment / Response
- [ ] [ ] After Final
- [x] [x] Affidavits/declaration(s) under 1,131 with Exhibits
- [ ] Extension of Time Request
- [ ] Express Abandonment Request
- [ ] Information Disclosure Statement
- [ ] Certified Copy of Priority Document(s)
- [ ] Response to Missing Parts/ Incomplete Application
- [ ]
- [ ] [ ] Response to Missing Parts under 37 CFR 1.52 or 1.53
- [ ] Assignment Papers *(for an Application)*
- [ ] Drawing(s)
- [ ] Licensing-related Papers
- [ ] Petition Routing Slip (PTO/SB/69) and Accompanying Petition
- [ ] To Convert a Provisional Application
- [ ] Power of Attorney, Revocation Change of Correspondence Address
- [ ] Terminal Disclaimer
- [ ] Small Entity Statement
- [ ] Request of Refund
- [ ] After Allowance Communication to Group
- [ ] Appeal Communication to Board of Appeals and Interferences
- [ ] Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*
- [ ] Proprietary Information
- [ ] Status Letter
- [x] Additional Enclosure(s) *(please identify below):*
- [x] Return Postcard
- [x] Exhibits to Declaration under 1.131
- [ ]

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT/ CORRESPONDENCE ADDRESS

Firm *or* Individual name	Dov Rosenfeld, Reg. No. 38687
Signature	[signature]
Date	March 2, 2005

ADDRESS FOR CORRESPONDENCE

Firm *or* Individual name	Dov Rosenfeld 5507 College Avenue, Suite 2, Oakland, CA 94618, Tel: 510-547-3378

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on this date: March 2, 2005

Type or printed name	Amy Drury		
Signature	[signature]	Date	March 2, 2005

Applicant(s): Dietz, *et al.* Application No.: 10/684,776 Filed: October 14, 2003 Title: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK	Group Art Unit: 2157 Examiner: Moustafa M. Meky

DECLARATION UNDER 37 CFR 1.131

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Commissioner:

1. I am an inventor of claims 11–59 of the above referenced patent application.

2. Claims 11–59 have been rejected under 35 USC 102(e) as anticipated by Muller et al. (U.S. Patent 6,483,804) that has a reference date of March 1, 1999.

3. Prior to the reference date of March 1, 1999, I conceived of the invention of claims 11, 29, and 54 shown in the following:

- **Exhibit A0:**Directory of documents
- **Exhibit A1:** Technically Elite MeterFlow Accelerator Modules System Specification (Document MFASystem.pdf)
- **Exhibit A2:** Technically Elite MeterFlow Accelerator Parser Module Specification (Document MFAParser.pdf)
- **Exhibit A3:** Technically Elite MeterFlow Accelerator Analyzer Module Specification (Document MFAAnalyze.pdf)
- **Exhibit A4:** Protocol Tracking Summary (Document MFAProtocolLayout.pdf)

A copy of each of Exhibits A0 to A4 is attached. The dates on each such copy has been deleted. **I confirm that the dates are all prior to March 1, 1999**. These exhibits are as follows.

Exhibit A0 is a dated computer directory of documents that describe the design and tests to run the design on real data.

Exhibit A1 is the overall design of the system that implements the method claims 11 and 54, and includes the elements of claim 29.

Exhibit A2 is a detailed design of the parsing/extraction unit that carries out step (b) of claim 11, that corresponds to element (c): the parser subsystem of claim 29, and that carries out the parsing/extraction operations of element (b) of claim 54.

Exhibit A3 is a detailed design of the analyzer that carries out the operations of elements (c), (d) ,and (e) of method claim 29, that corresponds to elements (d), (e) and (f) unit parsing/extraction unit that carries out carries out the operations of elements (c), (d) ,and (e) of method claim 54, that corresponds to element (c): the parser subsystem of claim 29, and that carries out the parsing/extraction operations of element (b) of claim 54.

Exhibit A4 is a summary of the protocols that the system can analyze.

Note that Technically Elite was the name of the predecessor of the assignee of the present invention at the time.

3. Prior to the reference date of March 1, 1999, I reduced to practice the invention of claims 11, 29, and 54 of the above referenced patent application as shown in the following documents:

- **Exhibit B0** is a dated computer directory of test data and documents used therefore.
- **Exhibit B1:** Technically Elite MeterFlow Accelerator Modules Testbench Specification (Document MFATest.pdf in directory of Exhibit A0)
- **Exhibit B2:** The first page of file big.cpl.

  The cpl files (big.cpl, bigfgc3.cpl, bigfgpc.cpl, bigfpayl.cpl, bigfpayl2.cpl, bigfpgrp.cpl, bigfpgrp2.cpl, bigfrag.cpl, bigfrag2.cpl, output.cpl, Protocols.cpl, short.cpl, shrtfpg2.cpl, shrtfps3.cpl, shrtfps4.cpl, shrtfps5.cpl, shrttunl.cpl) are files for the protocol compiler of all the actual protocols recognized by the system. These files include a description of the parser information for the parser to perform the parsing/extracting operation according to the protocol. They also contain the state processing states for the state operations of elements (d) and (e) of claim 54. The first page of one file is provided.
- **Exhibit B3:** The first four pages of a printout of file MFATEST.HEX that contains the actual packets captured by the packet acquisition device described in element (a) of claims 11 and 54, and corresponding to the contents of element (b), the input buffer memory of claim 29. The packet acquisition device for the experiment was a SUN workstation connected to a connection point of a network.
- **Exhibit B4**: The file packets.txt that describes the nature of the packets in MFATEST.HEX.
- **Exhibit B5:** The contents of files mfaptpkt.txt and mfaptpkt2.txt that are files that contain the elements that were extracted by the parsing/extracting of
- **Exhibit B6:** The contents of files mfaptkey.txt and mfaptkey2.txt that are files that contain the keys that were generated from the extracted data (Exhibit B4) and used for looking up the flow-entry database per element (c) of method claims 11 and 54, which are operations carried out by the lookup engine of element (e) of claim 29.

- **Exhibit B7:** The first four pages of a printout of file MFATEST.TXT that includes the decoded packets that were generated by operation of the method that includes the elements of each of method claims 11 and 54, by an apparatus that includes the elements of claim 29.
- **Exhibit B8:** Protocol Definition Language (PDL) Reference Guide (the document MFS-PDL-Reference.pdf) that provides a reference to the protocol definition language used in cpl files.
- **Exhibit B9:** State-based Sub-Classification Overview (document MFS-State-Classification.pdf) that describes the states of some of the protocols that are supported.

The invention functioned for its intended purpose by running the apparatus on a computer, and a program implementing the method on test data that was part of a node of a network.

The above exhibits are each a copy. The date on each copy has been deleted. I confirm that the deleted dates are each prior to March 1, 1999.

Therefore, and in summary, I declare that the inventions of claims 11, 29, and 54 were reduced to practice prior to the reference data of March 1, 1999.

I hereby declare under penalty of perjury under the laws of the United States of America 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 Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Signed,

March 1, 2005
Date

Russell S. Dietz

Address for correspondence:
Dov Rosenfeld
5507 College Avenue, Suite 2,
Oakland, CA 94618
Tel. 510-547-3378
Fax: +1-510-291-2985; Email:dov@inventek.com

- **Exhibit A0:**Directory of documents

[redacted]Hardware Specification-dir.txt
Directory of M:\aaa-----INVENTEK CLIENTS\Hifn\Patents\APPT-001-1-1 filed [redacted]\Proof of Reductn to Practice\[redacted]-Hardware Specification\

M:\aaa-----INVENTEK CLIENTS\Hifn\Patents\APPT-001-1-1 filed [redacted]\Proof of Reductn to Practice\[redacted]-Hardware Specification\

==

MFAAnalyze.pdf	317 KB	[redacted]	04:47:46 AM	a
MFAAppl.pdf	12 KB	[redacted]	05:50:46 AM	a
MFAParser.pdf	124 KB	[redacted]	03:56:18 AM	a
MFAProtocolLayout.pdf	13 KB	[redacted]	09:24:38 AM	a
MFASystem.pdf	93 KB	[redacted]	03:56:50 AM	a
MFATest.pdf	18 KB	[redacted]	03:56:26 AM	a

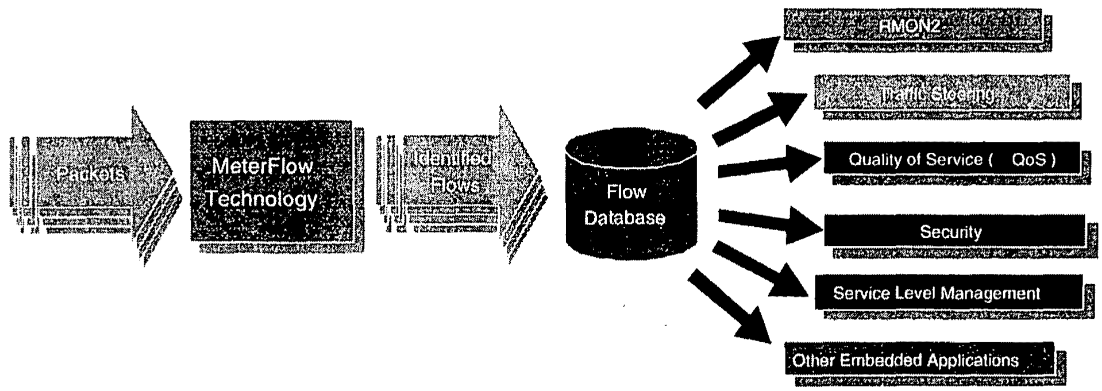
^^
Total 0 folder(s); 6 file(s)

Total files size: 1 MB; 580 KB; 593910 Bytes

^^

- **Exhibit A1:** Technically Elite MeterFlow Accelerator Modules System Specification (Document MFASystem.pdf)

Technically Elite MeterFlow Accelerator Modules System Specification



1 Introduction

The Technically Elite MeterFlow Accelerator is a set of synthesizable modules designed to do wire speed hardware based application traffic recognition for Fast Ethernet and Gigabit Ethernet. Originally designed for RMON2 network management the MeterFlow Accelerator also allows Layer 3 (Network) through Layer 7 (Application) visibility for switches and routers .The MeterFlow Accelerator poaches the network traffic and builds a "flow" database that is then extracted for further processing. Each flow consists of the information necessary to track the conversation between the two end points of the traffic. This conversation is also characterized and vital statistics counted. The resulting flow database is useful for many applications. Some of these include RMON2 network management, traffic steering, quality of service, security, and service level management.

1.1 Technically Elite MeterFlow Accelerator Highlights

- Synthesizable modules written in both the Verilog and VHDL
- Processes up to Gigabit speeds
- Complete traffic data
- State based parallel processing architecture
- Distributes work to eliminate bottlenecks
- Layer 3 network protocols to dynamic transaction oriented applications at Layer 7
- Scalable architecture for any size switch or probe
- Can recognize over 2000 different protocols
- Extensible to new protocols
- Recognizes encapsulations
- Open interface
- Easy to use software tools including protocol compiler and C model

2 Overview

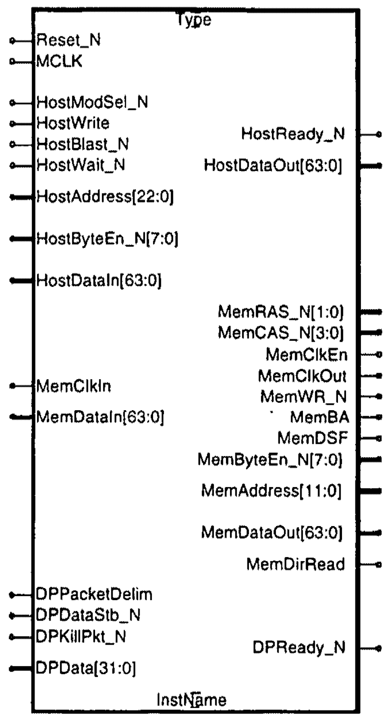
The Technically Elite MeterFlow Accelerator Modules System Specification outlines the general system requirements. It provides an overview of how the modules interact with each other and external devices. It also provides guidance for the testing methodology to be used in the verification of the cores.

The Technically Elite network analysis suite consists of three main components. These are the parser, the analyzer and software. The parser works on the information contained in a single packet. The analyzer builds flow information across multiple packets. The software consists of a compiler, a C model and a database of protocol information. The database delineates all the information needed by the parser to recognize the protocols and build the flow key. The database also delineates how each protocol's flow entry should be updated as well as the procedure to recognize multi-packet protocols (state processing). Also included in the module set is a host interface module. This module defines a burst oriented bus interface compatible with the Intel i960. This module can be easily modified to interface to other bus types.

After initialization the network data first goes to the parser. The parser attempts to recognize the various possible protocols in a particular packet. It then builds a flow key data structure that is passed to the analyzer. The analyzer first attempts to find a particular packets related flow in its' database. Then using the information it gathered from previous packets in this flow and the current packets' data it updates the flows' data base entry. Once a flow has been completely recognized, updates consist of gathering statistics. On a regular basis the external system reads the flow data base for further processing.

The parser and analyzer modules are RTL synthesizable modules written in both the Verilog and VHDL hardware description languages. Each major component of the cores has a matching testbench. The testbenches fully exercise the unit under test and provide an automated verification environment. Input stimulus files are automatically generated by the compiler and expected data files are automatically generated by the C model.

3 Top Level MeterFlow Accelerator Module Symbol



4 Top Level MeterFlow Accelerator Module Pin Descriptions

4.1.1.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low. When this signal is active the module sets it's registers to their default condition and suspends operation. It will only respond to host access cycles. The DataPort interface will keep **DPReady_N** active to avoid problems for the external circuitry.
MCLK	IN	1	Module Clock. All internal and external transfers except for memory transfers are synchronized by this signal.

4.1.1.2 Memory Interface

Signal	Dir	Width	Description
MemClkIn	IN	1	Memory clock in. This signal is used to generate the memory interface timing.
MemRAS_N	OUT	2	Memory Row Address Strobe bus – active low.
MemCAS_N	OUT	4	Memory Column Address Strobe bus– active low.
MemClkEn	OUT	1	Memory Clock Enable. Some memories require this signal to be disabled for a certain amount of time after reset.
MemClkOut	OUT	1	Memory Clock Out. This signal is used by synchronous memory for all operations. **MemClkIn** is buffered and sent out on this pin. This helps reduce skew between this clock and the other signals.
MemWR_N	OUT	1	Memory Write – active low.
MemBA	OUT	1	Memory Bank Address. Used by multi-bank memory to select the bank the current operation is to operate on.
MemDSF	OUT	1	Memory Special Function select.
MemByteEn_N	OUT	8	Memory Byte Enable bus– active low.
MemAddress	OUT	12	Memory Address bus.
MemDataIn	IN	64	Memory Data Input bus.
MemDataOut	OUT	64	Memory Data Output bus.
MemDirRead	OUT	1	Memory Data bus Direction is Read. This signal is used to control the tri-state enable on the bidirectional memory data bus. If **MemDirRead** is active data is coming into the module from the memory. If it is inactive the module is driving data out to the memory.

4.1.1.3 Host Interface Signals

Signal	Dir	Width	Description
HostModSel_N	IN	1	Host interface Module Select - active low. **HostModSel_N** is sampled on the rising edge of **MCLK**. If it is active, it signifies that the external host is attempting to access the module.
HostWrite	IN	1	Write. Write is sampled on the rising edge of **MCLK**. This signal is only valid when **HostModSel_N** is active. If this signal is active, the host is attempting to write to the module. Inactive this signal sign signifies a read from the module. It should also be used to control the direction of the host data bus if it is bidirectional.
HostBlast_N	IN	1	Burst Last – active low. **HostBlast_N** is sampled on the rising edge of **MCLK**. **HostBlast_N** tells the module that the current transfer is the last transfer in this burst.
HostWait_N	IN	1	Wait – active low. **HostWait_N** is sampled on the rising edge of **MCLK**. The host asserts **HostWait_N** when it wishes to slow transfers between itself and the module. This could also be used by additional interface logic to slow transfers so it can multiplex the bus down to a smaller size without additional FIFOs. If wait is active, **HostReady_N** is blocked.
HostReady_N	OUT	1	Ready – active low. **HostReady_N** should be sampled on the rising edge of **MCLK**. The module returns **HostReady_N** when the current cycle is completed. For a write operation, **HostReady_N** means that the **HostDataIn** bus has been latched. For a read operation **HostReady_N** means that the requested data is on the **HostDataOut** bus and is valid. **HostReady_N** is blocked by **HostWait_N**.
HostAddress	IN	23	Host Address bus. HostAddress is sampled on the rising edge of **MCLK** if **HostModSel_N** is active. This bus defines the first address in this burst to access in the 64 Megabyte address space of the module. See Section x.x.x for the Address Utilization Map.
HostByteEn_N	IN	8	Host Byte Enable bus – Active low. **HostWait_N** is sampled on the rising edge of **MCLK**.
HostDataIn	IN	64	Host Data Input bus. **HostDataIn** is sampled on the rising edge of **MCLK** if **HostWrite** is active and **HostWait_N** is inactive.
HostDataOut	OUT	64	Host Data Output bus. **HostDataOut** should be sampled on the rising edge of **MCLK**. Data on this bus is valid during a read cycle when **HostReady_N** is active.

4.1.1.4 Data Port Interface

Signal	Dir	Width	Description
DPPacketDelim	IN	1	Data Port Packet Delimiter. This signal should be driven active when the external logic wants to send a packet to the module. **DPPacketDelim** should remain active during the entire packet transfer. **DPPacketDelim** must go inactive for one clock between packets.
DPDataStb_N	IN	1	Data Port Data Strobe. When active, this signal tells the module that data on the **DPData** bus is valid. If **DPReady_N** was inactive at the end of the previous cycle, **DPDataStb_N** should not be driven active. If **DPReady_N** goes inactive in the same cycle as **DPDataStb_N**, then the module will latch the incoming data so that no data is lost.
DPKillPkt_N	IN	1	Data Port Kill Packet. If this signal becomes active while **DPPacketDelim** is active, the module will attempt to stop processing the current packet and flush it's input FIFO. If however, parsing of the packet is completed, the packet will not be able to be recalled. This should only be a problem in a 'cut through' implementation.
DPReady_N	OUT	1	Data Port Ready – active low. This signal when driven active means that the module can accept new data. If however the modules' input FIFO is filled, **DPReady_N** will be driven inactive. To prevent overruns, **DPReady_N** will go inactive when the module can actually accept one more data transfer.
DPData	IN	32	Data Port Data bus.

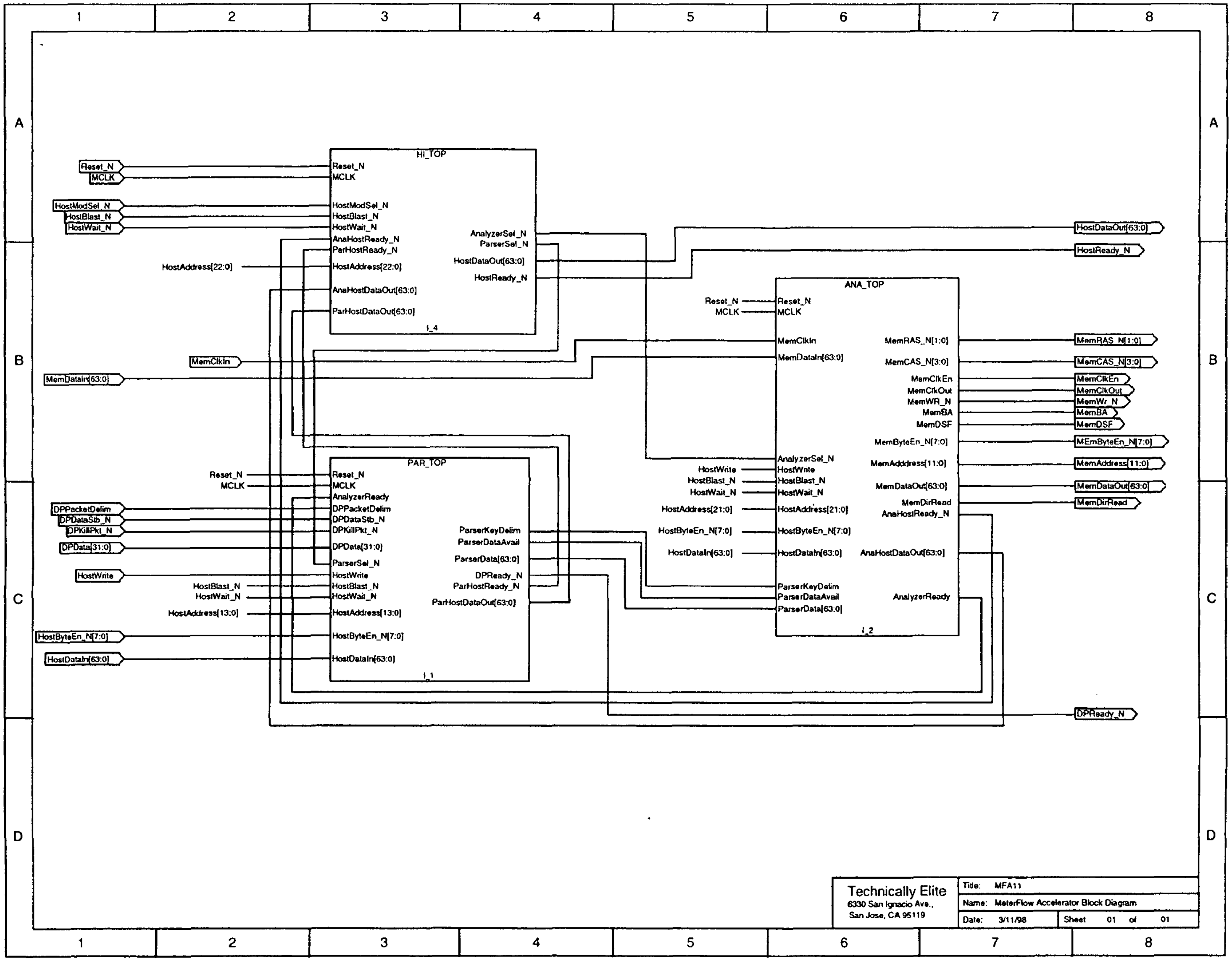
5 MeterFlow Accelerator Modules Block Diagrams

The following page is the top level block diagram for the MeterFlow Accelerator Module.

Confidential

MeterFlow Accelerator Modules System Specification 1g.doc
Page 7 of 22

[redacted]



Technically Elite

CONFIDENTIAL

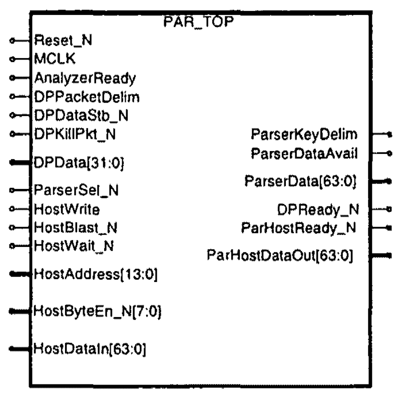
6 Description of Modules and Software

6.1 Parser Module

6.1.1 Parser Module Highlights

- Builds key and payload data structure for analyzer (flow key)
- Scaleable protocol pattern recognition engine
- Supports from 1 to 2048 simultaneous unique protocol patterns
- At 62.5 MegaHertz can process up to 1.5 MegaPackets per second
- Accepts protocol database output from MeterFlow compiler

6.1.2 Parser Module Symbol



6.1.3 Parser Module Description

The parser module consist of two main sub-modules. These are the pattern recognition engine and the slicer. The parser module pouches the network data through the DataPort interface. The data is first processed by the pattern recognition engine. This engine consists of a database and a comparison engine. The database can reside in ROM or RAM. If the database is in a RAM the parser can be programmed to recognize new protocols or a different set of protocols.

The set of specified protocols defines a tree of linked nodes. Each protocol is either a parent node or a terminal node. A protocol is a parent node if it links to other protocols that can be contained in it. For example IP is a parent to UDP. As each protocol is recognized, the pattern recognition engine emits a unique protocol identifier. It also emits a process code that the slicer uses to build the flow key.

The slicer extracts information from the packet to build the flow key. For example, it will extract the source and destination addresses from the packet and pack them into the flow key data structure. It may also process certain parts of the packet to speed up flow processing performed by the analyzer. It will build a hash value from certain parts of the packet to speed looking up the flow in the analyzers' database.

6.1.4 Parser Module Pin Descriptions

6.1.4.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low. When this signal is active the parser sets it's registers to their default condition and suspends operation. It will only respond to host access cycles. The DataPort interface will keep **DPReady_N** active to avoid problems for the external circuitry.
MCLK	IN	1	Module Clock. All internal and external transfers except for memory transfers are synchronized by this signal.

6.1.4.2 Analyzer Interface

Signal	Dir	Width	Description
AnalyzerReady	IN	1	Analyzer Ready. This signal tells the parser that the analyzer can accept data.
ParserKeyDelim	OUT	1	Parser Key Delimiter. The **ParserKeyDelim** signal becomes active when the first quadword of a new key is ready to transfer to the analyzer. It goes inactive when the last quadword of the key is transferred.
ParserDataAvail	OUT	1	Parser Data Available. If this signal is active the data on the **ParserData** bus is valid.
ParserData	OUT	64	Parser Data bus.

6.1.4.3 Data Port Interface

Signal	Dir	Width	Description
DPPacketDelim	IN	1	Data Port Packet Delimiter. This signal should be driven active when the external logic wants to send a packet to the parser. **DPPacketDelim** should remain active during the entire packet transfer. **DPPacketDelim** must go inactive for one clock between packets.
DPDataStb_N	IN	1	Data Port Data Strobe. When active, this signal tells the parser that data on the **DPData** bus is valid. If **DPReady_N** was inactive at the end of the previous cycle, **DPDataStb_N** should not be driven active. If **DPReady_N** goes inactive in the same cycle as **DPDataStb_N**, then the parser will latch the incoming data so that no data is lost.
DPKillPkt_N	IN	1	Data Port Kill Packet. If this signal becomes active while **DPPacketDelim** is active, the parser will attempt to stop processing the current packet and flush it's input FIFO. If however, parsing of the packet is completed, the packet will not be able to be recalled. This should only be a problem in a 'cut through' implementation.
DPReady_N	OUT	1	Data Port Ready – active low. This signal when driven active means that the parser can accept new data. If however the parser's input FIFO is filled, **DPReady_N** will be driven inactive. To prevent overruns, **DPReady_N** will go inactive when the parser can actually accept one more data transfer.
DPData	IN	32	Data Port Data bus.

6.1.4.4 Host Interface Signals

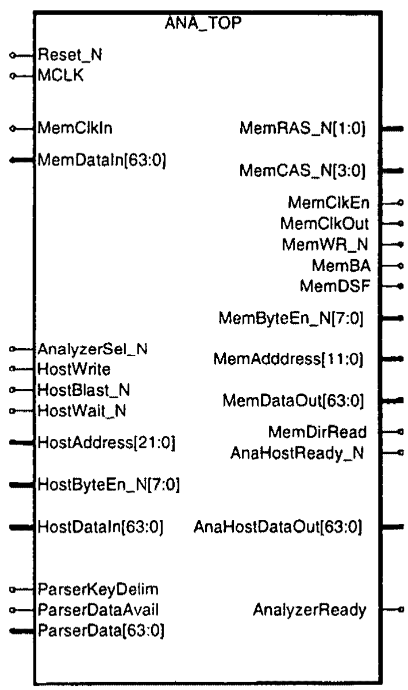
Signal	Dir	Width	Description
ParserSel_N	IN	1	Parser Select - active low. **ParserSel_N** is sampled on the rising edge of **MCLK**. If it is active, it signifies that the external host is attempting to access the parser.
HostWrite	IN	1	Write. Write is sampled on the rising edge of **MCLK**. This signal is only valid when **ParserSel _N** is active. If this signal is active, the host is attempting to write to the parser. Inactive this signal sign signifies a read from the parser.
HostBlast_N	IN	1	Burst Last – active low. **HostBlast_N** is sampled on the rising edge of **MCLK**. **HostBlast_N** tells the parser that the current transfer is the last transfer in this burst.
HostWait_N	IN	1	Wait – active low. **HostWait_N** is sampled on the rising edge of **MCLK**. The host asserts **HostWait_N** when it wishes to slow transfers between itself and the parser.
ParHostReady_N	OUT	1	Parser to Host Ready – active low. **ParHostReady_N** should be sampled on the rising edge of **MCLK**. The parser returns **ParHostReady_N** when the current cycle is completed. For a write operation, **ParHostReady_N** means that the **HostDataIn** bus has been latched. For a read operation **ParHostReady_N** means that the requested data is on the **ParHostDataOut** bus and is valid. **ParHostReady_N** is blocked by **HostWait_N**.
HostAddress	IN	13	Host Address bus. HostAddress is sampled on the rising edge of **MCLK** if **ParserSel_N** is active. This bus defines the first address in this burst to access in the 64 Kilobyte address space of the Parser. See Section x.x.x for the Address Utilization Map.
HostByteEn_N	IN	8	Host Byte Enable bus – Active low. **HostWait_N** is sampled on the rising edge of **MCLK**.
HostDataIn	IN	64	Host Data Input bus. **HostDataIn** is sampled on the rising edge of **MCLK** if **HostWrite** is active and **HostWait_N** is inactive.
ParHostDataOut	OUT	64	ParserHost Data Output bus. **ParHostDataOut** should be sampled on the rising edge of **MCLK**. Data on this bus is valid during a read cycle when **ParHostReady_N** is active.

6.2 Analyzer Module

6.2.1 Analyzer Module Highlights

- "Flexible" Rule-based Traffic Classification
- State-based Tracking of Traffic
- ***Multiple*** Packets for Layer Processing
- Internal Cache and Memory Controller (32 - 64KB)
- Direct High Bandwidth (64 bit) Memory Interface
- Up to 16MB of memory (75K Flows)
- SG/SDRAM Support
- Programmable Rules/State Engine
- Selectable Protocols in Flows
- Future Protocols Support
- Scalable System Design

6.2.2 Analyzer Module Symbol



6.2.3 Analyzer Module Description

The analyzer module consists of the flow lookup engine, the flow insertion/deletetion engine, the simple rules engine, the complex rules engine, the caching memory controller, the host update controller and the process synchronizer. Each of these sub-modules work in parallel to create and update flows.

As a flow key enters the analyzer, the lookup engine attempts to find it in the flow database. If the flow exists, the lookup engine retrieves the flow from the caching memory controller. It then makes a decision based on the state information included in the flow entry to either send it to the simple rules engine, the complex rules engine or to update the flow entry itself. This updating consists of adding values to counters in the flow database entry. If a flow does not exist, the flow key is sent to the flow insertion/deletetion engine which adds the flow to the database. Based on the flow key information the flow insertion/deletetion engine may be also send the new flow to one of the rules engines for processing.

The simple rules engine updates the flow based on the current state and the flow key information. The complex rules engine processes multi packet protocol recognition. It may have to search through a series of possible states to determine the flow's actual state. The result of the complex engine's processing is a consolidated flow entry. For example, a PointCast session will open multiple conversations that on a packet by packet basis look like separate flows. Since each conversation is merely a subflow under the PointCast master flow, a single flow that consolidates all of the information for the flow is desired.

The caching memory controller can be setup to work with various configurations of SDRAM or SGRAM. It uses it's cache to optimize memory bandwidth. On a typical network the packets will have a certain amount of congruity. This means that the cache can have a high hit rate.

6.2.4 Analyzer Module Pin Out

6.2.4.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low. When this signal is active the analyzer sets it's registers to their default condition and suspends operation. It will only respond to host access cycles.
MCLK	IN	1	Module Clock. All internal and external transfers except for memory transfers are synchronized by this signal.

6.2.4.2 Memory Interface

Signal	Dir	Width	Description
MemClkIn	IN	1	Memory clock in. This signal is used to generate the memory interface timing.
MemRAS_N	OUT	2	Memory Row Address Strobe bus - active low.
MemCAS_N	OUT	4	Memory Column Address Strobe bus- active low.
MemClkEn	OUT	1	Memory Clock Enable. Some memories require this signal to be disabled for a certain amount of time after reset.
MemClkOut	OUT	1	Memory Clock Out. This signal is used by synchronous memory for all operations. **MemClkIn** is buffered and sent out on this pin. This helps reduce skew between this clock and the other signals.
MemWR_N	OUT	1	Memory Write – active low.
MemBA	OUT	1	Memory Bank Address. Used by multi-bank memory to select the bank the current operation is to operate on.
MemDSF	OUT	1	Memory Special Function select.
MemByteEn_N	OUT	8	Memory Byte Enable bus- active low.
MemAddress	OUT	12	Memory Address bus.
MemDataIn	IN	64	Memory Data Input bus.
MemDataOut	OUT	64	Memory Data Output bus.
MemDirRead	OUT	1	Memory Data bus Direction is Read. This signal is used to control the tri-state enable on the bidirectional memory data bus. If **MemDirRead** is active data is coming into the analyzer from the memory. If it is inactive the analyzer is driving data out to the memory.

6.2.4.3 Host Interface Signals

Signal	Dir	Width	Description
AnalyzerSel_N	IN	1	Host interface Analyzer Select - active low. **AnalyzerSel_N** is sampled on the rising edge of **MCLK**. If it is active, it signifies that the external host is attempting to access the analyzer.
HostWrite	IN	1	Write. Write is sampled on the rising edge of **MCLK**. This signal is only valid when **AnalyzerSel_N** is active. If this signal is active, the host is attempting to write to the analyzer. Inactive this signal sign signifies a read from the analyzer. It should also be used to control the direction of the host data bus if it is bidirectional.
HostBlast_N	IN	1	Burst Last – active low. **HostBlast_N** is sampled on the rising edge of **MCLK**. **HostBlast_N** tells the analyzer that the current transfer is the last transfer in this burst.
HostWait_N	IN	1	Wait – active low. **HostWait_N** is sampled on the rising edge of **MCLK**. The host asserts **HostWait_N** when it wishes to slow transfers between itself and the analyzer. This could also be used by additional interface logic to slow transfers so it can multiplex the bus down to a smaller size without additional FIFOs. If wait is active, **HostReady_N** is blocked.
AnaHostReady_N	OUT	1	Analyzer to Host Ready – active low. **AnaHostReady _N** should be sampled on the rising edge of **MCLK**. The analyzer returns **AnaHostReady _N** when the current cycle is completed. For a write operation, **AnaHostReady _N** means that the **HostDataIn** bus has been latched. For a read operation **AnaHostReady _N** means that the requested data is on the **HostDataOut** bus and is valid. **AnaHostReady _N** is blocked by **HostWait_N**.
HostAddress	IN	22	Host Address bus. HostAddress is sampled on the rising edge of **MCLK** if **AnalizerSel_N** is active. This bus defines the first address in this burst to access in the 32 Megabyte address space of the analyzer. See Section x.x.x for the Address Utilization Map.
HostByteEn_N	IN	8	Host Byte Enable bus – Active low. **HostWait_N** is sampled on the rising edge of **MCLK**.
HostDataIn	IN	64	Host Data Input bus. **HostDataIn** is sampled on the rising edge of **MCLK** if **HostWrite** is active and **HostWait_N** is inactive.
AnaHostDataOut	OUT	64	Analyzer Host Data Output bus. **AnaHostDataOut** should be sampled on the rising edge of **MCLK**. Data on this bus is valid during a read cycle when **AnaHostReady _N** is active.

6.2.4.4 Parser Interface

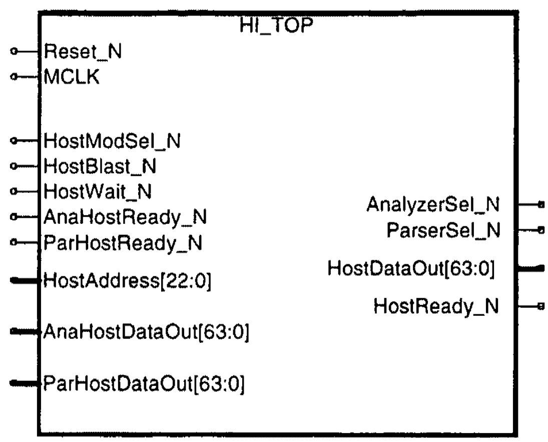
Signal	Dir	Width	Description
AnalyzerReady	OUT	1	Analyzer Ready. This signal tells the parser that the analyzer can accept data.
ParserKeyDelim	IN	1	Parser Key Delimiter. The **ParserKeyDelim** signal becomes active when the first quadword of a new key is ready to transfer to the analyzer. It goes inactive when the last quadword of the key is transferred.
ParserDataAvail	IN	1	Parser Data Available. If this signal is active the data on the ParserData bus is valid.
ParserData	IN	64	Parser Data bus.

6.3 Host Interface Module

6.3.1 Host Interface Module Highlights

- i960 style burst interface
- Easily modified for connection to other buses

6.3.2 Host Interface Symbol



6.3.3 Host Interface Module Description

The Host Interface module contains the host data multiplexer to select either the parser or the analyzer data bus. It also decodes the host address to create ParserSel_N or AanlyzerSel_N.

6.3.4 Host Interface Module Pin Out

6.3.4.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low. When this signal is active the analyzer sets it's registers to their default condition and suspends operation. It will only respond to host access cycles.
MCLK	IN	1	Module Clock. All internal and external transfers except for memory transfers are synchronized by this signal.

6.3.4.2 Host Interface Signals

Signal	Dir	Width	Description
AnalyzerSel_N	OUT	1	Host interface Analyzer Select - active low. **AnalyzerSel_N** is sampled on the rising edge of **MCLK**. If it is active, it signifies that the external host is attempting to access the analyzer.
ParserSel_N	OUT	1	Parser Select - active low. **ParserSel_N** is sampled on the rising edge of **MCLK**. If it is active, it signifies that the external host is attempting to access the parser.
HostBlast_N	IN	1	Burst Last – active low. **HostBlast_N** is sampled on the rising edge of **MCLK**. **HostBlast_N** tells the analyzer that the current transfer is the last transfer in this burst.
HostWait_N	IN	1	Wait – active low. **HostWait_N** is sampled on the rising edge of **MCLK**. The host asserts **HostWait_N** when it wishes to slow transfers between itself and the analyzer. This could also be used by additional interface logic to slow transfers so it can multiplex the bus down to a smaller size without additional FIFOs. If wait is active, **HostReady_N** is blocked.
AnaHostReady_N	IN	1	Analyzer to Host Ready – active low. **AnaHostReady _N** should be sampled on the rising edge of **MCLK**. The analyzer returns **AnaHostReady _N** when the current cycle is completed. For a write operation, **AnaHostReady _N** means that the **HostDataIn** bus has been latched. For a read operation **AnaHostReady _N** means that the requested data is on the **HostDataOut** bus and is valid. **AnaHostReady _N** is blocked by **HostWait_N**.
ParHostReady_N	IN	1	Parser to Host Ready – active low. **ParHostReady_N** should be sampled on the rising edge of **MCLK**. The parser returns **ParHostReady_N** when the current cycle is completed. For a write operation, **ParHostReady_N** means that the **HostDataIn** bus has been latched. For a read operation **ParHostReady_N** means that the requested data is on the **ParHostDataOut** bus and is valid. **ParHostReady_N** is blocked by **HostWait_N**.
HostReady_N	OUT	1	Ready – active low. **HostReady_N** should be sampled on the rising edge of **MCLK**. The module returns **HostReady_N** when the current cycle is completed. For a write operation, **HostReady_N** means that the **HostDataIn** bus has been latched. For a read operation **HostReady_N** means that the requested data is on the **HostDataOut** bus and is valid. **HostReady_N** is blocked by **HostWait_N**.
HostAddress	IN	23	Host Address bus. HostAddress is sampled on the rising edge of **MCLK** if **AnalizerSel_N** is active. This bus defines the first address in this burst to access in the 64 Megabyte address space of the analyzer. See Section x.x.x for the Address Utilization Map.

AnaHostDataOut	IN	64	Analyzer Host Data Output bus. **AnaHostDataOut** should be sampled on the rising edge of **MCLK**. Data on this bus is valid during a read cycle when **AnaHostReady _N** is active.
ParHostDataOut	IN	64	ParserHost Data Output bus. **ParHostDataOut** should be sampled on the rising edge of **MCLK**. Data on this bus is valid during a read cycle when **ParHostReady_N** is active.
HostDataOut	OUT	64	Host Data Output bus. **HostDataOut** should be sampled on the rising edge of **MCLK**. Data on this bus is valid during a read cycle when **HostReady_N** is active.

6.4 MeterFlow Compiler

6.4.1 MeterFlow Compiler Highlights

- ANSI compatible C implementation
- Simple Packet Description Language
- Technically Elite supplied Packet Description Language files for all common protocols
- Any or all protocols can be included
- Automatically generates parser module pattern recognition database
- Automatically generates slicer instructions
- Automatically generates unique protocol identifiers for use throughout system
- Automatically generates analyzer programming
- Automatically generates test input stimulus

6.4.2 MeterFlow Compiler Description

The MeterFlow Compiler generates all the information needed to program the MeterFlow accelerator. It's input is a set of files that define the protocols to recognize and the target system. It can also be used by the engineer to define the size of the databases required to support a certain set of protocols. The output of the compiler is a set of files used to program each part of the MeterFlow Accelerator.

The compiler first reads the protocol definition files defined in the protocol list file and creates a tree defining each protocols relationship to the others. Protocols with the same name are assumed to be the same. For example, FTP under UDP and TCP are condensed into a single entry linked to both parent protocols. The compiler then reads the hardware definition file or uses a default maximum definition. It then searches the protocol space to find a solution. If a solution is found that fits into the hardware constraints, the compiler outputs database in a form that can be loaded into either the testbenches, the C model, or the hardware.

If the t option is selected, the compiler will generate an input stimulus file for the testbenches. This file contains a series of packets one for each protocol in the protocol list.

6.4.3 MeterFlow Compiler Invocation

MFC <options>

6.4.3.1 Options

Option	Name	Description
i < filename>	Protocol list filename	The protocol list file contains the names of each protocol to be included in this run. The default is MeterFlow.pl. The names must match the filename prefix of the protocol definition language file associated with that protocol. For example, if the TCP protocol is to be included, and the file is called TCP.PDL, the protocol list file should contain the line: I TCP; If the children of TCP are to be included they can be added automatically by replacing the above line with: I TCP c; Child protocols can be excluded by the following line as a example: E FTP;

o <prefix>	Output file prefix	The output file prefix allows the user to change the start of the filename of the output files. The default is MeterFlow.
d <filename>	Hardware definition filename	This input file is used by the compiler to constrain processing to the available hardware resources. If the compiler cannot find a solution at the effort level it will output a set of files with the best solution it found and report an error.
e <n>	Effort	N is a number from 1 to 5. The default is 3. An effort level of 5 tells the compiler to exhaust the search space.
t	Generate input stimulus file	This option generates a file that can be run through the C model to generate expected output data. Then both files can be run through the testbenches for automated testing of the modules.

6.5 MeterFlow C Model

6.5.1 MeterFlow C Model Highlights

- ANSI compatible C implementation
- Models the MeterFlow Accelerator modules
- Outputs expected data for the testbenches
- Excepts the same input files as the testbenches

6.5.2 MeterFlow C Model Description

The MeterFlow C Model reads the same files used by the modules and emulates them. It is used to generate expected data for the automated testbenches included with the modules.

6.5.3 MeterFlow C Model Invocation

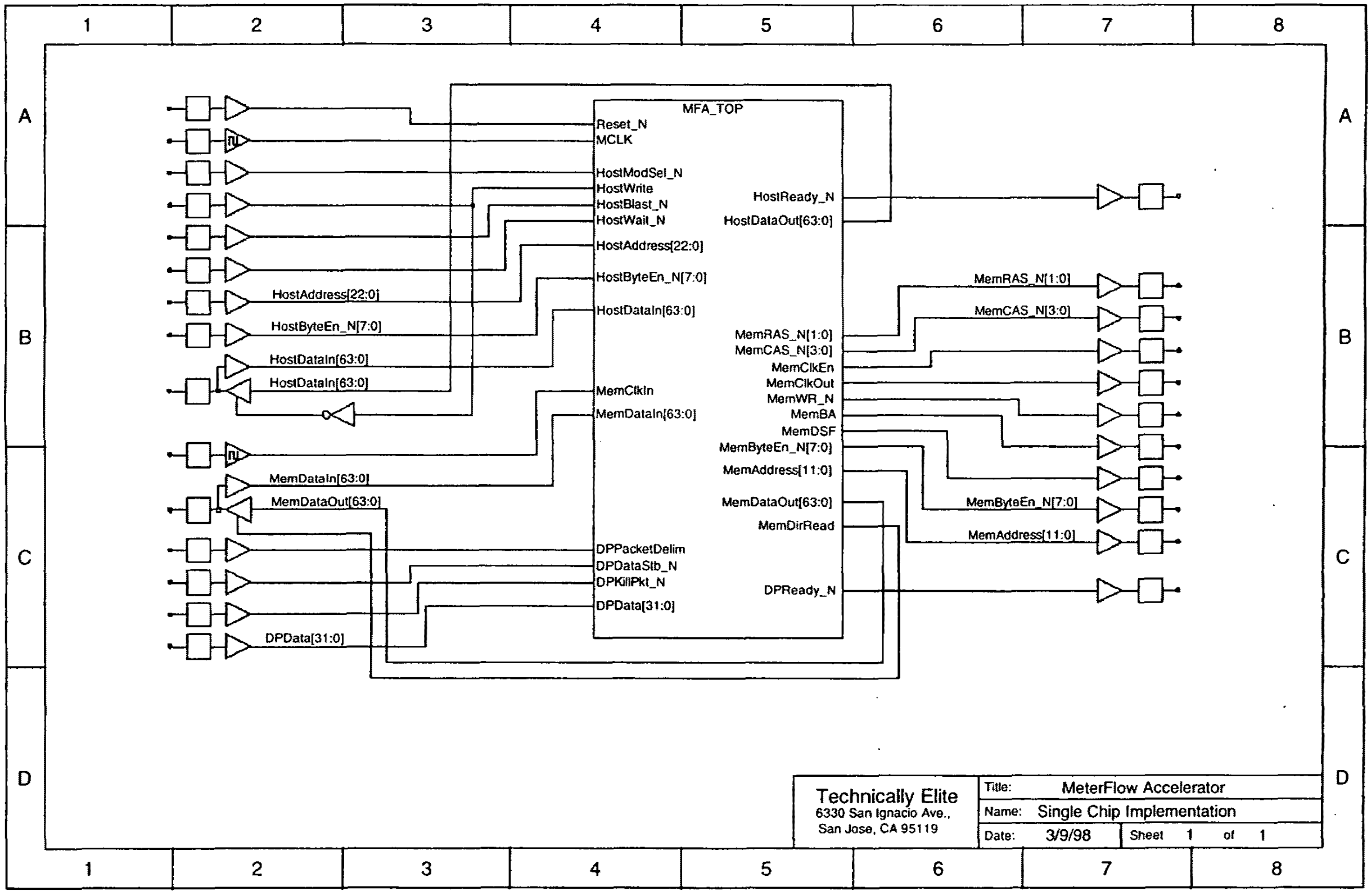
MCM <options>

6.5.3.1 Options

Option	Name	Description
i < filename>	Input filename prefix	The input file prefix allows the user to change the start of the filename of the input files. The default is MeterFlow.
o <prefix>	Output file prefix	The output file prefix allows the user to change the start of the filename of the output files. The default is MeterFlow.
d <filename>	Hardware definition filename	This input file is used by the C model to emulate the available hardware resources.

7 MeterFlow Accelerator Single Chip Implementation Top Level Schematic

[redacted]



Technically Elite CONFIDENTIAL

- **Exhibit A2:** Technically Elite MeterFlow Accelerator Parser Module Specification (Document MFAParser.pdf)

Technically Elite MeterFlow Accelerator Parser Module Specification

Not For External Release!

Revision History		
Version	Date	Description
0.9	[redacted]	First release
1.0		Rev 1

0 Table of Contents

1 Introduction

This document is designed to be the repository for all information related to the MeterFlow Accelerator Parser Module. This specification is designed to provide the engineer with enough information to fully implement the module. There will be revisions during and after the implementation process that will be reflected in this document.

Each part of this specification describes a different aspect of the module. It concentrates on the interfaces between the parser module and the other parts of the system. The other parts of the system include the analyzer module, the host interface module and importantly the software that models, programs and tests the system The key to a successful implementation is the interfaces between modules and between sub-module and sub-module. Each interface is described in detail. Any changes to the interfaces may affect the entire module and even the entire system. Care must be taken that each interface is understood completely before implementation is begun.

2 Technically Elite MeterFlow Accelerator Parser Module Highlights

- Synthesizable modules written in both the Verilog and VHDL
- Scalable architecture for any size switch or probe
- Can recognize over 2000 different protocols
- Extensible to new protocols
- Recognizes encapsulations
- Builds key and payload data structure for analyzer (flow key)
- Scaleable protocol pattern recognition engine
- At 62.5 MegaHertz can process up to 1.5 MegaPackets per second
- Accepts protocol database output from MeterFlow compiler

3 Architectural Overview

The parser module consist of two main sub-modules. These are the pattern recognition engine (PRE) and the slicer. The PRE analyzes the packet and the slicer builds the flow key from the packet and instructions from the pattern recognition engine .The parser has been split into two parts for several reasons. First and foremost, the split correctly partitions the functions to allow maximum reuse of silicon across the over two thousand protocols that can be supported. Another advantage of the split architecture is that the compiler can analyze the three dimensional space occupied by the offset, level, and pattern data of the specified protocols and compact the databases used in the parser module. The set of specified protocols defines a tree of linked nodes. Each protocol is either a parent node or a terminal node. A protocol is a parent node if it links to other protocols that can be contained in it. For example IP is a parent to UDP. Protocols can be the children of several parents. If a unique node was generated for each of the possible parent/child trees, the database would explode exponentially. Instead, child nodes are shared among multiple parents thus compacting the database.. Finally the PRE can be used on it's own when only protocol recognition is required.

The parser module pouches the network data through the DataPort interface. The data is first processed by the pattern recognition engine. This engine consists of a comparison engine and a database. The comparison engine has a first stage that checks the protocol type field to determine if it is an 802.3 packet and the field should be treated as a length. If it is not a length, the protocol is checked in the second stage. This is the only protocol level that is not programmable. This is because the detection of the protocol at this level is simple and well defined. It is implemented with partial CAMs that return a node identifier if hit. This second stage has two full sixteen bit CAMs defined for future protocol additions. After this detection is completed the engine initializes Current Offset Pointer (COP) to the next part of the packet that needs to be checked. The node identifier from the previous stage and the data pointed to by the COP are used by the PRE to lookup an entry in the database. As each protocol is recognized, the pattern recognition engine emits a unique protocol identifier. It also emits a process code that the slicer uses to build the flow key. This process is repeated until the node identifier's Terminal bit is set. At that point the PRE has completely recognized the protocols in the packet and readies itself for the next packet.

The slicer extracts information from the packet to build the flow key. For example, it will extract the source and destination addresses from the packet and pack them into the flow key data structure. It may also process certain parts of the packet to speed up flow processing performed by the analyzer. It will build a hash value from certain parts of the packet to speed looking up the flow in the analyzers' database. The slicer transfers data from it's input Buffer to it's output Buffer based on the sequence of instructions in it's instruction database. When the PRE recognizes a protocol it outputs both the protocol identifier and a process code to the slicer. The protocol identifier is added to the flow key and the process code is used to fetch the first instruction from the instruction database. Instructions consist an operation code and usually source and destination offsets as well as a length. The offsets and length are in bytes. A typical operation is the MOVE instruction. This instruction tells the slicer to copy n bytes data unmodified from the input Buffer to the output Buffer. The slicer contains a byte-wise barrel shifter so that the bytes moved can be packed into the flow key. The slicer contains another instruction called HASH. This instruction tells the slicer to copy from the input Buffer to the HASH generator. The result from the HASH generator is always written into the first two bytes of the flow key. It is used to accelerate the lookup of the flow in the analyzers flow database. Once the flow key is completed, the slicer transfers it to the analyzer for further processing.

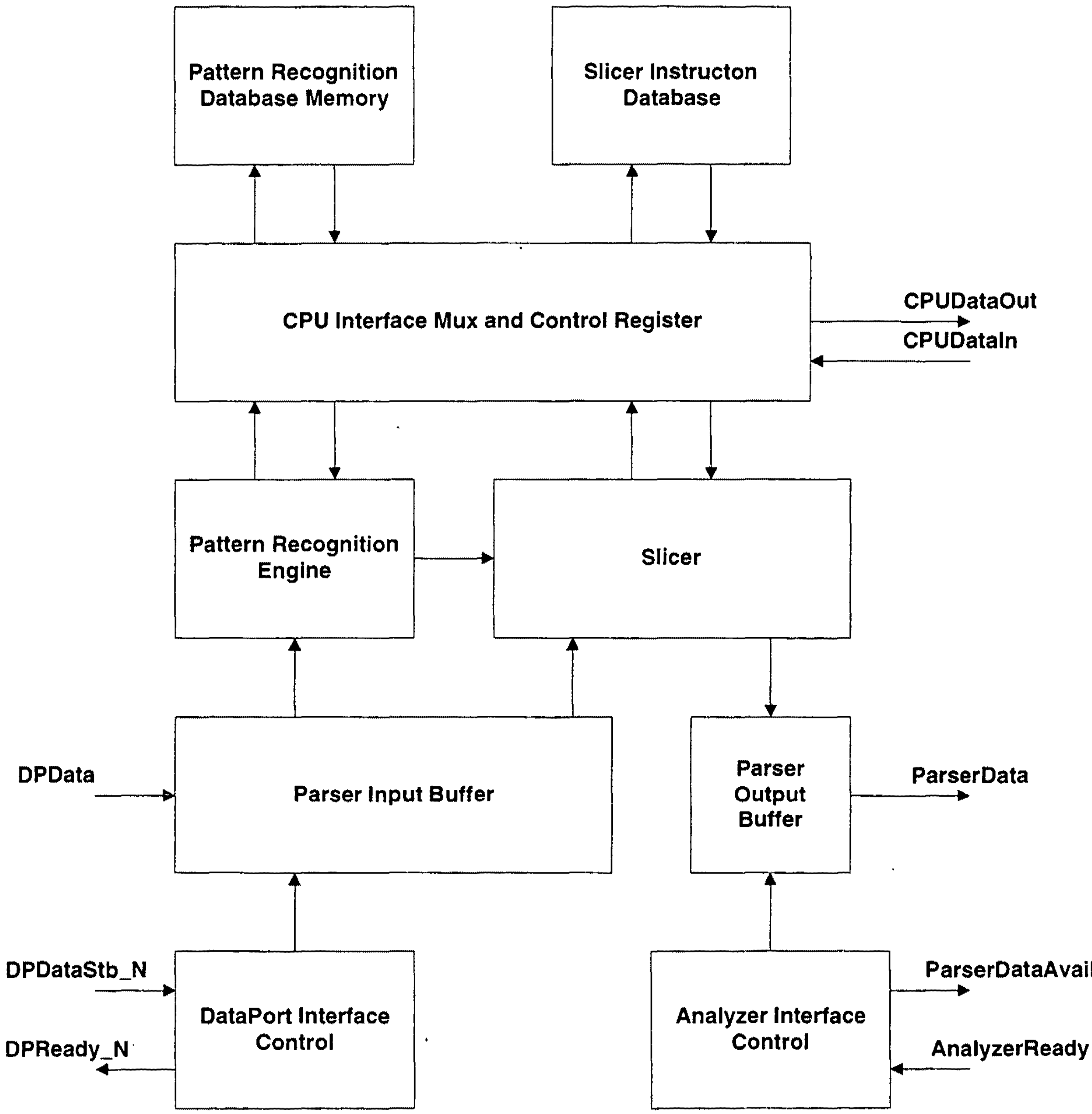
The parser module databases can reside in ROM or RAM. If the databases are in a RAM the parser can be programmed to recognize new protocols or a different set of protocols.

3.1 Bandwidth requirements

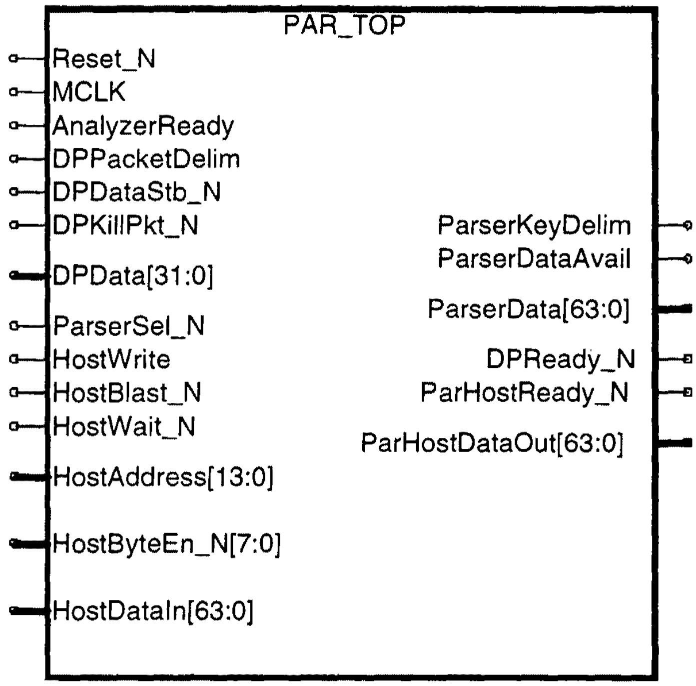
The target throughput for the MeterFlow Accelerator running at 62.5 Megahertz is 1.5 million packets per second (PPS). This is the sustained maximum throughput of a single Gigabit channel. At this rate the parser module has 41.6 cycles to process each packet. In order to reduce the need for front end buffering external to the parser module, the architecture has been designed to complete the protocol recognition generation in no more than 36 cycles. Since there could be up to 12 different protocols in each to be processed, the parser module has been designed to average three cycles per protocol. This is the very worst case because a packet that has twelve levels of protocols in it will most likely be much larger than the minimum packet size. This can be used as to advantage again in the reduction of external buffering. The slicer must also complete the flow key generation within 36 cycles to keep the system in balance and unstalled. This however can be extended if the payload copying instructions run to there maximum values.

The average packet will have between 4 and 5 levels of protocol with no encapsulations. At three cycles per protocol the PRE will use only 15 cycles to complete a packet. This means that the PRE has a typical sustained throughput of over three million packets per second.

3.2 Architectural Block Diagram



4 Top Level MeterFlow Accelerator Parser Module Symbol



5 MeterFlow Accelerator Parser Module Top Level Pin Descriptions

5.1.1.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low. When this signal is active the parser sets it's registers to their default condition and suspends operation. It will only respond to host access cycles. The DataPort interface will keep **DPReady_N** active to avoid problems for the external circuitry.
MCLK	IN	1	Module Clock. All internal and external transfers except for memory transfers are synchronized by this signal.

5.1.1.2 Analyzer Interface

Signal	Dir	Width	Description
AnalyzerReady	IN	1	Analyzer Ready. This signal tells the parser that the analyzer can accept data.
ParserKeyDelim	OUT	1	Parser Key Delimiter. The **ParserKeyDelim** signal becomes active when the first quadword of a new key is ready to transfer to the analyzer. It goes inactive when the last quadword of the key is transferred.
ParserDataAvail	OUT	1	Parser Data Available. If this signal is active the data on the **ParserData** bus is valid.
ParserData	OUT	64	Parser Data bus.

5.1.1.3 DataPort Interface

Signal	Dir	Width	Description
DPPacketDelim	IN	1	DataPort Packet Delimiter. This signal should be driven active when the external logic wants to send a packet to the parser. **DPPacketDelim** should remain active during the entire packet transfer. **DPPacketDelim** must go inactive for one clock between packets.
DPDataStb_N	IN	1	DataPort Data Strobe. When active, this signal tells the parser that data on the **DPData** bus is valid. If **DPReady_N** was inactive at the end of the previous cycle, **DPDataStb_N** should not be driven active. If **DPReady_N** goes inactive in the same cycle as **DPDataStb_N**, then the parser will latch the incoming data so that no data is lost.
DPKillPkt_N	IN	1	DataPort Kill Packet. If this signal becomes active while **DPPacketDelim** is active, the parser will attempt to stop processing the current packet and flush it's input Buffer. If however, parsing of the packet is completed, the packet will not be able to be recalled. This should only be a problem in a 'cut through' implementation.
DPReady_N	OUT	1	DataPort Ready – active low. This signal when driven active means that the parser can accept new data. If however the parser's input Buffer is filled, **DPReady_N** will be driven inactive. To prevent overruns, **DPReady_N** will go inactive when the parser can actually accept one more data transfer.
DPData	IN	32	DataPort Data bus.

5.1.1.4 Host Interface Signals

Signal	Dir	Width	Description
ParserSel_N	IN	1	Parser Select - active low. **ParserSel_N** is sampled on the rising edge of **MCLK**. If it is active, it signifies that the external host is attempting to access the parser.
HostWrite	IN	1	Write. Write is sampled on the rising edge of **MCLK**. This signal is only valid when **ParserSel _N** is active. If this signal is active, the host is attempting to write to the parser. Inactive this signal sign signifies a read from the parser.
HostBlast_N	IN	1	Burst Last – active low. **HostBlast_N** is sampled on the rising edge of **MCLK**. **HostBlast_N** tells the parser that the current transfer is the last transfer in this burst.
HostWait_N	IN	1	Wait – active low. **HostWait_N** is sampled on the rising edge of **MCLK**. The host asserts **HostWait_N** when it wishes to slow transfers between itself and the parser.
ParHostReady_N	OUT	1	Parser to Host Ready – active low. **ParHostReady_N** should be sampled on the rising edge of **MCLK**. The parser returns **ParHostReady_N** when the current cycle is completed. For a write operation, **ParHostReady_N** means that the **HostDataIn** bus has been latched. For a read operation **ParHostReady_N** means that the requested data is on the **ParHostDataOut** bus and is valid. **ParHostReady_N** is blocked by **HostWait_N**.
HostAddress	IN	13	Host Address bus. HostAddress is sampled on the rising edge of **MCLK** if **ParserSel_N** is active. This bus defines the first address in this burst to access in the 64 Kilobyte address space of the Parser. See Section x.x.x for the Address Utilization Map.
HostByteEn_N	IN	8	Host Byte Enable bus – Active low. **HostWait_N** is sampled on the rising edge of **MCLK**.
HostDataIn	IN	64	Host Data Input bus. **HostDataIn** is sampled on the rising edge of **MCLK** if **HostWrite** is active and **HostWait_N** is inactive.
ParHostDataOut	OUT	64	ParserHost Data Output bus. **ParHostDataOut** should be sampled on the rising edge of **MCLK**. Data on this bus is valid during a read cycle when **ParHostReady_N** is active.

6 MeterFlow Accelerator Parser Module Top Level VHDL Entity

```
entity PAR_TOP is
        Port
        (
                AnalyzerReady : In    std_logic;
                DPDataStb_N : In    std_logic;
                DPData : In    std_logic_vector (31 downto 0);
                DPKillPkt_N : In    std_logic;
                DPPacketDelim : In    std_logic;
                HostAddress : In    std_logic_vector (13 downto 0);
                HostBlast_N : In    std_logic;
                HostByteEn_N : In    std_logic_vector (7 downto 0);
                HostDataIn : In    std_logic_vector (63 downto 0);
                HostWait_N : In    std_logic;
                HostWrite : In    std_logic;
                MCLK : In    std_logic;
                ParserSel_N : In    std_logic;
                Reset_N : In    std_logic;
                DPReady_N : Out   std_logic;
                ParHostDataOut : Out   std_logic_vector (63 downto 0);
                ParHostReady_N : Out   std_logic;
                ParserDataAvail : Out   std_logic;
                ParserData : Out   std_logic_vector (63 downto 0);
                ParserKeyDelim : Out   std_logic
        );
end PAR_TOP;
```

7 MeterFlow Accelerator Parser Module Top Level Verilog Module

```
module par_top( AnalyzerReady, DPData, DPDataStb, DPKillPkt_N, DPPacketDelim, DPReady_N,
          HostAddress, HostBlast_N, HostDataIn, HostWait_N, HostWrite,
          MCLK, ParHostDataOut, ParHostReady_N, ParserData,
          ParserDataAvail, ParserKeyDelim, ParserSel_N, Reset_N );
 input AnalyzerReady;
 input [63:0] DPData;
 input DPDataStb, DPKillPkt_N, DPPacketDelim;
output DPReady_N;
 input [12:0] HostAddress;
 input HostBlast_N;
 input [63:0] HostDataIn;
 input HostWait_N, HostWrite, MCLK;
output [63:0] ParHostDataOut;
output ParHostReady_N;
output [63:0] ParserData;
output ParserDataAvail, ParserKeyDelim;
 input ParserSel_N, Reset_N;
  wire [8:0] DPICAdd;
  wire [63:0] PIBuSIData;
```

```
 wire [8:0] SIPBAdd;
 wire [63:0] PIBuPREData;
 wire [8:0] PREnPIBAdd;
 wire [29:0] CMCoSIData;
 wire [8:0] SIAdd;
 wire [3:0] PREnSIProtocol;
 wire [63:0] SIPOBData;
 wire [5:0] PREnSICommand;
 wire [8:0] SIPOBAdd;
 wire [8:0] CMCoPRDAdd;
 wire [22:0] CMCoPRDData;
 wire [3:0] BaseOffset;
 wire [22:0] CMCoPREData;
 wire [8:0] PREAdd;
 wire [22:0] PRDData;
 wire [29:0] SIData;
 wire [8:0] CMCoSIDAdd;
 wire [8:0] AICPOBAdd;
 wire [29:0] SIDData;
 wire [29:0] CMCoSIDData;
 wire [8:0] AICoPOBAdd;
 wire [8:0] SIFlowKeySize;
 wire [8:0] SIPIBAdd;
wire AICDone;
wire CMCoSIDWr;
wire CMCoPRDWr;
wire PREnSIEn;
wire SIWrStb;
wire SIDone;
wire PREDone;
wire DPICWrStb;
wire DPICDone;
wire ParserEn;

AIC I11 ( .AICDone(AICDone), .AICoPOBAdd(AICoPOBAdd[8:0]),
    .AnalyzerReady(AnalyzerReady), .MCLK(MCLK),
    .ParserDataAvail(ParserDataAvail), .ParserEn(ParserEn),
    .ParserKeyDelim(ParserKeyDelim), .Reset_N(Reset_N), .SIDone(SIDone),
    .SIFlowKeySize(SIFlowKeySize[8:0]) );
PRE I10 ( .BaseOffset(BaseOffset[3:0]), .CMCoPREData(CMCoPREData[22:0]),
    .MCLK(MCLK), .ParserEn(ParserEn), .PIBuPREData(PIBuPREData[63:0]),
    .PREAdd(PREAdd[8:0]), .PREDone(PREDone), .PREnPIBAdd(PREnPIBAdd[8:0]),
    .PREnSICommand(PREnSICommand[5:0]), .PREnSIEn(PREnSIEn),
    .PREnSIProtocol(PREnSIProtocol[3:0]), .Reset_N(Reset_N) );
DPIC I1 ( .DPDataStb(DPDataStb), .DPICAdd(DPICAdd[8:0]), .DPICDone(DPICDone),
    .DPICWrStb(DPICWrStb), .DPKillPkt_N(DPKillPkt_N),
    .DPPacketDelim(DPPacketDelim), .DPReady_N(DPReady_N), .MCLK(MCLK),
    .ParserEn(ParserEn), .PREDone(PREDone), .Reset_N(Reset_N) );
Slicer I2 ( .CMCoSIData(CMCoSIData[29:0]), .MCLK(MCLK), .ParserEn(ParserEn),
     .PIBuSIData(PIBuSIData[63:0]), .PREDone(PREDone),
     .PREnSICommand(PREnSICommand[5:0]), .PREnSIEn(PREnSIEn),
     .PREnSIProtocol(PREnSIProtocol[3:0]), .Reset_N(Reset_N),
     .SIAdd(SIAdd[8:0]), .SIDone(SIDone),
     .SIFlowKeySize(SIFlowKeySize[8:0]), .SIPIBAdd(SIPIBAdd[8:0]),
```

```
    .SIPOBAdd(SIPOBAdd[8:0]), .SIPOBData(SIPOBData[63:0]),
    .SIWrStb(SIWrStb) );
SID I3 ( .CMCoSIDAdd(CMCoSIDAdd[8:0]), .CMCoSIDData(CMCoSIDData[29:0]),
   .CMCoSIDWr(CMCoSIDWr), .SIDData(SIDData[29:0]) );
PRD I4 ( .CMCoPRDAdd(CMCoPRDAdd[8:0]), .CMCoPRDData(CMCoPRDData[22:0]),
   .CMCoPRDWr(CMCoPRDWr), .PRDData(PRDData[22:0]) );
POB I5 ( .AICDone(AICDone), .AICoPOBAdd(AICoPOBAdd[8:0]), .MCLK(MCLK),
   .ParserData(ParserData[63:0]), .ParserEn(ParserEn), .Reset_N(Reset_N),
   .SIDone(SIDone), .SIPOBAdd(SIPOBAdd[8:0]), .SIPOBData(SIPOBData[63:0]),
   .SIWrStb(SIWrStb) );
PIB I6 ( .DPData(DPData[63:0]), .DPICAdd(DPICAdd[8:0]), .DPICDone(DPICDone),
   .DPICWrStb(DPICWrStb), .MCLK(MCLK), .ParserEn(ParserEn),
   .PIBuPREData(PIBuPREData[63:0]), .PIBuSIData(PIBuSIData[63:0]),
   .PREDone(PREDone), .PREnPIBAdd(PREnPIBAdd[8:0]), .Reset_N(Reset_N),
   .SIDone(SIDone), .SIPIBAdd(SIPBAdd[8:0]) );
CMC I8 ( .BaseOffset(BaseOffset[3:0]), .CMCoPRDAdd(CMCoPRDAdd[8:0]),
   .CMCoPRDData(CMCoPRDData[22:0]), .CMCoPRDWr(CMCoPRDWr),
   .CMCoPREData(CMCoPREData[22:0]), .CMCoSIDAdd(CMCoSIDAdd[8:0]),
   .CMCoSIDData(CMCoSIDData[8:0]), .CMCoSIDWr(CMCoSIDWr),
   .CMCoSIData(CMCoSIData[29:0]), .HostAddress(HostAddress[12:0]),
   .HostBlast_N(HostBlast_N), .HostDataIn(HostDataIn[63:0]),
   .HostWait_N(HostWait_N), .HostWrite(HostWrite), .MCLK(MCLK),
   .ParHostDataOut(ParHostDataOut[63:0]),
   .ParHostReady_N(ParHostReady_N), .ParserEn(ParserEn),
   .ParserSel_N(ParserSel_N), .PRDData(PRDData[22:0]),
   .PREAdd(PREAdd[8:0]), .Reset_N(Reset_N), .SIDData(SIData[29:0]),
   .SIAdd(SIAdd[8:0]) );

endmodule // par_top
```

8 MeterFlow Accelerator Parser Module Top Level Schematic

Insert Schematic Here

9 Parser Module Constants Files

The parser module constants files contain a list of constants used to allow rapid configuration of the module. For example the size of the slicers instruction database data bus is defined as :

Verilog

```
‘define PAR_SLI_DWIDTH 23 // Parser Slicer Instruction Database Data Bus Width
```

VHDL

```
constant PAR_SLI_DWIDTH : integer :=  23; -- Parser Slicer Instruction Database Data Bus Width
```

9.1 Parser module Verilog Constants File – ParserConstants.v

```
‘define PAR_COM_SHIFT 3 // Parser Command Shift
‘define PAR_SLI_DWIDTH 23
‘define PAR_DP_DWIDTH 32 // Parser Data Port Data Bus Width
‘define PAR_PIB_DWIDTH 64 // Parser Input Buffer Data Bus Width
‘define PAR_PIB_AWIDTH 9 // Parser Input Buffer Address Bus Width
‘define PAR_PRD_AWIDTH 9 // Parser Pattern Recognition Database Address Bus Width
‘define PAR_PRD_DWIDTH 23 // Parser Pattern Recognition Database Data Bus Width
‘define PAR_SID_AWIDTH 9 // Parser Slicer Instruction Database Address Bus Width
‘define PAR_SID_DWIDTH 30 // Parser Slicer Instruction Database Data Bus Width
‘define PAR_POB_DWIDTH 64 // Parser Output Buffer Data Bus Width
‘define PAR_POB_AWIDTH 9 // Parser Output Buffer Address Bus Width
‘define PAR_BASE_OFF_WIDTH 4 // Parser Base Offset Width
‘define PAR_HOST_AWIDTH 13 // Parser Host Address Bus Width
‘define PAR_HOST_BE_WIDTH 8 // Parser Host Byte Enable Bus Width
‘define PAR_HOST_DWIDTH 64 // Parser Host Data Bus Width
‘define PAR_PRE_COM_WIDTH 6 // Parser Command Width
‘define PAR_COM_CT_WIDTH 4 // Parser Command Count Width
‘define PAR_PRE_PRO_WIDTH 4 // Parser
‘define PAR_CONTROL_REG_SIZE 5 // Parser Control Register Size
‘define PAR_H_SIDDELTA 34
‘define PAR_H_PRDDELTA 41
‘define PAR_H_CRDELTA 59 // CAN'T BE NESTED!
‘define PAR_SL_FKS_WIDTH 9 // Parser Slicer Flow Key Size Width
```

9.2 Parser module VHDL Constants File – ParserConstants.vhd

Insert ParserConstants.vhd here

10 Sub-module Descriptions

10.1 Pattern Recognition Engine Sub-module – PRE

10.1.1 Symbol

10.1.2 Highlights

- Scaleable protocol pattern recognition engine
- Supports from 1 to 2048 simultaneous unique protocol patterns
- At 62.5 MegaHertz can process up to 1.5 MegaPackets per second
- Accepts protocol database output from MeterFlow compiler

10.1.3 Description

The Pattern Recognition Engine module searches it's database and the packet in order to recognize the protocols the packet contains. The database consists of a series of linked lookup tables. Each lookup table uses eight bits of addressing. The first lookup table is always at address zero. The Pattern Recognition Engine uses the **BaseOffset** from the control register to start the comparison. It loads this value into the Current Offset Pointer (COP). It then reads the byte at **BaseOffset** from the Parser Input Buffer and uses it as an address into the first lookup table.

Each lookup table returns a word that links to another lookup table or it returns a terminal flag. If the lookup produces a recognition event the database also returns a command for the Slicer. Finally it returns the value to add to the COP.

10.1.4 Search Algorithm Psuedo-code

10.1.5 Implementation Information

10.1.5.1 Database Word Definition

Bit	Description
1:0	Opcode 00 Terminal Node found 01 Intermediate Node 10 Ending Terminal Node found
*	Next Lookup table * uses PAR_PRE_LU_WIDTH
*	Slicer Command * uses PAR_PRE_COM_WIDTH
*	Mask * uses PAR_PRE_MASK_WIDTH

10.1.6 File Names

Top: PRE.v(hd)

Uses: ParserConstants.v(hd)

10.1.7 Pin Descriptions

10.1.7.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low.
MCLK	IN	1	Module Clock.
PREDone	OUT	1	Pattern Recognition Engine Done.
ParserEn	IN	1	Parser Enable bit from control register

10.1.7.2 Slicer Interface

Signal	Dir	Width	Description
PREnSlEn	OUT	1	Pattern Recognition Engine to Slicer Enable
PREnSlCommand	OUT	*	Pattern Recognition Engine to Slicer Command bus * uses PAR_PRE_COM_WIDTH
PREnSlProtocol	OUT	*	Pattern Recognition Engine to Slicer Protocol bus * uses PAR_PRE_PRO_WIDTH

10.1.7.3 CPU Interface MUX Interface

Signal	Dir	Width	Description
PREAdd	OUT	*	Pattern Recognition Engine Address bus * uses PAR_PRD_AWIDTH
BaseOffset	IN	4	Base Offset. This is the first offset the Pattern Recognition Engine will check.
CMCoPREData	IN	*	CMC to Pattern Rcognition Engine Data bus * uses PAR_PRD_DWIDTH

10.1.7.4 Parser Input Buffer Interface

Signal	Dir	Width	Description
PREnPIBAdd	OUT	*	Pattern Recognition Engine Parser Input Buffer Address bus. * Uses PAR_PIB_AWIDTH
PIBuPREData	IN	*	Parser Input Buffer to Pattern Recognition Engine Data bus. * Uses PAR_PIB _DWIDTH

10.1.8 Verilog Module

```
/*
PRE.v
PRE - Pattern Recognition Module
```

```
*/
'include "ParserConstants.v"

module PRE(Reset_N, MCLK ,ParserEn, PREDone, PREnSlEn, PREnSlCommand, PREnSlProtocol,
PREAdd, BaseOffset, CMCoPREData, PREnPIBAdd, PIBuPREData);

// General Interface Interface
input Reset_N;
input MCLK;
input ParserEn;
output PREDone;
// Slicer Interface
output PREnSlEn;
output ['PAR_PRE_COM_WIDTH-1 : 0] PREnSlCommand;
output ['PAR_PRE_PRO_WIDTH-1 : 0] PREnSlProtocol;
// CMC Interface
output ['PAR_PRD_AWIDTH-1 : 0] PREAdd;
input ['PAR_BASE_OFF_WIDTH-1 : 0] BaseOffset;
input ['PAR_PRD_DWIDTH-1 : 0] CMCoPREData;
// Parser Input Buffer Interface
output ['PAR_PIB_AWIDTH-1 : 0] PREnPIBAdd;
input ['PAR_PIB_DWIDTH-1 : 0] PIBuPREData;
```

10.2 Slicer Sub-module

10.2.1 Symbol

10.2.2 Description

The Slicer cuts up the packet to build the flow key. The Slicer module accepts commands from the Pattern Recognition Engine. Based on the command received, the Slicer either transfers data from the Parser Input Buffer to the Parser Output Buffer or it transfers data from the Parser Input Buffer to it's internal hash generator. It contains a buffer that FIFO's up the commands. When the Pattern Recognition Engine asserts **PREDone** the Slicer completes any pending commands, transfers the hash to the Parser Output Buffer and asserts **SlDone**.

10.2.2.1 Instruction Word Definition

Bit	Description
1:0	Opcode 00 Nop 01 Move 10 Hash 11 Done
*	Source Address * uses PAR_PIB_AWIDTH
*	Destination Address * uses PAR_POB_AWIDTH
*	Length * uses PAR_SL_LEN_WIDTH

10.2.3 Implementation Information

The Slicer contains a byte wise barrel shifter that is used to pack data into the flow key. A Moore finite state machine controls the execution of commands. The command comes into the Slicer and is shifted to provide an address. The Slicer uses this address to read the Slicer Instruction Database.

10.2.4 File Names

Top: Slicer.v(hd)
Uses: ParserConstants.v(hd)

10.2.5 Pin Descriptions

10.2.5.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low.
MCLK	IN	1	Module Clock.
SlDone	OUT	1	Slicer Done. This output is used to tell the rest of the Parser that the Slicer has finished processing the current packet.
ParserEn	IN	1	Parser Enable bit from control register

10.2.5.2 Parser Input Buffer Interface

Signal	Dir	Width	Description
SlPIBAdd	OUT	*	Slicer Parser Input Buffer Address bus. * Uses PAR_PIB_AWIDTH
PIBuSlData	IN	*	Parser Input Buffer to Slicer Data bus. * Uses PAR_PIB_DWIDTH

10.2.5.3 Parser Output Buffer Interface

Signal	Dir	Width	Description
SlPOBAdd	OUT	*	Slicer Parser Output Buffer Address bus. * Uses PAR_POB_AWIDTH
SlWrStb	OUT	1	Slicer Write Strobe.
SlPOBData	OUT	*	Slicer to Parser Output Buffer Data bus. * Uses PAR_POB_DWIDTH

10.2.5.4 CPU Interface MUX Interface

Signal	Dir	Width	Description
SiAdd	OUT	*	Slicer Address bus * uses PAR_SID_AWIDTH
CMCoSlData	IN	*	CMC to Slicer Data bus * uses PAR_SID_DWIDTH

10.2.5.5 Pattern Recognition Engine Interface

Signal	Dir	Width	Description
PREnSlEn	IN	1	Pattern Recognition Engine to Slicer Enable
PREDone	IN	1	Pattern Recognition Engine Done.
PREnSlCommand	IN	*	Pattern Recognition Engine to Slicer Command bus * uses PAR_PRE_COM_WIDTH
PREnSlProtocol	IN	*	Pattern Recognition Engine to Slicer Protocol bus

			* uses PAR_PRE_PRO_WIDTH

10.2.5.6 Analyzer Interface Control Interface

Signal	Dir	Width	Description
SlFlowKeySize	OUT	*	Pattern Recognition Engine to Slicer Protocol bus * uses PAR_SL_FKS_WIDTH

10.2.6 Verilog Module

```
/*
Slicer.v
Slicer Module



*/
'include "ParserConstants.v"

module Slicer(Reset_N, MCLK ,ParserEn, SlDone, SlPIBAdd, PIBuSlData, SlPOBAdd, SlWrStb,
SlPOBData, SlAdd, CMCoSlData, PREnSlEn, PREDone, PREnSlCommand, PREnSlProtocol,
SlFlowKeySize);

// General Interface Interface
input Reset_N;
input MCLK;
input ParserEn;
output SlDone;
// Parser Input Buffer Interface
output ['PAR_PIB_AWIDTH-1 : 0] SlPIBAdd;
input ['PAR_PIB_DWIDTH-1 : 0] PIBuSlData;
// Parser Output Buffer Interface
output ['PAR_POB_AWIDTH-1 : 0] SlPOBAdd;
output SlWrStb;
output ['PAR_POB_DWIDTH-1 : 0] SlPOBData;
// CMC Interface
output ['PAR_SID_AWIDTH-1 : 0] SlAdd;
output ['PAR_SID_DWIDTH-1 : 0] CMCoSlData;
// Pattern Recognition Engine Interface
input PREnSlEn;
input PREDone;
input ['PAR_PRE_COM_WIDTH-1 : 0] PREnSlCommand;
input ['PAR_PRE_PRO_WIDTH-1 : 0] PREnSlProtocol;
// AIC
output ['PAR_SL_FKS_WIDTH-1 : 0] SlFlowKeySize;
```

10.3 Pattern Recognition Database Sub-module - PRD

10.3.1 Symbol

10.3.2 Highlights

- Scaleable implementation
- Wraps either RAM or ROM instantiation or can be synthesized latches

10.3.3 Description

The Pattern Recognition Database Memory module is a wrapper for the storage medium used to hold the pattern recognition database. Only the CPU can write this memory.

10.3.4 Implementation Information

The module can be synthesized or a RAM or ROM cell can be instantiated into the wrapper.

10.3.5 File Names

Top: PRD.v(hd)
Uses: ParserConstants.v(hd),GenericRAM.v(hd)

10.3.6 Pin Descriptions

10.3.6.1 CPU Interface MUX Interface

Signal	Dir	Width	Description
CMCoPRDWr	IN	1	CMC to PRD Write Strobe
CMCoPRDAdd	IN	*	CMC to PRD Address bus * uses PAR_PRD_AWIDTH
PRDData	OUT	*	PRD Data bus * uses PAR_PRD_DWIDTH
CMCoPRDData	IN	*	CMC to PRD Data bus * uses PAR_PRD_DWIDTH

10.3.7 Verilog Module

```
/*
PRD.v

*/
'include "ParserConstants.v"

module PRD(CMCoPRDData, PRDData, CMCoPRDAdd, CMCoPRDWr);

input ['PAR_PRD_AWIDTH-1 : 0] CMCoPRDAdd;
input ['PAR_PRD_DWIDTH-1 : 0] CMCoPRDData;
output ['PAR_PRD_DWIDTH-1 : 0] PRDData;
input CMCoPRDWr;
```

10.4 Slicer Instruction Database Sub-module -SID

10.4.1 Symbol

10.4.2 Highlights

- Scaleable implementation
- Wraps either RAM or ROM instantiation or can be synthesized latches

10.4.3 Description

The Slicer Instruction Database module is a wrapper for the storage medium used to hold the pattern recognition database. Only the CPU can write this memory.

10.4.4 Implementation Information

The module can be synthesized or a RAM or ROM cell can be instantiated into the wrapper.

10.4.5 File Names

Top: SID.v(hd)
Uses: ParserConstants.v(hd),GenericRAM.v(hd)

10.4.6 Pin Descriptions

10.4.6.1 CPU Interface MUX Interface

Signal	Dir	Width	Description
CMCoSIDWr	IN	1	CMC to SID Write Strobe
CMCoSIDAdd	IN	*	CMC to SID Address bus * uses PAR_SID_AWIDTH
SIDData	OUT	*	SID Data bus * uses PAR_SID_DWIDTH
CMCoSIDData	IN	*	CMC to SID Data bus * uses PAR_SID_DWIDTH

10.4.7 Verilog Module

```
/*
SID.v


*/
'include "ParserConstants.v"

module SID(CMCoSIDData, SIDData, CMCoSIDAdd, CMCoSIDWr);

input ['PAR_SID_AWIDTH-1 : 0] CMCoSIDAdd;
input ['PAR_SID_DWIDTH-1 : 0] CMCoSIDData;
output ['PAR_SID_DWIDTH-1 : 0] SIDData;
```

input CMCoSIDWr;

10.5 CPU Interface MUX and Control Register Sub-module - CMC

10.5.1 Symbol

10.5.2 Description

The CPU Interface MUX and Control Register module controls the communication between the external CPU and the Parser. The CMC contains a MUX for the CPU read back. It also contains the control register for the Parser.

10.5.3 File Names

Top: CMC.v(hd)
Uses: ParserConstants.v(hd)

10.5.4 Pin Descriptions

10.5.4.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low.
MCLK	IN	1	Module Clock.
ParserEn	OUT	1	Parser Enable bit from control register. When this bit becomes active

10.5.4.2 Slicer Instruction Database Interface

Signal	Dir	Width	Description
CMCoSIDWr	OUT	1	CMC to SID Write Strobe
CMCoSIDAdd	OUT	*	CMC to SID Address bus * uses PAR_SID_AWIDTH
SIDData	IN	*	SID Data bus * uses PAR_SID_DWIDTH
CMCoSIDData	OUT	*	CMC to SID Data bus * uses PAR_SID_DWIDTH

10.5.4.3 Pattern Recognition Database Interface

Signal	Dir	Width	Description
CMCoPRDWr	OUT	1	CMC to PRD Write Strobe
CMCoPRDAdd	OUT	*	CMC to PRD Address bus * uses PAR_PRD_AWIDTH
PRDData	IN	*	PRD Data bus * uses PAR_PRD_DWIDTH
CMCoPRDData	OUT	*	CMC to PRD Data bus * uses PAR_PRD_DWIDTH

10.5.4.4 Slicer Interface

Signal	Dir	Width	Description
SiAdd	IN	*	Slicer Address bus * uses PAR_SID_AWIDTH
COCoSiData	OUT	*	CMC to Slicer Data bus * uses PAR_SID_DWIDTH

10.5.4.5 Pattern Recognition Engine Interface

Signal	Dir	Width	Description
PREAdd	IN	*	Pattern Recognition Engine Address bus * uses PAR_PRD_AWIDTH
BaseOffset	OUT	4	Base Offset. This is the first offset the Pattern Recognition Engine will check.
CMCoPREData	OUT	*	CMC to Pattern Rcognition Engine Data bus * uses PAR_PRD_DWIDTH

10.5.4.6 Host Interface Signals

Signal	Dir	Width	Description
ParserSel_N	IN	1	Parser Select - active low. **ParserSel_N** is sampled on the rising edge of **MCLK**. If it is active, it signifies that the external host is attempting to access the parser.
HostWrite	IN	1	Write. Write is sampled on the rising edge of **MCLK**. This signal is only valid when **ParserSel _N** is active. If this signal is active, the host is attempting to write to the parser. Inactive this signal sign signifies a read from the parser.
HostBlast_N	IN	1	Burst Last – active low. **HostBlast_N** is sampled on the rising edge of **MCLK**. **HostBlast_N** tells the parser that the current transfer is the last transfer in this burst.
HostWait_N	IN	1	Wait – active low. **HostWait_N** is sampled on the rising edge of **MCLK**. The host asserts **HostWait_N** when it wishes to slow transfers between itself and the parser.
ParHostReady_N	OUT	1	Parser to Host Ready – active low. **ParHostReady_N** should be sampled on the rising edge of **MCLK**. The parser returns **ParHostReady_N** when the current cycle is completed. For a write operation, **ParHostReady_N** means that the **HostDataIn** bus has been latched. For a read operation **ParHostReady_N** means that the requested data is on the **ParHostDataOut** bus and is valid. **ParHostReady_N** is blocked by **HostWait_N**.

HostAddress	IN	13	Host Address bus. HostAddress is sampled on the rising edge of **MCLK** if **ParserSel_N** is active. This bus defines the first address in this burst to access in the 64 Kilobyte address space of the Parser. See Section x.x.x for the Address Utilization Map.
HostByteEn_N	IN	8	Host Byte Enable bus – Active low. **HostWait_N** is sampled on the rising edge of **MCLK.**
HostDataIn	IN	64	Host Data Input bus. **HostDataIn** is sampled on the rising edge of **MCLK** if **HostWrite** is active and **HostWait_N** is inactive.
ParHostDataOut	OUT	64	ParserHost Data Output bus. **ParHostDataOut** should be sampled on the rising edge of **MCLK.** Data on this bus is valid during a read cycle when **ParHostReady_N** is active.

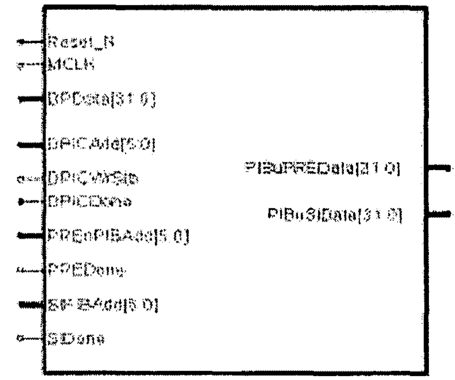
10.5.5 Verilog Module

```
/*
CMC.v
CMC - CPU Interface MUX and Control Register Module


*/
`include "ParserConstants.v"

module CMC(Reset_N, MCLK ,ParserEn, CMCoSIDWr, CMCoSIDAdd, SIDData, CMCoSIDData,
CMCoPRDWr, CMCoPRDAdd, PRDData, CMCoPRDData, SlAdd, CMCoSlData, PREAdd, BaseOffset,
CMCoPREData, ParserSel_N, HostWrite, HostBlast_N, HostWait_N, ParHostReady_N, HostAddress,
HostDataIn, ParHostDataOut);

// General Interface Interface
input Reset_N;
input MCLK;
output ParserEn;
// Sicer Instruction Database Interface
output CMCoSIDWr;
output [`PAR_SID_AWIDTH-1 : 0] CMCoSIDAdd;
input [`PAR_SID_DWIDTH-1 : 0] SIDData;
output [`PAR_SID_AWIDTH-1 : 0] CMCoSIDData;
// Pattern Recognition Database Interface
output CMCoPRDWr;
output [`PAR_PRD_AWIDTH-1 : 0] CMCoPRDAdd;
input [`PAR_PRD_DWIDTH-1 : 0] PRDData;
output [`PAR_PRD_DWIDTH-1 : 0] CMCoPRDData;
// Slicer Interface
input [`PAR_SID_AWIDTH-1 : 0] SlAdd;
output [`PAR_SID_DWIDTH-1 : 0] CMCoSlData;
// Pattern Recognition Engine Interface
input [`PAR_PRD_AWIDTH-1 : 0] PREAdd;
output [`PAR_BASE_OFF_WIDTH-1 : 0] BaseOffset;
output [`PAR_PRD_DWIDTH-1 : 0] CMCoPREData;
//Host Interface
input ParserSel_N;
input HostWrite;
```

```
input HostBlast_N;
input HostWait_N;
output ParHostReady_N;
input ['PAR_HOST_AWIDTH-1 : 0] HostAddress;
input ['PAR_HOST_DWIDTH-1 : 0] HostDataIn;
output ['PAR_HOST_DWIDTH-1 : 0] ParHostDataOut;
```

10.6 Parser Input Buffer Sub-module – PIB

10.6.1 Symbol



10.6.2 Highlights

- Scaleable implementation
- Asynchronous three ported RAM
- Can be build from three separate single port RAM cells
- Wraps either RAM instantiation or can be synthesized latches
- Separate dual read and a single write interfaces

10.6.3 Description

The Parser Input Buffer is a wrapper for the buffer that is used to store the start of the packet. It is three ported with separate dual read and a single write interfaces. The data from the DataPort interface is stored in one of three logical or physical buffers through the write port. The Pattern Recognition Engine uses one of the read ports and the Slicer uses the other. The three interfaces never access the same third of the buffer at the same time. Each of the interfaces looks like a single buffer to the attached modules. The Parser Input Buffer controls which of the three buffers the module is controlling. When the first packet comes in the DataPort Interface Control module writes the data into one of the three buffers. It then increments a modulo three counter to point to the next buffer. The Pattern Recognition Engine will then begin processing the packet. Finally after the Pattern Recognition Engine is finished the Slicer will get access to the buffer. In this way each of the three processes have access to a buffer and each get access to the packet in turn.

10.6.4 Implementation Information

The module can be synthesized or RAM cells can be instantiated into the wrapper. The instantiated RAM can be either a single three ported cell or three separate RAM cells. The Parser Input Buffer can be three separate RAM cells because the control logic will never try to read and write the same third of the buffer at the same time.

10.6.5 File Names

Top: PIB.v(hd)
Uses: ParserConstants.v(hd), Generic3PortRAM.v(hd)

10.6.6 Pin Descriptions

10.6.6.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low.
MCLK	IN	1	Module Clock.
ParserEn	IN	1	Parser Enable bit from control register

10.6.6.2 DataPort Interface

Signal	Dir	Width	Description
DPData	IN	*	DataPort Data bus. * Uses PAR_DP_DWIDTH

10.6.6.3 DataPort Interface Control Interface

Signal	Dir	Width	Description
DPICAdd	IN	*	DataPort Interface Control Address bus. * Uses PAR_PIB_AWIDTH
DPICDone	IN	1	DataPort Interface Control Done. This input is used to tell the Parser Input Buffer that the DataPort Interface Control module has finished writing the buffer. The Parser Input Buffer also uses this signal to increment it's internal pointer so that the next address from the DataPort Interface Control will point to the next packet buffer. **DPICAdd** is ignored for one cycle after **DPICDone** is active.
DPICWriteStb	IN	1	DataPort Interface Control Write Strobe.

10.6.6.4 Pattern Recognition Engine Interface

Signal	Dir	Width	Description
PREnPIBAdd	IN	*	Pattern Recognition Engine Parser Input Buffer Address bus. * Uses PAR_PIB_AWIDTH

10.6.6.4 Pattern Recognition Engine Interface

Signal	Dir	Width	Description
PREDone	IN	1	Pattern Recognition Engine Done. This input is used to tell the Parser Input Buffer that the Pattern Recognition Engine has finished processing the current packet and the buffer can be freed. The Parser Input Buffer also uses this signal to increment it's internal pointer so that the next address from the Pattern Recognition Engine will point to the next packet buffer. **PREnPIBAdd** is ignored for one cycle after **PREDone** is active.
PIBuPREData	OUT	*	Parser Input Buffer to Pattern Recognition Engine Data bus. * Uses PAR_PIB _DWIDTH

10.6.6.5 Slicer Interface

Signal	Dir	Width	Description
SlPIBAdd	IN	*	Slicer Parser Input Buffer Address bus. * Uses PAR_PIB_AWIDTH
SlDone	IN	1	Slicer Done. This input is used to tell the Parser Input Buffer that the Slicer has finished processing the current packet and the buffer can be freed. The Parser Input Buffer also uses this signal to increment it's internal pointer so that the next address from the Slicer will point to the next packet buffer. **SlPIBAdd** is ignored for one cycle after **SlDone** is active.
PIBuSlData	OUT	*	Parser Input Buffer to Slicer Data bus. * Uses PAR_PIB_DWIDTH

10.6.7 Verilog Module

```
/*
PIB.v



*/
`include "ParserConstants.v"

module PIB(Reset_N, MCLK ,ParserEn, DPData, DPICAdd, DPICDone, DPICWrStb, PREnPIBAdd,
PREDone, PIBuPREData, SlPIBAdd, SlDone, PIBuSlData);

input Reset_N;
input MCLK;
input ParserEn;
input DPICDone;
input DPICWrStb;
input PREDone;
input SlDone;
input [`PAR_PIB_DWIDTH-1 : 0] DPData;
input [`PAR_PIB_AWIDTH-1 : 0] DPICAdd;
input [`PAR_PIB_AWIDTH-1 : 0] PREnPIBAdd;
```

```
input ['PAR_PIB_AWIDTH-1 : 0] SlPIBAdd;
output ['PAR_PIB_DWIDTH-1 : 0] PIBuPREData;
output ['PAR_PIB_DWIDTH-1 : 0] PIBuSlData;
```

10.7 Parser Output Buffer Sub-module - POB

10.7.1 Symbol

10.7.2 Highlights

- Scaleable implementation
- Asynchronous dual ported RAM
- Can be build from two separate single port RAM cells
- Wraps either RAM instantiation or can be synthesized latches
- Separate read and write interfaces

10.7.3 Description

The Parser Output Buffer is a wrapper for the buffer that is used to store the output of the Slicer. It is dual ported with separate read and write interfaces. The write interface is controlled by the Slicer. The read interface is controlled by the Analyzer Interface Control logic. The Parser Output Buffer maintains a pointer to the two buffers such that one buffer is controled by the Slicer and one is controlled by the Analyzer Interface Control logic.

10.7.4 Implementation Information

The module can be synthesized or RAM cells can be instantiated into the wrapper. The instantiated RAM can be either a single dual ported cell or two separate RAM cells. The Parser Output Buffer can be two separate RAM cells because the control logic will never try to read and write the same half of the buffer at the same time.

10.7.5 File Names

Top: POB.v(hd)
Uses: ParserConstants.v(hd), Generic2PortRAM.v(hd)

10.7.6 Pin Descriptions

10.7.6.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low.
MCLK	IN	1	Module Clock.
ParserEn	IN	1	Parser Enable bit from control register

10.7.6.2 Slicer Interface

Signal	Dir	Width	Description
SlPOBAdd	IN	*	Slicer Parser Output Buffer Address bus. * Uses PAR_POB_AWIDTH
SlDone	IN	1	Slicer Done. This input is used to tell the Parser Output Buffer that the Slicer has finished processing the current flow and the buffer can be sent to the Analyzer. The Parser Output Buffer also uses this signal to increment it's internal pointer so that the next address from the Slicer will point to the next flow buffer. **SlPOBAdd** is ignored for one cycle after **SlDone** is active.
SlWrStb	IN	1	Slicer Write Strobe.
SlPOBData	IN	*	Slicer to Parser Output Buffer Data bus. * Uses PAR_POB_DWIDTH

10.7.6.3 Analyzer Interface Control Interface

Signal	Dir	Width	Description
AICoPOBAdd	IN	*	Analyzer Interface Control to Parser Output Buffer Address bus. * Uses PAR_POB_AWIDTH
AICDone	IN	1	Analyzer Interface Control Done. This input is used to tell the Parser Output Buffer that the Analyzer Interface Control has finished sending the current flow to the Analyzer. The Parser Output Buffer also uses this signal to increment it's internal pointer so that the next address from the Analyzer Interface Control will point to the next flow buffer. **AICoPOBAdd** is ignored for one cycle after **AICDone** is active.

10.7.6.4 Analyzer Interface

Signal	Dir	Width	Description
ParserData	OUT	*	Parser Data bus. * Uses PAR_ANA_DWIDTH

10.7.7 Verilog Module

```
/*
POB.v


*/
'include "ParserConstants.v"

module POB(Reset_N, MCLK ,ParserEn, SIPOBData, SIPOBAdd, SIDone, SIWrStb,
                                            AICoPOBAdd, AICDone, ParserData);

input Reset_N;
input MCLK;
input ParserEn;
input SIDone;
input SIWrStb;
input AICDone;
input ['PAR_POB_DWIDTH-1 : 0] SIPOBData;
input ['PAR_POB_AWIDTH-1 : 0] SIPOBAdd;
input ['PAR_POB_AWIDTH-1 : 0] AICoPOBAdd;
output ['PAR_POB_DWIDTH-1 : 0] ParserData;
```

10.8 DataPort Interface Control Sub-module - DPIC

10.8.1 Symbol

10.8.2 Description

The DataPort Interface Control module handshakes with the external source of packets. The external device starts sending the packet to the DataPort Interface Control module by asserting **DPPacketDelim**. The transfer of data is coordinated by the **DPDataStb_N/DPReady_N** pair. If the external device decides to about the packet it can assert **DPKillPkt_N**.

10.8.3 Implementation Information

The Analyzer Interface Control module is implemented as a Moore type finite state machine. Each of the outputs of the state machine are registered to assure maximum setup time for the external device.

10.8.4 File Names

Top: DPIC.v(hd)
Uses: ParserConstants.v(hd)

10.8.5 Pin Descriptions

10.8.5.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low.
MCLK	IN	1	Module Clock.
ParserEn	IN	1	Parser Enable bit from control register

10.8.5.2 DataPort Interface

Signal	Dir	Width	Description
DPPacketDelim	IN	1	DataPort Packet Delimiter. This signal should be driven active when the external logic wants to send a packet to the parser. **DPPacketDelim** should remain active during the entire packet transfer. **DPPacketDelim** must go inactive for one clock between packets.
DPDataStb_N	IN	1	DataPort Data Strobe. When active, this signal tells the parser that data on the **DPData** bus is valid. If **DPReady_N** was inactive at the end of the previous cycle, **DPDataStb_N** should not be driven active. If **DPReady_N** goes inactive in the same cycle as **DPDataStb_N**, then the parser will latch the incoming data so that no data is lost.

10.8.5.2 DataPort Interface

Signal	Dir	Width	Description
DPKillPkt_N	IN	1	DataPort Kill Packet. If this signal becomes active while **DPPacketDelim** is active, the parser will attempt to stop processing the current packet and flush it's input Buffer. If however, parsing of the packet is completed, the packet will not be able to be recalled. This should only be a problem in a 'cut through' implementation.
DPReady_N	OUT	1	DataPort Ready – active low. This signal when driven active means that the parser can accept new data. If however the parser's input Buffer is filled, **DPReady_N** will be driven inactive. To prevent overruns, **DPReady_N** will go inactive when the parser can actually accept one more data transfer.

10.8.5.3 Parser Input Buffer Interface

Signal	Dir	Width	Description
DPICAdd	OUT	*	DataPort Interface Control Address bus. * Uses PAR_PIB_AWIDTH
DPICDone	OUT	1	DataPort Interface Control Done. This output is used to tell the Parser Input Buffer that the DataPort Interface Control module has finished writing the buffer.
DPICWriteStb	OUT	1	DataPort Interface Control Write Strobe.

10.8.5.4 Pattern Recognition Engine Interface

Signal	Dir	Width	Description
PREDone	IN	1	Pattern Recognition Engine Done

10.8.6 Verilog Module

```
/*
DPIC.v



*/
`include "ParserConstants.v"

module DPIC(Reset_N, MCLK ,ParserEn, DPPacketDelim, DPDataStb, DPKillPkt_N, DPReady_N,
DPICAdd, DPICDone, DPICWrStb, PREDone);

input Reset_N;
input MCLK;
input ParserEn;
input DPPacketDelim;
```

```
input DPDataStb;
input DPKillPkt_N;
input PREDone;
output DPReady_N;
output DPICDone;
output DPICWrStb;
output [‘PAR_PIB_AWIDTH-1 : 0] DPICAdd;
```

10.9 Analyzer Interface Control Sub-module -AIC

10.9.1 Symbol

10.9.2 Description

The Analyzer Interface Control module handshakes with the Analyzer in order to transfer the flow key for further processing. The Analyzer Interface Control module starts a transfer to the Analyzer by asserting **ParserKeyDelim**. It then transfers the data via the **AnalyzerReady/ParserDataAvail** handshake pair. The Analyzer Interface Control module also sends the address of the data to be sent to the Parser Output Buffer.

10.9.3 Implementation Information

The Analyzer Interface Control module is implemented as a Moore type finite state machine. Each of the outputs of the state machine are registered to assure maximum setup time for the Analyzer interface.

10.9.4 File Names

Top: AIC.v(hd)
Uses: ParserConstants.v(hd)

10.9.5 Pin Descriptions

10.9.5.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low.
MCLK	IN	1	Module Clock.
ParserEn	IN	1	Parser Enable bit from control register

10.9.5.2 Analyzer Interface

Signal	Dir	Width	Description
AnalyzerReady	IN	1	Analyzer Ready. This signal tells the parser that the analyzer can accept data.
ParserKeyDelim	OUT	1	Parser Key Delimiter. The **ParserKeyDelim** signal becomes active when the first quadword of a new key is ready to transfer to the analyzer. It goes inactive when the last quadword of the key is transferred.
ParserDataAvail	OUT	1	Parser Data Available. If this signal is active the data on the **ParserData** bus is valid.

10.9.5.3 Slicer Interface

Signal	Dir	Width	Description
SlFlowKeySize	IN	*	Slicer Flow Key Size bus. This bus is valid when **SlDone** is active. It communicates the size of the flow key so the Analyzer Interface Control can send the right amount of data to the Analyzer. * uses PAR_MAX_FLOW_KEY_SIZE
SlDone	IN	1	Slicer Done. This input is used to tell the Analyzer Interface Control that the Slicer has finished processing the current packet and can be sent to the Analyzer.

10.9.5.4 Parser Output Buffer Interface

Signal	Dir	Width	Description
AICoPOBAdd	OUT	*	Analyzer Interface Control to Parser Output Buffer Address bus. * Uses PAR_POB_AWIDTH
AICDone	OUT	1	Analyzer Interface Control Done. This output is used to tell the Parser Output Buffer that the Analyzer Interface Control has finished sending the current flow to the Analyzer.

10.9.6 Verilog Module

```
/*
AIC.v



*/
'include "ParserConstants.v"

module AIC(Reset_N, MCLK ,ParserEn, AnalyzerReady, ParserKeyDelim, ParserDataAvail,
SlFlowKeySize, SlDone, AICoPOBAdd,AICDone);

input Reset_N;
input MCLK;
input ParserEn;
input AnalyzerReady;
output ParserKeyDelim;
output ParserDataAvail;
input SlDone;
input ['PAR_SL_FKS_WIDTH-1 : 0]SlFlowKeySize;
output ['PAR_PIB_AWIDTH-1 : 0] AICoPOBAdd;
output AICDone;
```

- **Exhibit A3:** Technically Elite MeterFlow Accelerator Analyzer Module Specification (Document MFAAnalyze.pdf)

Technically Elite MeterFlow Accelerator Analyzer Module Specification

Not For External Release!

Revision History		
Version	**Date**	**Description**
0.2	[redacted]	
0.9	[redacted]	Final Prerelease

0 Table of Contents

1 Introduction

This document is designed to be the repository for all information related to the MeterFlow Accelerator Analyzer Module. This specification is designed to provide the engineer with enough information to fully implement the module. There will be revisions during and after the implementation process that will be reflected in this document.

Each part of this specification describes a different aspect of the module. It concentrates on the interfaces between the analyzer module and the other parts of the system. The other parts of the system include the parser module, the host interface module and importantly the software that models, programs and tests the system The key to a successful implementation is the interfaces between modules and between sub-module and sub-module. Each interface is described in detail. Any changes to the interfaces may affect the entire module and even the entire system. Care must be taken that each interface is understood completely before implementation is begun.

2 Technically Elite MeterFlow Accelerator Analyzer Module Highlights

- Flexible Rule-based Traffic Classification
- State-based Tracking of Traffic
- *Multiple* Packets for Layer Processing
- Internal Cache and Memory Controller
- Direct High Bandwidth (64 bit) Memory Interface
- SG/SDRAM Support
- Programmable Rules/State Processor
- Selectable Protocols in Flows
- Future Protocols Support
- Scalable System Design

3 Architectural Overview

The analyzer module consists five major sub-modules with several supporting sub-modules. The major sub-modules are the flow lookup/update engine, the flow insertion and deletion engine, the state processor, the cache, and the unified memory controller. Each of these sub-modules work in parallel to create and update flows.

As a flow key enters the analyzer, the lookup engine attempts to find it in the flow database. If the flow exists, the lookup engine retrieves the flow from the cache. It then makes a decision based on the state information included in the flow entry to either send it to the state processor or not. In either case it updates the flow entry. This updating consists of adding values to counters in the flow database entry. If a flow does not exist, the state processor sends the flow key to the flow insertion and deletion engine which adds the flow to the database.

The state processor updates the flow based on the current state and the flow key information. The state processor processes single and multi packet protocol recognition. It may have to search through a series of possible states to determine the flow's actual state. The result of the state processor's processing is a consolidated flow entry. For example, a PointCast session will open multiple conversations that on a packet by packet basis look like separate flows. Since each conversation is merely a subflow under the PointCast master flow, a single flow that consolidates all of the information for the flow is desired.

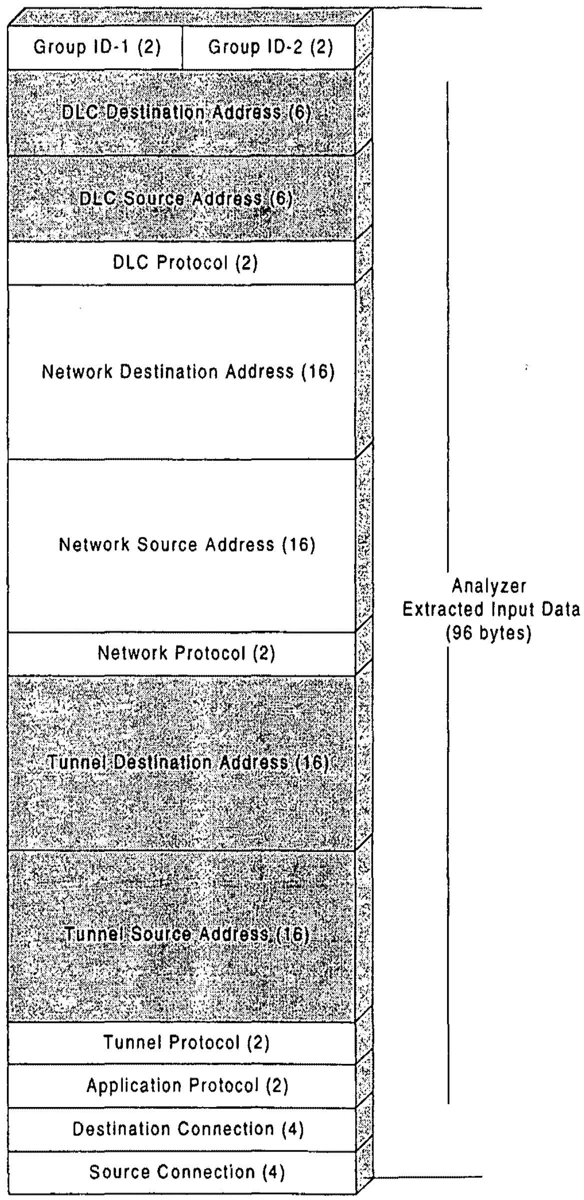
The unified memory controller can be setup to work with various configurations of SDRAM or SGRAM. It also controls the SRAM tag memory for shadowing of flow entries.

The cache is used to optimize memory bandwidth. On a typical network the packets will have a certain amount of congruity. This means that the cache can have a high hit rate with .

3.1 Flow Database

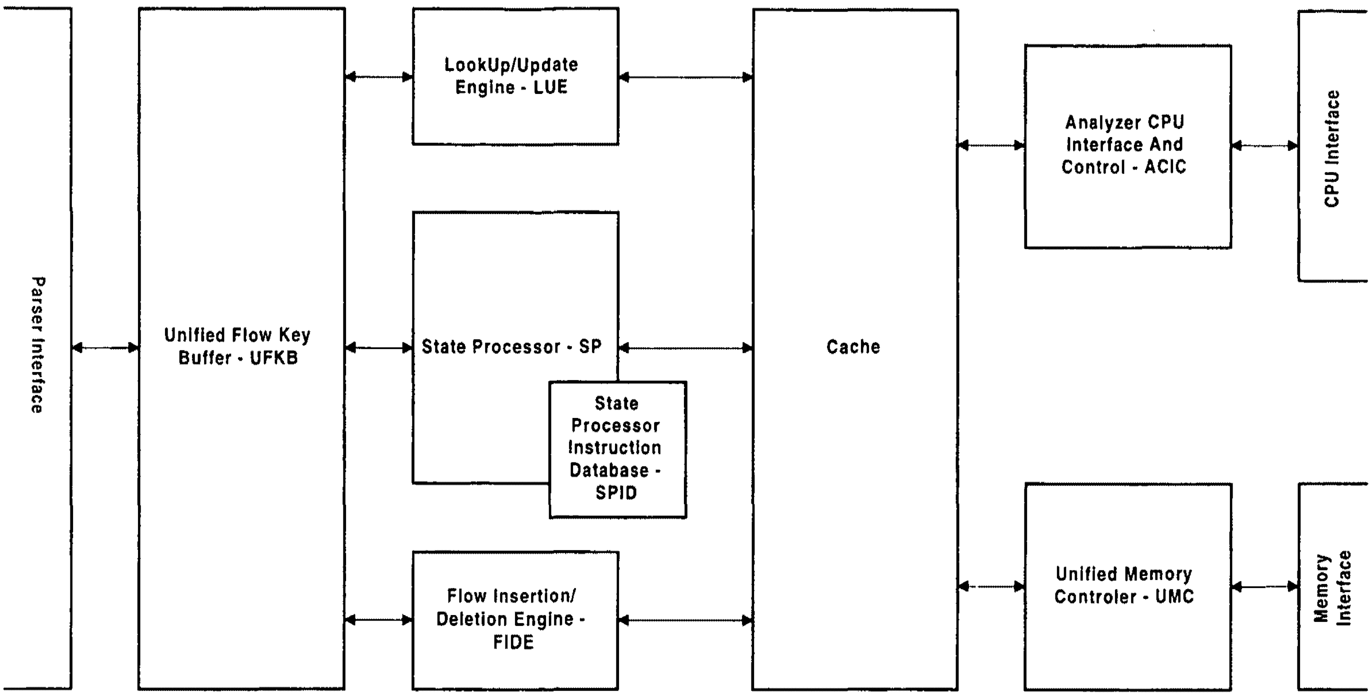
The Flow Database consists of a series of 128 byte entries. Each entry completely describes a flow. The format and information contained in the flow is described in section xxx. The database is organized into buckets. Each bucket contains n flow entries. N is determined by the designer. Buckets are accessed via a hash value created by the Parser based on information in the packet. This hash spreads the flows across the database and is based on a proprietary Technically Elite algorithm. This method allows fast look up of an entry while allowing for shallower buckets. The designer selects the bucket depth based on the amount of memory attached to the analyzer and the number of bits of the hash value used. For example, for 128k flow entries 16 Megabytes are required. Using a 16 bit hash gives two entries per bucket. This has been empirically shown to be more than adequate for the vast majority of cases.

3.1.1 Extracted Input Data from Parser Diagram

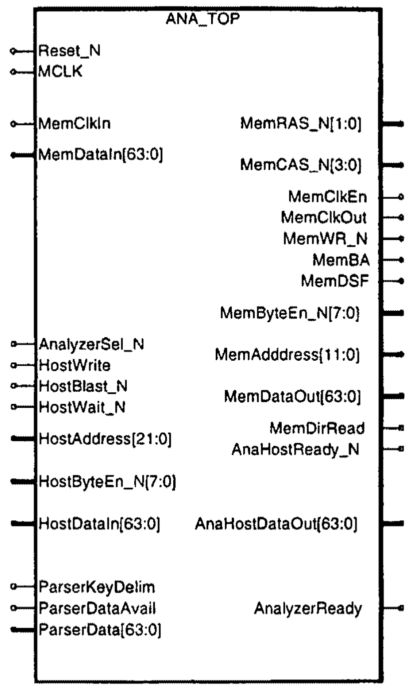


3.1.2 Flow Entry Description

3.2 Architectural Block Diagram



4 Top Level MeterFlow Accelerator Analyzer Module Symbol



5 MeterFlow Accelerator Analyzer Module Top Level Pin Descriptions

5.1.1.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low. When this signal is active the analyzer sets it's registers to their default condition and suspends operation. It will only respond to host access cycles.
MCLK	IN	1	Module Clock. All internal and external transfers except for memory transfers are synchronized by this signal.

5.1.1.2 Memory Interface

Signal	Dir	Width	Description
MemClkIn	IN	1	Memory clock in. This signal is used to generate the memory interface timing.
MemRAS_N	OUT	*	Memory Row Address Strobe bus – active low. * uses AN_MEM_RASWIDTH
MemCAS_N	OUT	*	Memory Column Address Strobe bus– active low. * uses AN_MEM_CASWIDTH
MemClkEn	OUT	1	Memory Clock Enable. Some memories require this signal to be disabled for a certain amount of time after reset.
MemClkOut	OUT	1	Memory Clock Out. This signal is used by synchronous memory for all operations. **MemClkIn** is buffered and sent out on this pin. This helps reduce skew between this clock and the other signals.
MemWR_N	OUT	1	Memory Write – active low.
MemBA	OUT	1	Memory Bank Address. Used by multi-bank memory to select the bank the current operation is to operate on.
MemDSF	OUT	1	Memory Special Function select.
MemByteEn_N	OUT	*	Memory Byte Enable bus– active low. * uses AN_MEM_BEWIDTH
MemAddress	OUT	*	Memory Address bus. * uses AN_MEM_AWIDTH
MemDataIn	IN	*	Memory Data Input bus. * uses AN_MEM_DWIDTH
MemDataOut	OUT	*	Memory Data Output bus. * uses AN_MEM_DWIDTH

5.1.1.2 Memory Interface

Signal	Dir	Width	Description
MemDirRead	OUT	1	Memory Data bus Direction is Read. This signal is used to control the tri-state enable on the bidirectional memory data bus. If **MemDirRead** is active data is coming into the analyzer from the memory. If it is inactive the analyzer is driving data out to the memory.

5.1.1.3 Host Interface Signals

Signal	Dir	Width	Description
AnalyzerSel_N	IN	1	Host interface Analyzer Select - active low. **AnalyzerSel_N** is sampled on the rising edge of **MCLK**. If it is active, it signifies that the external host is attempting to access the analyzer.
HostWrite	IN	1	Write. Write is sampled on the rising edge of **MCLK**. This signal is only valid when **AnalyzerSel_N** is active. If this signal is active, the host is attempting to write to the analyzer. Inactive this signal sign signifies a read from the analyzer. It should also be used to control the direction of the host data bus if it is bidirectional.
HostBlast_N	IN	1	Burst Last – active low. **HostBlast_N** is sampled on the rising edge of **MCLK**. **HostBlast_N** tells the analyzer that the current transfer is the last transfer in this burst.
HostWait_N	IN	1	Wait – active low. **HostWait_N** is sampled on the rising edge of **MCLK**. The host asserts **HostWait_N** when it wishes to slow transfers between itself and the analyzer. This could also be used by additional interface logic to slow transfers so it can multiplex the bus down to a smaller size without additional FIFOs. If wait is active, **HostReady_N** is blocked.
AnaHostReady_N	OUT	1	Analyzer to Host Ready – active low. **AnaHostReady _N** should be sampled on the rising edge of **MCLK**. The analyzer returns **AnaHostReady _N** when the current cycle is completed. For a write operation, **AnaHostReady _N** means that the **HostDataIn** bus has been latched. For a read operation **AnaHostReady _N** means that the requested data is on the **HostDataOut** bus and is valid. **AnaHostReady _N** is blocked by **HostWait_N**.
HostAddress	IN	*	Host Address bus. HostAddress is sampled on the rising edge of **MCLK** if **AnalizerSel_N** is active. This bus defines the first address in this burst to access in the 32 Megabyte address space of the analyzer. See Section x.x.x for the Address Utilization Map. * Uses AN_HOST_AWIDTH
HostByteEn_N	IN	*	Host Byte Enable bus – Active low. **HostWait_N** is sampled on the rising edge of **MCLK**. * Uses AN_HOST_BEWIDTH

5.1.1.3 Host Interface Signals

Signal	Dir	Width	Description
HostDataln	IN	*	Host Data Input bus. **HostDataln** is sampled on the rising edge of **MCLK** if **HostWrite** is active and **HostWait_N** is inactive. * Uses AN_HOST_DWIDTH
AnaHostDataOut	OUT	*	Analyzer Host Data Output bus. **AnaHostDataOut** should be sampled on the rising edge of **MCLK**. Data on this bus is valid during a read cycle when **AnaHostReady _N** is active. * Uses AN_HOST_DWIDTH

5.1.1.4 Parser Interface

Signal	Dir	Width	Description
AnalyzerReady	OUT	1	Analyzer Ready. This signal tells the parser that the analyzer can accept data.
AnalyzerAbort	OUT	1	Analyzer Abort. This signal tells the parser that the analyzer does not need any more of the flow key.
ParserKeyDelim	IN	1	Parser Key Delimiter. The **ParserKeyDelim** signal becomes active when the first quadword of a new key is ready to transfer to the analyzer. It goes inactive when the last quadword of the key is transferred or **AnalyzerAbort** Is active.
ParserDataAvail	IN	1	Parser Data Available. If this signal is active the data on the ParserData bus is valid.
ParserData	IN	*	Parser Data bus.

5.1.1.5 Known Flow Interface

Signal	Dir	Width	Description
PacketRef	OUT	*	Packet Reference number bus. This bus outputs the packet reference number copied from the UFKB. * Uses AN_FR_WIDTH
Protocol	OUT	*	Protocol bus. This bus outputs the highest level protocol the State Processor has determined the packet contains. * Uses AN_APP_WIDTH
KnownFlowStb	OUT	1	Known Flow Strobe. When this signal is active, the data on the **PacketRef** and the **Protocol** busses are valid.

6 MeterFlow Accelerator Analyzer Module Top Level VHDL Entity

7 MeterFlow Accelerator Analyzer Module Top Level Verilog Module

8 MeterFlow Accelerator Analyzer Module Top Level Schematic

Insert Schematic Here

9 Analyzer Module Constants Files

The analyzer module constants files contain a list of constants used to allow rapid configuration of the module. For example the size of the Analyzer's input buffer data bus:

Verilog

'define AN_UFKB_DWIDTH 64 // Unified Flow Key Buffer Data Bus Width

VHDL

constant AN_UFKB_DWIDTH : integer := 64; -- Unified Flow Key Buffer Data Bus Width

9.1 Analyzer module Verilog Constants File – ParserConstants.v

Insert AnalyzerConstants.v here

9.2 Analyzer module VHDL Constants File – ParserConstants.vhd

Insert AnalyzerConstants.vhd here

10 Sub-module Descriptions

10.1 Unified Flow Key Buffer - UFKB

10.1.1 Symbol

10.1.2 Highlights

- Scaleable implementation
- Can be build from four separate dual port RAM cells
- Wraps either RAM instantiation or can be synthesized latches
- Separate read and write interfaces

10.1.3 Description

The Unified Flow Key Buffer is a wrapper for the buffers that are used to store the flow keys from the Parser and modified flow keys from the Lookup and Update Engine and the State Processor. It is four ported with separate read and write interfaces. The four connections are to the Parser/Parser Interface Control, the Lookup and Update Engine, the State Processor and the Flow Insertion and Deletion Engine. In the Unified Flow Key Buffer logic hides from the interface which of the buffers is being accessed.

When the first word of the flow key arrives from the Parser, the Lookup and Update Engine is notified. The Lookup and Update Engine places the first address it wants on the **LUEnUFKBAdd** bus and asserts **LUEnUFKBRdReq**. If the address requested is in the buffer the Unified Flow Key Buffer asserts **UFKBuLUERdy**. If not it waits for either the data to arrive or the transfer is terminated. Once the Lookup and Update Engine finishes processing the flow key it asserts **LUEDone**. At the same time it will assert **LUEHoldBuf**. **LUEHoldBuf** tells the system that the buffer is to be sent to the State Processor.

The State Processor and Flow Insertion and Deletion Engine have similar interfaces except that the data is assumed to be already in the buffer so no ready is returned. Also Flow Insertion and Deletion Engine has no need to hold the buffer for another process so that once **FIDEDone** is asserted the buffer is freed.

10.1.4 Implementation Information

The module can be synthesized or RAM cells can be instantiated into the wrapper. The instantiated RAM should be four separate dual ported RAM cells.

The RAM must complete a write or read in a single cycle with simultaneous read and write to SEPARATE locations.

10.1.5 File Names

Top: UFKB.v(hd)
Uses: AnalyzerConstants.v(hd), Generic4PortRAM.v(hd)

10.1.6 Pin Descriptions

10.1.6.1 General Interface Signals

Signal	Dir	Width	Description

10.1.6.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low.
MCLK	IN	1	Module Clock.
AnalyzerEn	IN	1	Analyzer Enable bit from the control register

10.1.6.2 Parser Interface

Signal	Dir	Width	Description
AnalyzerReady	OUT	1	Analyzer Ready. This signal tells the parser that the analyzer can accept data.
AnalyzerAbort	OUT	1	Analyzer Abort. This signal tells the parser that the analyzer does not need any more of the flow key. It is generated if the Lookup and Update Engine asserts **LUEDone** and not **LUEHoldBuf** before **ParserKeyDelim** goes inactive.
ParserKeyDelim	IN	1	Parser Key Delimiter. The **ParserKeyDelim** signal becomes active when the first word of a new key is ready to transfer to the analyzer. It goes inactive when the last word of the key is transferred.
ParserDataAvail	IN	1	Parser Data Available. If this signal is active, the data on the **ParserData** bus is valid.

10.1.6.3 Lookup and Update Engine Interface

Signal	Dir	Width	Description
UFKBuLUEData	OUT	*	Unified Flow Key Buffer to Lookup and Update Engine read Data bus. * Uses AN_UFKB_DWIDTH
LUEnUFKBData	IN	*	Lookup and Update Engine to Unified Flow Key Buffer write Data bus. * Uses AN_ UFKB _DWIDTH
LUEnUFKBAdd	IN	*	Lookup and Update Engine to Unified Flow Key Buffer Address bus. * Uses AN_ UFKB _AWIDTH
FlowKeySt	OUT	1	Flow Key Start. This signal tells the Lookup and Update Engine that the Unified Flow Key Buffer module has placed the first word of a flow key buffer.
UFKBuLUERdy	OUT	1	Unified Flow Key Buffer to Lookup and Update Engine Ready.
UFKBuLUEErr	OUT	1	Unified Flow Key Buffer to Lookup and Update Engine Error. Asserted if a read request times out.
LUEnUFKBRdReq	IN	1	Lookup and Update Engine to Unified Flow Key Buffer Read Request.
LUEnUFKBWrStb	IN	1	Lookup and Update Engine to Unified Flow Key Buffer Write Strobe.

10.1.6.3 Lookup and Update Engine Interface

Signal	Dir	Width	Description
LUEDone	IN	1	Lookup and Update Engine Done. This input is used to tell the Unified Flow Key Buffer that the Lookup and Update Engine has finished with the current flow. The Unified Flow Key Buffer also uses this signal to increment it's internal pointer so that the next address from Lookup and Update Engine will point to the next flow buffer.
LUEHoldBuf	IN	1	Lookup and Update Engine Hold Buffer. This input is used to tell the Unified Flow Key Buffer that the Lookup and Update Engine is transferring processing of this buffer to the State Processor.

10.1.6.4 State Processor Interface

Signal	Dir	Width	Description
UFKBuSPData	OUT	*	Unified Flow Key Buffer to State Processor read Data bus. * Uses AN_UFKB_AWIDTH
SPrUFKBData	IN	*	State Processor to Unified Flow Key Buffer write Data bus. * Uses AN_ UFKB _AWIDTH
SPrUFKBAdd	IN	*	State Processor to Unified Flow Key Buffer Address bus. * Uses AN_ UFKB _AWIDTH
SPFlowKeyAv	OUT	1	State Processor Flow Key Available. This signal tells the State Processor that the Unified Flow Key Buffer module a flow key for it to process.
SPrUFKBWrStb	IN	1	State Processor to Unified Flow Key Buffer Write Strobe.
SPDone	IN	1	State Processor Done. This input is used to tell the Unified Flow Key Buffer that the State Processor has finished with the current flow. The Unified Flow Key Buffer also uses this signal to increment it's internal pointer so that the next address from State Processor will point to the next flow buffer.
SPHoldBuf	IN	1	State Processor Hold Buffer. This input is used to tell the Unified Flow Key Buffer that the State Processor is transferring processing of this buffer to the Flow Insertion and Deletion Engine.

10.1.6.5 Flow Insertion and Deletion Engine Interface

Signal	Dir	Width	Description
UFKBuFIDEData	OUT	*	Unified Flow Key Buffer to Flow Insertion and Deletion Engine read Data bus. * Uses AN_UFKB_AWIDTH
FIDEnUFKBAdd	IN	*	Flow Insertion and Deletion Engine to Unified Flow Key Buffer Address bus. * Uses AN_ UFKB _AWIDTH
FIDEFlowKeyAv	OUT	1	Flow Insertion and Deletion Engine Flow Key Available. This signal tells the Flow Insertion and Deletion Engine that the Unified Flow Key Buffer module a flow key for it to process.

10.1.6.5 Flow Insertion and Deletion Engine Interface

Signal	Dir	Width	Description
FIDEDone	IN	1	Flow Insertion and Deletion Engine Done. This input is used to tell the Unified Flow Key Buffer that the Flow Insertion and Deletion Engine has finished with the current flow. The Unified Flow Key Buffer also uses this signal to increment it's internal pointer so that the next address from Flow Insertion and Deletion Engine will point to the next flow buffer.

10.1.7 Verilog Module

```
module UFKB(Reset_N, MCLK ,AnalyzerEn ,AnalyzerReady ,AnalyzerAbort
,ParserKeyDelim ,ParserDataAvail ,UFKBuLUEData ,LUEnUFKBData
,LUEnUFKBAdd ,FlowKeySt ,UFKBuLUERdy ,UFKBuLUEErr ,LUEnUFKBRdReq
,LUEnUFKBWrStb ,LUEDone ,LUEHoldBuf ,UFKBuSPData ,SPrUFKBData
,SPrUFKBAdd ,SPFlowKeyAv ,SPrUFKBWrStb ,SPDone ,SPHoldBuf ,UFKBuFIDEData
,FIDEnUFKBAdd ,FIDEFlowKeyAv ,FIDEDone);

// General Interface Interface
input Reset_N;
input MCLK;
input AnalyzerEn;
// Parser Interface
output AnalyzerReady;
output AnalyzerAbort;
input ParserKeyDelim;
input ParserDataAvail;
// Lookup and Update Engine Interface
output [`AN_UFKB_DWIDTH-1 : 0] UFKBuLUEData;
input [`AN_UFKB_DWIDTH-1 : 0] LUEnUFKBData;
input [`AN_UFKB_AWIDTH-1 : 0] LUEnUFKBAdd;
output FlowKeySt;
output UFKBuLUERdy;
output UFKBuLUEErr;
input LUEnUFKBRdReq;
input LUEnUFKBWrStb;
input LUEDone;
input LUEHoldBuf;
// State Processor Interface
output [`AN_UFKB_DWIDTH-1 : 0] UFKBuSPData;
input [`AN_UFKB_DWIDTH-1 : 0] SPrUFKBData;
input [`AN_UFKB_AWIDTH-1 : 0] SPrUFKBAdd;
output SPFlowKeyAv;
input SPrUFKBWrStb;
input SPDone;
input SPHoldBuf;
// Flow Insertion and Deletion Engine
output [`AN_UFKB_DWIDTH-1 : 0] UFKBuFIDEData;
input [`AN_UFKB_AWIDTH-1 : 0] FIDEnUFKBAdd;
```

```
output FIDEFlowKeyAv;
input FIDEDone;
```

10.1.8 VHDL Component

10.2 Lookup and Update Engine - LUE

10.2.1 Symbol

10.2.2 Highlights

- Looks up flow entries
- Compares flow key from parser to flow entries
- Updates packet count and byte count tables
- 64 bit byte count adder with early out
- Checks flow state to see if processing by the state processor is required

10.2.3 Description

The Lookup and Update Engine begins processing as soon as a flow key arrives from the parser. The first transfer from the parser contains a hash value that is used as an offset into the flow entry database. The LUE checks the entry to see if it matches the flow key by comparing the unique identification for that flow. If there is a match, the LUE updates the counters for the flow entry. The LUE also check the entry's flow state to see if the flow key needs to be sent to the state processor.

The Lookup and Update Engine also outputs on a special data bus, two 16 bit values. One value is a word from the flow key that can be a packet identifier or any thing else the design wants. The other is the protocol identifier for the flow. This can be programmed to output this data on every packet or only for packets that the corresponding flow is in the IDENTIFIED state.

10.2.4 Implementation Information

10.2.5 File Names

Top: LUE.v(hd)
Uses: AnalyzerConstants.v(hd)

10.2.6 Pin Descriptions

10.2.6.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low.
MCLK	IN	1	Module Clock.
AnalyzerEn	IN	1	Analyzer Enable bit from the control register

10.2.6.2 Unified Flow Key Buffer Interface

Signal	Dir	Width	Description
UFKBuLUEData	IN	*	Unified Flow Key Buffer to Lookup and Update Engine read Data bus. * Uses AN_UFKB_DWIDTH

10.2.6.2 Unified Flow Key Buffer Interface

Signal	Dir	Width	Description
LUEnUFKBData	OUT	*	Lookup and Update Engine to Unified Flow Key Buffer write Data bus. * Uses AN_ UFKB _DWIDTH
LUEnUFKBAdd	OUT	*	Lookup and Update Engine to Unified Flow Key Buffer Address bus. * Uses AN_ UFKB _AWIDTH
FlowKeySt	IN	1	Flow Key Start. This signal tells the Lookup and Update Engine that the Unified Flow Key Buffer module has placed the first word of a flow key buffer.
UFKBuLUERdy	IN	1	Unified Flow Key Buffer to Lookup and Update Engine Ready.
UFKBuLUEErr	IN	1	Unified Flow Key Buffer to Lookup and Update Engine Error. Asserted if a read request times out.
LUEnUFKBRdReq	OUT	1	Lookup and Update Engine to Unified Flow Key Buffer Read Request.
LUEnUFKBWrStb	OUT	1	Lookup and Update Engine to Unified Flow Key Buffer Write Strobe.
LUEDone	OUT	1	Lookup and Update Engine Done. This input is used to tell the Unified Flow Key Buffer that the Lookup and Update Engine has finished with the current flow. The Unified Flow Key Buffer also uses this signal to increment it's internal pointer so that the next address from Lookup and Update Engine will point to the next flow buffer.
LUEHoldBuf	OUT	1	Lookup and Update Engine Hold Buffer. This input is used to tell the Unified Flow Key Buffer that the Lookup and Update Engine is transferring processing of this buffer to the State Processor.

10.2.6.3 Cache Interface

Signal	Dir	Width	Description
CaLUEReady	IN	1	Cache to Lookup Engine Ready. This signal tells the Lookup Engine that during a read, the data on the **CaLUEData** bus is valid and during a write that the Cache has latched the data on the **LUEnCaData** bus.
CaLUEData	IN	*	Cache to Lookup Engine Data bus. * Uses AN_CA_DWIDTH
LUEnCaData	OUT	*	Lookup Engine to Cache Data bus. * Uses AN_CA_DWIDTH
LUEAdd	OUT	*	Lookup Engine to Cache Address bus. * Uses AN_CA_AWIDTH
LUEMemReq	OUT	1	Lookup Engine Memory Request. If this signal is active, the address on the **LUEAdd** bus is valid.
LUEMemWr	OUT	1	Lookup Engine Memory Write. If this signal is active, the current transaction is a write..

10.2.6.4 Known Flow Interface

Signal	Dir	Width	Description
PacketRef	OUT	*	Packet Reference number bus. This bus outputs the packet reference number copied from the UFKB. * Uses AN_FR_WIDTH
Protocol	OUT	*	Protocol bus. This bus outputs the highest level protocol the State Processor has determined the packet contains. * Uses AN_PRO_WIDTH
KnownFlowStb	OUT	1	Known Flow Strobe. When this signal is active, the data on the **PacketRef** and the **Protocol** busses are valid.

10.2.7 Verilog Module

```
module LUE(Reset_N, MCLK ,AnalyzerEn ,UFKBuLUEData ,LUEnUFKBData
,LUEnUFKBAdd ,FlowKeySt ,UFKBuLUERdy ,UFKBuLUEErr ,LUEnUFKBRdReq
,LUEnUFKBWrStb ,LUEDone ,LUEHoldBuf ,CaLUEData ,LUEnCaData ,LUEAdd
,LUEMemReq ,LUEMemWr);

// General Interface Interface
input Reset_N;
input MCLK;
input AnalyzerEn;
// Unified Flow Key Buffer Interface
input [`AN_UFKB_DWIDTH-1 : 0] UFKBuLUEData;
output [`AN_UFKB_DWIDTH-1 : 0] LUEnUFKBData;
output [`AN_UFKB_AWIDTH-1 : 0] LUEnUFKBAdd;
input FlowKeySt;
input UFKBuLUERdy;
input UFKBuLUEErr;
output LUEnUFKBRdReq;
output LUEnUFKBWrStb;
output LUEDone;
output LUEHoldBuf;
// Cache Interface
input CaLUEReady;
input [`AN_CA_DWIDTH-1 : 0] CaLUEData;
output [`AN_CA_DWIDTH-1 : 0] LUEnCaData;
output [`AN_CA_AWIDTH-1 : 0] LUEAdd;
output LUEMemReq;
output LUEMemWr;
// Known Flow Interface
output [`AN_FR_WIDTH-1 : 0] PacketRef;
output [`AN_PRO_WIDTH-1 : 0] Protocol;
output KnownFlowStb;
```

10.2.8 VHDL Component

10.3 Analyzer CPU Interface and Control - ACIC

10.3.1 Symbol

10.3.2 Description

The Analyzer CPU Interface Control module controls the communication between the external CPU and the Analyzer. The ACIC contains MUX's for the CPU read back path. It also contains the control register for the Analyzer.

10.3.3 File Names

Top: ACIC.v(hd)
Uses: AnalyzerConstants.v(hd)

10.3.4 Pin Descriptions

10.3.4.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low.
MCLK	IN	1	Module Clock.
AnalyzerEn	OUT	1	Analyzer Enable bit from the control register

10.3.4.2 Host Interface Signals

Signal	Dir	Width	Description
AnalyzerSel_N	IN	1	Host interface Analyzer Select - active low. **AnalyzerSel_N** is sampled on the rising edge of **MCLK**. If it is active, it signifies that the external host is attempting to access the analyzer.
HostWrite	IN	1	Write. Write is sampled on the rising edge of **MCLK**. This signal is only valid when **AnalyzerSel_N** is active. If this signal is active, the host is attempting to write to the analyzer. Inactive this signal sign signifies a read from the analyzer. It should also be used to control the direction of the host data bus if it is bidirectional.
HostBlast_N	IN	1	Burst Last – active low. **HostBlast_N** is sampled on the rising edge of **MCLK**. **HostBlast_N** tells the analyzer that the current transfer is the last transfer in this burst.
HostWait_N	IN	1	Wait – active low. **HostWait_N** is sampled on the rising edge of **MCLK**. The host asserts **HostWait_N** when it wishes to slow transfers between itself and the analyzer. This could also be used by additional interface logic to slow transfers so it can multiplex the bus down to a smaller size without additional FIFOs. If wait is active, **HostReady_N** is blocked.

10.3.4.2 Host Interface Signals

Signal	Dir	Width	Description
AnaHostReady_N	OUT	1	Analyzer to Host Ready – active low. **AnaHostReady _N** should be sampled on the rising edge of **MCLK**. The analyzer returns **AnaHostReady _N** when the current cycle is completed. For a write operation, **AnaHostReady _N** means that the **HostDataIn** bus has been latched. For a read operation **AnaHostReady _N** means that the requested data is on the **HostDataOut** bus and is valid. **AnaHostReady _N** is blocked by **HostWait_N**.
HostAddress	IN	*	Host Address bus. HostAddress is sampled on the rising edge of **MCLK** if **AnalizerSel_N** is active. This bus defines the first address in this burst to access in the 32 Megabyte address space of the analyzer. See Section x.x.x for the Address Utilization Map. * Uses AN_HOST_AWIDTH
HostByteEn_N	IN	*	Host Byte Enable bus – Active low. **HostWait_N** is sampled on the rising edge of **MCLK**. * Uses AN_HOST_BEWIDTH
HostDataIn	IN	*	Host Data Input bus. **HostDataIn** is sampled on the rising edge of **MCLK** if **HostWrite** is active and **HostWait_N** is inactive. * Uses AN_HOST_DWIDTH
AnaHostDataOut	OUT	*	Analyzer Host Data Output bus. **AnaHostDataOut** should be sampled on the rising edge of **MCLK**. Data on this bus is valid during a read cycle when **AnaHostReady _N** is active. * Uses AN_HOST_DWIDTH

10.3.4.3 Cache Interface

Signal	Dir	Width	Description
CaACICReady	IN	1	Cache to Analyzer CPU Interface Control Ready. This signal tells the Analyzer CPU Interface Control that during a read, the data on the **CaACICData** bus is valid and during a write that the Cache has latched the data on the **ACICnCaData** bus.
CaACICData	IN	*	Cache to Analyzer CPU Interface Control Data bus. * Uses AN_CA_DWIDTH
ACICoCaData	OUT	*	Analyzer CPU Interface Control to Cache Data bus. * Uses AN_CA_DWIDTH
ACICAdd	OUT	*	Analyzer CPU Interface Control to Cache Address bus. * Uses AN_CA_AWIDTH
ACICMemReq	OUT	1	Analyzer CPU Interface Control Memory Request. If this signal is active, the address on the **ACICAdd** bus is valid.
ACICMemWr	OUT	1	Analyzer CPU Interface Control Memory Write. If this signal is active, the current transaction is a write..

10.3.4.4 State Processor Instruction Database Interface

Signal	Dir	Width	Description
ACICoSPIDWr	OUT	1	Analyzer CPU Interface Control to State Processor Instruction Database Write Strobe
ACICoSPIDAdd	OUT	*	Analyzer CPU Interface Control to State Processor Instruction Database Address bus * uses AN_SPID_AWIDTH
SPIDData	IN	*	State Processor Instruction Database Data bus * uses AN_SPID _DWIDTH
ACICoSPIDData	OUT	*	Analyzer CPU Interface Control to State Processor Instruction Database Data bus * uses AN_SPID _DWIDTH

10.3.5 Verilog Module

```
module ACIC(Reset_N, MCLK ,AnalyzerEn ,AnalyzerSel_N ,HostWrite
,HostBlast_N ,HostWait_N ,AnaHostReady_N ,HostAddress ,HostByteEn_N
,HostDataIn ,AnaHostDataOut ,CaACICReady ,CaACICData ,ACICoCaData
,ACICAdd , ACICMemReq ,ACICMemWr ,ACICoSPIDWr ,ACICoSPIDAdd ,SPIDData
,ACICoSPIDData);

// General Interface Interface
input Reset_N;
input MCLK;
output AnalyzerEn;
// Host Interface
input AnalyzerSel_N;
input HostWrite;
input HostBlast_N;
input HostWait_N;
output AnaHostReady_N;
input [`AN_HOST_AWIDTH-1 : 0] HostAddress;
input [`AN_HOST_BEWIDTH-1 : 0] HostByteEn_N;
input [`AN_HOST_DWIDTH-1 : 0] HostDataIn;
output [`AN_HOST_DWIDTH-1 : 0] AnaHostDataOut;
// Cache  Interface
input CaACICReady;
input [`AN_CA_DWIDTH-1 : 0] CaACICData;
output [`AN_CA_DWIDTH-1 : 0] ACICoCaData;
output [`AN_CA_AWIDTH-1 : 0] ACICAdd;
output ACICMemReq;
output ACICMemWr;
// State Processor Instruction Database Interface
output ACICoSPIDWr;
output [`AN_SPID_AWIDTH-1 : 0] ACICoSPIDAdd;
input [`AN_SPID_DWIDTH-1 : 0] SPIDData;
output [`AN_SPID_DWIDTH-1 : 0] ACICoSPIDData;
```

10.3.6 VHDL Component

10.4 Flow Insertion and Deletion Engine - FIDE

10.4.1 Symbol

10.4.2 Highlights

- Maintains flow entry database
- Deletes and inserts flows based on a LRU algorithm
- Builds flows from flow key and State Processor instructions

10.4.3 Description

The Flow Insertion and Deletion Engine maintains the flow entry database. Flows are grouped into buckets by hash value. When a new flow needs to be inserted first the FIDE sees which of the entries in the corresponding bucket is the oldest. It then builds the flow entry from the flow key and State Processor instructions. Finally it places the entry in the database.

10.4.4 Implementation Information

10.4.5 File Names

Top: FIDE.v(hd)
Uses: AnalyzerConstants.v(hd)

10.4.6 Pin Descriptions

10.4.6.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low.
MCLK	IN	1	Module Clock.
AnalyzerEn	IN	1	Analyzer Enable bit from the control register

10.4.6.2 Unified Flow Key Buffer Interface

Signal	Dir	Width	Description
UFKBuFIDEData	IN	*	Unified Flow Key Buffer to Flow Insertion and Deletion Engine read Data bus. * Uses AN_UFKB_AWIDTH
FIDEnUFKBAdd	OUT	*	Flow Insertion and Deletion Engine to Unified Flow Key Buffer Address bus. * Uses AN_ UFKB _AWIDTH
FIDEFlowKeyAv	IN	1	Flow Insertion and Deletion Engine Flow Key Available. This signal tells the Flow Insertion and Deletion Engine that the Unified Flow Key Buffer module a flow key for it to process.

10.4.6.2 Unified Flow Key Buffer Interface

Signal	Dir	Width	Description
FIDEDone	OUT	1	Flow Insertion and Deletion Engine Done. This input is used to tell the Unified Flow Key Buffer that the Flow Insertion and Deletion Engine has finished with the current flow. The Unified Flow Key Buffer also uses this signal to increment it's internal pointer so that the next address from Flow Insertion and Deletion Engine will point to the next flow buffer.

10.4.6.3 Cache Interface

Signal	Dir	Width	Description
CaFIDEReady	IN	1	Cache to Flow Insertion and Deletion Engine Ready. This signal tells the Flow Insertion and Deletion Engine that during a read, the data on the **CaFIDEData** bus is valid and during a write that the Cache has latched the data on the **FIDEnCaData** bus.
CaFIDEData	IN	*	Cache to Flow Insertion and Deletion Engine Data bus. * Uses AN_CA_DWIDTH
FIDEnCaData	OUT	*	Flow Insertion and Deletion Engine to Cache Data bus. * Uses AN_CA_DWIDTH
FIDEAdd	OUT	*	Flow Insertion and Deletion Engine to Cache Address bus. * Uses AN_CA_AWIDTH
FIDEMemReq	OUT	1	Flow Insertion and Deletion Engine Memory Request. If this signal is active, the address on the **FIDEAdd** bus is valid.
FIDEMemWr	OUT	1	Flow Insertion and Deletion Engine Memory Write. If this signal is active, the current transaction is a write..

10.4.7 Verilog Module

```
module FIDE(Reset_N, MCLK ,AnalyzerEn ,UFKBuFIDEData ,FIDEnUFKBAdd
,FIDEFlowKeyAv ,FIDEDone ,CaFIDEData ,FIDEnCaData ,FIDEAdd ,FIDEMemReq
,FIDEMemWr);

// General Interface Interface
input Reset_N;
input MCLK;
input AnalyzerEn;
// Unified Flow Key Buffer Interface
input [`AN_UFKB_DWIDTH-1 : 0] UFKBuFIDEData;
output [`AN_UFKB_AWIDTH-1 : 0] FIDEnUFKBAdd;
input FIDEFlowKeyAv;
output FIDEDone;
// Cache Interface
input CaFIDEReady;
input [`AN_CA_DWIDTH-1 : 0] CaFIDEData;
output [`AN_CA_DWIDTH-1 : 0] FIDEnCaData;
```

```
output [`AN_CA_AWIDTH-1 : 0] FIDEAdd;
output FIDEMemReq;
output FIDEMemWr;
```

10.4.8 VHDL Component

10.5 State Processor Instruction Database - SPID

10.5.1 Symbol

10.5.2 Highlights

- Scaleable implementation
- Wraps either RAM or ROM instantiation or can be synthesized latches

10.5.3 Description

The State Processor Instruction Database module is a wrapper for the storage medium used to hold the State Processor Instruction database. Only the CPU can write this memory. The CPU interface is active if **AnalyzerEn** is active.

10.5.4 Implementation Information

The module can be synthesized or a RAM or ROM cell can be instantiated into the wrapper.

10.5.5 File Names

Top: SPID.v(hd)
Uses: AnalyzerConstants.v(hd)

10.5.6 Pin Descriptions

10.5.6.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low.
MCLK	IN	1	Module Clock.
AnalyzerEn	IN	1	Analyzer Enable bit from the control register

10.5.6.2 Analyzer CPU Interface Control Interface

Signal	Dir	Width	Description
ACICoSPIDWr	IN	1	Analyzer CPU Interface Control to State Processor Instruction Database Write Strobe
ACICoSPIDAdd	IN	*	Analyzer CPU Interface Control to State Processor Instruction Database Address bus * uses AN_SPID_AWIDTH
SPIDData	OUT	*	State Processor Instruction Database Data bus * uses AN_SPID _DWIDTH
ACICoSPIDData	IN	*	Analyzer CPU Interface Control to State Processor Instruction Database Data bus * uses AN_SPID _DWIDTH

10.5.6.3 State Processor Interface

Signal	Dir	Width	Description
SPrSPIDAdd	IN	*	State Processor to State Processor Instruction Database Address bus * uses AN_SPID_AWIDTH

10.5.7 Verilog Module

```
module SPID(Reset_N, MCLK ,AnalyzerEn ,ACICoSPIDWr ,ACICoSPIDAdd
,SPIDData ,ACICoSPIDData ,SPrSPIDAdd);

// General Interface Interface
input Reset_N;
input MCLK;
input AnalyzerEn;
// Analyzer CPU Interface Control Interface
input ACICoSPIDWr;
input [`AN_SPID_AWIDTH-1 : 0] ACICoSPIDAdd;
output [`AN_SPID_DWIDTH-1 : 0] SPIDData;
input [`AN_SPID_DWIDTH-1 : 0] ACICoSPIDData;
// State Processor Interface
input [`AN_SPID_AWIDTH-1 : 0] SPrSPIDAdd;
```

10.5.8 VHDL Component

10.6 Unified Memory Controller - UMC

10.6.1 Symbol

10.6.2 Highlights

- Supports Both SDRAM and SGRAM
- Maintains RAM refresh

10.6.3 Description

The Unified Memory Controller module controls the caches' access to the flow database contained in external RAM. Synchronous DRAM is controlled through a series of instructions feed to the RAM through the control pins. Synchronous DRAM requires at startup a specific series of commands for initialization. The Unified Memory Controller handles both processes thorough a state machine. Since the nature of the flow database requires random access, there is little use in attempting to keep multiple banks open. Auto-refresh is continuous when memory is not being accessed by the cache.

10.6.4 Implementation Information

The Unified Memory Controller module is implemented as a Moore type finite state machine. Each of the outputs of the state machine are registered to assure maximum setup time for the external device.

10.6.5 File Names

Top: UMC.v(hd)
Uses: AnalyzerConstants.v(hd)

10.6.6 Pin Descriptions

10.6.6.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low.
MCLK	IN	1	Module Clock.
AnalyzerEn	IN	1	Analyzer Enable bit from the control register

10.6.6.2 Memory Interface

Signal	Dir	Width	Description
MemClkIn	IN	1	Memory clock in. This signal is used to generate the memory interface timing.
MemRAS_N	OUT	*	Memory Row Address Strobe bus – active low. * uses AN_MEM_RASWIDTH
MemCAS_N	OUT	*	Memory Column Address Strobe bus– active low. * uses AN_MEM_CASWIDTH
MemClkEn	OUT	1	Memory Clock Enable. Some memories require this signal to be disabled for a certain amount of time after reset.

10.6.6.2 Memory Interface

Signal	Dir	Width	Description
MemClkOut	OUT	1	Memory Clock Out. This signal is used by synchronous memory for all operations. **MemClkIn** is buffered and sent out on this pin. This helps reduce skew between this clock and the other signals.
MemWR_N	OUT	1	Memory Write – active low.
MemBA	OUT	1	Memory Bank Address. Used by multi-bank memory to select the bank the current operation is to operate on.
MemDSF	OUT	1	Memory Special Function select.
MemByteEn_N	OUT	*	Memory Byte Enable bus– active low. * uses AN_MEM_BEWIDTH
MemAddress	OUT	*	Memory Address bus. * uses AN_MEM_AWIDTH
MemDataIn	IN	*	Memory Data Input bus. * uses AN_MEM_DWIDTH
MemDataOut	OUT	*	Memory Data Output bus. * uses AN_MEM_DWIDTH
MemDirRead	OUT	1	Memory Data bus Direction is Read. This signal is used to control the tri-state enable on the bidirectional memory data bus. If **MemDirRead** is active data is coming into the analyzer from the memory. If it is inactive the analyzer is driving data out to the memory.

10.6.6.3 Cache Interface

Signal	Dir	Width	Description
UMCoCaReady	IN	1	Unified Memory Controller to Cache Ready. This signal tells the Cache that during a read, the data on the **UMCoCaData** bus is valid and during a write that the Unified Memory Controller has latched the data on the **CaUMCData** bus.
UMCoCaData	IN	*	Unified Memory Controller to Cache Data bus. * Uses AN_CA_DWIDTH
CaUMCData	OUT	*	Cache to Unified Memory Controller Data bus. * Uses AN_CA_DWIDTH
CaUMCAdd	OUT	*	Cache to Unified Memory Controller Address bus. * Uses AN_CA_AWIDTH
CaMemReq	OUT	1	Cache Memory Request. If this signal is active, the address on the **CaUMCAdd** bus is valid.
CaMemWr	OUT	1	Cache Memory Write. If this signal is active, the current transaction is a write..

10.6.7 Verilog Module

```
module UMC(Reset_N, MCLK ,AnalyzerEn ,MemClkIn ,MemRAS_N ,MemCAS_N
,MemClkEn ,MemClkOut ,MemWR_N ,MemBA ,MemDSF ,MemByteEn_N ,MemAddress
```

```
,MemDataIn ,MemDataOut ,MemDirRead ,UMCoCaReady ,UMCoCaData ,CaUMCData
,CaUMCAdd ,CaMemReq ,CaMemWr);

// General Interface Interface
input Reset_N;
input MCLK;
input AnalyzerEn;
// Memory Interface
input MemClkIn;
output [`AN_MEM_RASWIDTH-1 : 0] MemRAS_N;
output [`AN_MEM_CASWIDTH-1 : 0] MemCAS_N;
output MemClkEn;
output MemClkOut;
output MemWR_N;
output MemBA;
output MemDSF;
output [`AN_MEM_BEWIDTH-1 : 0] MemByteEn_N;
output [`AN_MEM_AWIDTH-1 : 0] MemAddress;
input [`AN_MEM_DWIDTH-1 : 0] MemDataIn;
output [`AN_MEM_DWIDTH-1 : 0] MemDataOut;
output MemDirRead;
// Cache Interface
input UMCoCaReady;
input [`AN_CA_DWIDTH-1 : 0] UMCoCaData;
output [`AN_CA_DWIDTH-1 : 0] CaUMCData;
output [`AN_CA_AWIDTH-1 : 0] CaUMCAdd;
output CaMemReq;
output CaMemWr;
```

10.6.8 VHDL Component

10.7 Cache

10.7.1 Symbol

10.7.2 Highlights

- Fully associative
- True least recently used cache updating
- Simultaneous one write and two reads.

10.7.3 Description

The Cache module contains a fully associative, true LRU cache memory. Full associatively is achieved through the use of a content addressable memory (CAM). The need for a fully associative cache arises from the fact that the hash uses to generate the initial look up into the flow entry database spreads the entries pseudo randomly throughout the memory. Each hash value corresponds to a bucket containing N flow entries. N is set by the designer (see section xxx).

The Cache can service two read transfers at one time. If there are more than two read requests active at one time the Cache services them in the order shown in section xxx.

The CAM contains the hash value associated with the corresponding bucket in the cache memory. When there is a cache hit, the CAM produces the most significant bits of the address in cache memory where the bucket is stored. The cache then accesses the cache memory at the address indicated concatenating the lower address bits provided by the requesting module. The cache then remembers that the requesting module had a cache hit and the memory location returned. This allows a cache lookup for a requesting module to occur only once per request. When the requesting module requires a different bucket, it drops then again raises its request and another CAM cycle is initiated.

The least recently used algorithm requires the CAM to also be a stack. When there is a cache hit the CAM location that produced the hit is put on the top of the stack. The other locations above the hit location are shifted down to fill in the gap. If there is a miss, the bottom location is read to determine the address in the cache memory to put the new bucket. All the locations shifted down as normally. Finally the new hash value and cache memory address are put at the top of the stack.

10.7.3.1 Priority

The Cache processes requests from the attached modules in the following order:

1 - LRU dirty write back. The Cache writes back the least recently used bucket if it is dirty so that there will always be a space for the fetching of cache misses.
2 – Lookup and Update Engine.
3 – State Processor.
4 – Flow Insertion and Deletion Engine.
5 – Analyzer CPU Interface and Control
6 – Dirty write back from LRU –1 to MRU. When there is nothing else pending the Cache writes dirty entries back to memory.

10.7.4 Implementation Information

10.7.5 File Names

Top: Cache.v(hd)
Uses: AnalyzerConstants.v(hd)

10.7.6 Pin Descriptions

10.7.6.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low.
MCLK	IN	1	Module Clock.
AnalyzerEn	IN	1	Analyzer Enable bit from the control register

10.7.6.2 Unified Memory Controller Interface

Signal	Dir	Width	Description
UMCoCaReady	OUT	1	Unified Memory Controller to Cache Ready. This signal tells the Cache that during a read, the data on the **UMCoCaData** bus is valid and during a write that the Unified Memory Controller has latched the data on the **CaUMCData** bus.
UMCoCaData	OUT	*	Unified Memory Controller to Cache Data bus. * Uses AN_CA_DWIDTH
CaUMCData	IN	*	Cache to Unified Memory Controller Data bus. * Uses AN_CA_DWIDTH
CaUMCAdd	IN	*	Cache to Unified Memory Controller Address bus. * Uses AN_CA_AWIDTH
CaMemReq	IN	1	Cache Memory Request. If this signal is active, the address on the **CaUMCAdd** bus is valid.
CaMemWr	IN	1	Cache Memory Write. If this signal is active, the current transaction is a write..

10.7.6.3 Flow Insertion and Deletion Engine Interface

Signal	Dir	Width	Description
CaFIDEReady	OUT	1	Cache to Flow Insertion and Deletion Engine Ready. This signal tells the Flow Insertion and Deletion Engine that during a read, the data on the **CaFIDEData** bus is valid and during a write that the Cache has latched the data on the **FIDEnCaData** bus.
CaFIDEData	OUT	*	Cache to Flow Insertion and Deletion Engine Data bus. * Uses AN_CA_DWIDTH
FIDEnCaData	IN	*	Flow Insertion and Deletion Engine to Cache Data bus. * Uses AN_CA_DWIDTH
FIDEAdd	IN	*	Flow Insertion and Deletion Engine to Cache Address bus. * Uses AN_CA_AWIDTH

10.7.6.3 **Flow Insertion and Deletion Engine Interface**

Signal	Dir	Width	Description
FIDEMemReq	IN	1	Flow Insertion and Deletion Engine Memory Request. If this signal is active, the address on the **FIDEAdd** bus is valid.
FIDEMemWr	IN	1	Flow Insertion and Deletion Engine Memory Write. If this signal is active, the current transaction is a write..

10.7.6.4 **Analyzer CPU Interface Control Interface**

Signal	Dir	Width	Description
CaACICReady	OUT	1	Cache to Analyzer CPU Interface Control Ready. This signal tells the Analyzer CPU Interface Control that during a read, the data on the **CaACICData** bus is valid and during a write that the Cache has latched the data on the **ACICnCaData** bus.
CaACICData	OUT	*	Cache to Analyzer CPU Interface Control Data bus. * Uses AN_CA_DWIDTH
ACICoCaData	IN	*	Analyzer CPU Interface Control to Cache Data bus. * Uses AN_CA_DWIDTH
ACICAdd	IN	*	Analyzer CPU Interface Control to Cache Address bus. * Uses AN_CA_AWIDTH
ACICMemReq	IN	1	Analyzer CPU Interface Control Memory Request. If this signal is active, the address on the **ACICAdd** bus is valid.
ACICMemWr	IN	1	Analyzer CPU Interface Control Memory Write. If this signal is active, the current transaction is a write..

10.7.6.5 **Lookup Engine Interface**

Signal	Dir	Width	Description
CaLUEReady	OUT	1	Cache to Lookup Engine Ready. This signal tells the Lookup Engine that during a read, the data on the **CaLUEData** bus is valid and during a write that the Cache has latched the data on the **LUEnCaData** bus.
CaLUEData	OUT	*	Cache to Lookup Engine Data bus. * Uses AN_CA_DWIDTH
LUEnCaData	IN	*	Lookup Engine to Cache Data bus. * Uses AN_CA_DWIDTH
LUEAdd	IN	*	Lookup Engine to Cache Address bus. * Uses AN_CA_AWIDTH
LUEMemReq	IN	1	Lookup Engine Memory Request. If this signal is active, the address on the **LUEAdd** bus is valid.
LUEMemWr	IN	1	Lookup Engine Memory Write. If this signal is active, the current transaction is a write..

10.7.6.6 State Processor Interface

Signal	Dir	Width	Description
CaSPReady	OUT	1	Cache to State Processor Ready. This signal tells the Lookup Engine that during a read, the data on the **CaSPData** bus is valid and during a write that the Cache has latched the data on the **SPnCaData** bus.
CaSPData	OUT	*	Cache to State Processor Data bus. * Uses AN_CA_DWIDTH
SPnCaData	IN	*	State Processor to Cache Data bus. * Uses AN_CA_DWIDTH
SPAdd	IN	*	State Processor to Cache Address bus. * Uses AN_CA_AWIDTH
SPMemReq	IN	1	State Processor Memory Request. If this signal is active, the address on the **SPAdd** bus is valid.
SPMemWr	IN	1	State Processor Memory Write. If this signal is active, the current transaction is a write..

10.7.7 Verilog Module

```
module Cache(Reset_N, MCLK ,AnalyzerEn ,UMCoCaReady ,UMCoCaData
,CaUMCData ,CaUMCAdd ,CaMemReq ,CaMemWr ,CaFIDEReady ,CaFIDEData
,FIDEnCaData ,FIDEAdd ,FIDEMemReq ,FIDEMemWr ,CaACICReady ,CaACICData
,ACICoCaData ,ACICAdd ,ACICMemReq ,ACICMemWr ,CaLUEReady ,CaLUEData
,LUEnCaData ,LUEAdd ,LUEMemReq ,LUEMemWr ,CaSPReady ,CaSPData ,SPnCaData
,SPAdd ,SPMemReq ,SPMemWr);

// General Interface Interface
input Reset_N;
input MCLK;
input AnalyzerEn;
// Unified Memory Controller Interface
output UMCoCaReady;
output [`AN_CA_DWIDTH-1 : 0] UMCoCaData;
input [`AN_CA_DWIDTH-1 : 0] CaUMCData;
input [`AN_CA_AWIDTH-1 : 0] CaUMCAdd;
input CaMemReq;
input CaMemWr;
// Flow Insertion and Deletion Engine Interface
output CaFIDEReady;
output [`AN_CA_DWIDTH-1 : 0] CaFIDEData;
input [`AN_CA_DWIDTH-1 : 0] FIDEnCaData;
input [`AN_CA_AWIDTH-1 : 0] FIDEAdd;
input FIDEMemReq;
input FIDEMemWr;
// Analyzer CPU Interface Control Interface
output CaACICReady;
output [`AN_CA_DWIDTH-1 : 0] CaACICData;
input [`AN_CA_DWIDTH-1 : 0] ACICoCaData;
input [`AN_CA_AWIDTH-1 : 0] ACICAdd;
input ACICMemReq;
```

```
input ACICMemWr;
// Lookup Engine  Interface
output CaLUEReady;
output [`AN_CA_DWIDTH-1 : 0] CaLUEData;
input [`AN_CA_DWIDTH-1 : 0] LUEnCaData;
input [`AN_CA_AWIDTH-1 : 0] LUEAdd;
input LUEMemReq;
input LUEMemWr;
// State Processor  Interface
output CaSPReady;
output [`AN_CA_DWIDTH-1 : 0] CaSPData;
input [`AN_CA_DWIDTH-1 : 0] SPnCaData;
input [`AN_CA_AWIDTH-1 : 0] SPAdd;
input SPMemReq;
input SPMemWr;
```

10.7.8 VHDL Component

10.8 State Processor - SP

10.8.1 Symbol

10.8.2 Highlights

- Flexible Rule-based Traffic Classification
- State-based Tracking of Traffic
- ***Multiple*** Packets for Layer Processing
- Programmable Rules/State Processor
- Selectable Protocols in Flows
- Future Protocols Support

10.8.3 Description

The State Processor module analyzes both new and existing flows in order to classify them by application. It does this by proceeding from state to state based on rules defined by the engineer. A rule is a test followed by the next state to proceed to if the test is true. The State Processor goes through each rule until the test is true or there are no more tests to perform. The State Processor starts the process by using the last protocol recognized by the Parser as an offset into a jump table. The jump table takes us to the instructions to use for that protocol. Most instructions test something in the Unified Flow Key Buffer or the flow entry if it exists. The State Processor may have to test bits, do comparisons, add or subtract to perform the test.

10.8.4 Architecture

The State Processor contains several sub-blocks:

10.8.4.1 Scratch Pad Registers

The State Processor contains four scratch pad registers. These registers are the source and/or the destination for all instructions. It is implemented as a register file with one write and two read ports.

10.8.4.2 Instruction Pointer and Stack

The Instruction Pointer is used to point to the State Processor Instruction Database address that the State Processor is executing. The Instruction Pointer is initialized with the last protocol recognized by the Parser. This first instruction is a jump to the subroutine where the protocol is decoded. The State Processor supports calls so the Instruction Pointer block contains a two level stack. A one bit stack pointer points to the top of the stack that the Instruction Pointer is pushed to or popped from.

10.8.4.3 Flag Register

The Flag Register contains several bits used for conditional branching.

10.8.4.3.1 Flag Register Word Definition

Bit	Description

10.8.4.3.1 Flag Register Word Definition

Bit	Description

10.8.4.4 Compare Block

The Compare Block compares two operands by exclusive-oring them together. The Compare Mask Register is contained in this block. If a bit is set in the Compare Mask Register, that bit is ignored in the compare operation.

10.8.4.5 Flow Key Pointer

The Flow Key Pointer provides the address that the State Processor is accessing in the Unified Flow Key Buffer. The Flow Key Pointer can perform both direct and indirect addressing. Indirect addressing is used to offset into a protocols' header.

10.8.4.6 Flow Entry Pointer

The Flow Entry Pointer provides the address that the State Processor is accessing in the Flow Entry in the Cache. If the flow entry exists, the upper address bits come from the hash used to lookup the bucket in the Flow database. The middle bits come from the bucket entry found. The lower bits come from the offset the State Processor is using.

10.8.5 Instruction Definitions

The following sections describe the instructions available in the State Processor. It should be noted that no assembler is provided for the State Processor. This is because the engineer need not write code for this processor. The MeterFlow Compiler writes the database entered into the State Processor Instruction Database from the protocols defined in the Protocol List.

10.8.5.1 Jump

This instruction causes the Instruction Pointer to be loaded with the address in the JumpAddress field of the State Processor Instruction Database. This instruction is always conditional. Whether the branch is taken or not depends on the on the ConditionCode field in the instruction and the state of the flag register.

10.8.5.2 Call

This instruction causes the Instruction Pointer to be loaded with the address in the JumpAddress field of the State Processor Instruction Database. At the same time the current address in the Instruction Pointer is pushed onto the stack. This instruction is always conditional. Whether the call is taken made or not depends on the on the ConditionCode field in the instruction and the state of the flag register.

10.8.5.3 Return

This instruction causes the Instruction Pointer to be loaded with the address at the top of the stack. This instruction is always conditional. Whether the return is executed or not depends on the on the ConditionCode field in the instruction and the state of the flag register.

10.8.5.4 Copy

The Copy instruction moves data from:

- Flow Key to Scratch Pad Register
- Cache to Scratch Pad Register
- ImmediateData to Scratch Pad Register
- Scratch Pad Register to Flow Key
- Scratch Pad Register to Cache
- Scratch Pad Register to Compare Mask Register

The external address can be either a direct or indirect access.

10.8.5.5 Compare

This instruction compares two operands . The operands must be either from a Scratch Pad Register or an immediate value from the instruction's ImmediateData field. The Compare Mask Register is used to set bit to don't care.

10.8.5.6 Instruction Word Definition

Bit	Description

10.8.6 Implementation Information

10.8.7 File Names

Top: SP.v(hd)
Uses: AnalyzerConstants.v(hd)

10.8.8 Pin Descriptions

10.8.8.1 General Interface Signals

Signal	Dir	Width	Description
Reset_N	IN	1	Reset - active low.
MCLK	IN	1	Module Clock.
AnalyzerEn	IN	1	Analyzer Enable bit from the control register

10.8.8.2 Unified Flow Key Buffer Interface

Signal	Dir	Width	Description

10.8.8.2 Unified Flow Key Buffer Interface

Signal	Dir	Width	Description
UFKBuSPData	IN	*	Unified Flow Key Buffer to State Processor read Data bus. * Uses AN_UFKB_AWIDTH
SPrUFKBData	OUT	*	State Processor to Unified Flow Key Buffer write Data bus. * Uses AN_ UFKB _AWIDTH
SPrUFKBAdd	OUT	*	State Processor to Unified Flow Key Buffer Address bus. * Uses AN_ UFKB _AWIDTH
SPFlowKeyAv	IN	1	State Processor Flow Key Available. This signal tells the State Processor that the Unified Flow Key Buffer module a flow key for it to process.
SPrUFKBWrStb	OUT	1	State Processor to Unified Flow Key Buffer Write Strobe.
SPDone	OUT	1	State Processor Done. This input is used to tell the Unified Flow Key Buffer that the State Processor has finished with the current flow. The Unified Flow Key Buffer also uses this signal to increment it's internal pointer so that the next address from State Processor will point to the next flow buffer.
SPHoldBuf	OUT	1	State Processor Hold Buffer. This input is used to tell the Unified Flow Key Buffer that the State Processor is transferring processing of this buffer to the Flow Insertion and Deletion Engine.

10.8.8.3 Cache Interface

Signal	Dir	Width	Description
CaSPReady	IN	1	Cache to State Processor Ready. This signal tells the Lookup Engine that during a read, the data on the **CaSPData** bus is valid and during a write that the Cache has latched the data on the **SPnCaData** bus.
CaSPData	IN	*	Cache to State Processor Data bus. * Uses AN_CA_DWIDTH
SPrCaData	OUT	*	State Processor to Cache Data bus. * Uses AN_CA_DWIDTH
SPAdd	OUT	*	State Processor to Cache Address bus. * Uses AN_CA_AWIDTH
SPMemReq	OUT	1	State Processor Memory Request. If this signal is active, the address on the **SPAdd** bus is valid.
SPMemWr	OUT	1	State Processor Memory Write. If this signal is active, the current transaction is a write..

10.8.8.4 State Processor Interface

Signal	Dir	Width	Description
SPIDData	IN	*	State Processor to State Processor Instruction Database Data bus * uses AN_SPID_DWIDTH
SPrSPIDAdd	OUT	*	State Processor to State Processor Instruction Database Address bus * uses AN_SPID_AWIDTH

10.8.9 Verilog Module

```
module SP(Reset_N, MCLK ,AnalyzerEn ,UFKBuSPData ,SPrUFKBData
,SPrUFKBAdd ,SPFlowKeyAv ,SPnUFKBWrStb ,SPDone ,SPHoldBuf ,CaSPData
,SPrCaData ,SPAdd ,SPMemReq ,SPMemWr);

// General Interface Interface
input Reset_N;
input MCLK;
input AnalyzerEn;
// Unified Flow Key Buffer Interface
input [`AN_UFKB_DWIDTH-1 : 0] UFKBuSPData;
output [`AN_UFKB_DWIDTH-1 : 0] SPrUFKBData;
output [`AN_UFKB_AWIDTH-1 : 0] SPrUFKBAdd;
input SPFlowKeyAv;
output SPrUFKBWrStb;
output SPDone;
output SPHoldBuf;
// Cache Interface
input CaSPReady;
input [`AN_CA_DWIDTH-1 : 0] CaSPData;
output [`AN_CA_DWIDTH-1 : 0] SPrCaData;
output [`AN_CA_AWIDTH-1 : 0] SPAdd;
output SPMemReq;
output SPMemWr;
// State Processor Instruction Database
input [`AN_SPID_DWIDTH-1 : 0] SPIDData;
output [`AN_SPID_AWIDTH-1 : 0] SPrSPIDAdd;
```

10.8.10 VHDL Component

11 Appendix A - Multi-Packet State Processing

11.1 Overview

The MeterFlow Accelerator system is composed of four major subsystems. Each system interacts with the others by passing specific information and identification to parse, extract, generate flows and analyze single or multiple packets in data flow on a communications network.

One of the major subsystems is the Analyzer. This component is responsible for creating and maintaining classified traffic flows, processing statistics for packets and flows, managing the traffic flow database and cache, and performing state-based analysis of traffic flows.

This document describes the processes required for recognizing and maintaining state information for traffic flows. There are several different processes, which are detailed in the following sections.

11.2 Analyzer Data Input Requirements

In order for the Analyzer to successfully classify traffic by application, there are several data elements required from each packet to be analyzed. Prior to sending a packet of information to the Analyzer, all additional information must be formatted and sent along with the appropriate packet content.

The Analyzer must specifically receive each packets in a conversation in the order which they are exchange between the client and the server. The order is crucial for proper state based classification.

11.3 State-base Traffic Classification

More applications running over data networks utilize complex methods of classifying traffic through the creation of multiple states. The creation of the state based traffic classification causes the need for managing and maintaining learned states from traffic derived in the network.

There are several different methods in place for the creation of states in client/server network traffic. Even though there are several different methods for the creation of state. It is possible to isolate these different approaches into two basic categories.

The first category is commonly referred to as "server announcement". In the server announcement mode there are messages which are put out onto the network, in either a broadcast or multicast approach which, all stations in the network receive and decode to derive the appropriate connection point for communicating for that particular application, with the particular server. There are several examples for this type of server announcement implementation with state based protocols. Using the server announcement method, a particular application communicates using a service channel, in the form of a TCP or UDP socket or Port as in the IP protocol suite, or using a SAP as in the Novell IPX protocol suite.

The second category is referred to as "in-stream analysis". This method is used either as a primary or secondary recognition process. As a primary process, in-stream analysis assists in extracting detailed information which will be used to further recognize both the specific application and application component. A good example of in-stream analysis is any Web-based applications. The commonly used Pointcast Web information application can be recognized using this process. During the initial connection between a Pointcast server and client, specific key tokens exist in the data exchange that will result in a signature for Pointcast.

The in stream analysis process may also be combined with the server announcement process. In many cases in stream analysis will augment other recognition processes. An example of combining in stream analysis with server announcement can be found in business applications such as SAP and BAAN.

11.3.1 Session Tracking

One of the primary processes for tracking applications in the stream of the client/server packet exchange, is through session tracking. The process of tracking sessions requires an initial connection to a predefined socket or Port. This method of communication is used in a variety of transport layer protocols. It is most commonly seen in the TCP and UDP transports of the IP protocol.

During the process of session tracking, a client will make the request of a server using a specific Port or socket number. This initial request will cause the server to create a TCP or UDP Port to exchange the remainder of the data between the client and the server. The server then replies to the request of the client using this newly created Port. The original Port used by the client to connect to server will never be used again during this data exchange.

One of the best examples of session tracking is TFTP. During the client/server exchange process of TFTP, a specific Port is always used to initiate the conversation. When the client begins the process of communicating, a request is made to UDP Port 67. Once the server receives this request, a new Port is created on the server. The server then replies to the client using the new Port. In this example, it is clear that in order to recognize TFTP the process must analyze the initial request from the client. Also, the reply from the server with the key Port information must be analyzed and used to create a key for monitoring the remainder of this data exchange.

Another important component in session tracking is the understanding of the current state for particular connections in the network. Many of the application protocols, which can be monitored, are transported via protocols that have built-in state information. An example of such a transport protocol is TCP. This transport provides a reliable means of sending information between a client and a server. When he data exchange is initiated a TCP request for synchronization message is sent. This message contains a specific sequence number that is used to track and acknowledgement from the server. Once the server has knowledge to the synchronization request, data is exchange between the client and the server. When communications are no longer required, the client would send a finish or complete message to the server. The server willing knowledge this finish request, with a reply containing the sequence numbers from the request. This sequence of events is known as a connection oriented data exchange. Many of the events used to track the state in a conversation are directly related to these types of connection and maintenance messages.

All of the processes discussed above are required to track sessions. The capability to track sessions is a requirement for understanding the current state to analyze.

11.3.2 Server Announcement

The process of server announcement consists of a server with multiple applications, which are all required to be simultaneously accessed from multiple clients. Many applications are beginning to use this process as a means of multiplexing a single Port or socket into many applications and services. The individual methods of server announcement protocols tend very. However, the basic underlying process remains similar between all of these different announcement exchanges.

11.3.2.1 Sun RPC Analysis

Sun-RPC and Net-RPC are to good examples of server announcement oriented communications processes. In this section we will analyze the requirements for recognizing applications which utilize the sun implementation of RPC. RPC stands for remote procedure call. This is a quite clear description of the process. A remote or client that wishes to use a server or procedure must establish a connection using the RPC protocol.

Using the Sun-RPC protocol as a model for server announcement is completed through the following process. Each server running the Sun-RPC protocol must maintain a process and database called the Port Mapper. The Port Mapper creates a direct association between a Sun-RPC program or application and a TCP or UDP socket or Port. An application or program number is a 32-bit unique identifier assigned by IANA. Each Port Mapper on a Sun-RPC server can present the mappings between a unique program number and a specific transport socket through the use of specific request or a directed announcement.

The first approach we will review is the specific request method. Using this process the client makes a specific request to the server on a predefined UDP or TCP socket. Once the Port Mapper process on the sun RPC server receives the request, the specific mapping is returned in a directed reply to the client.

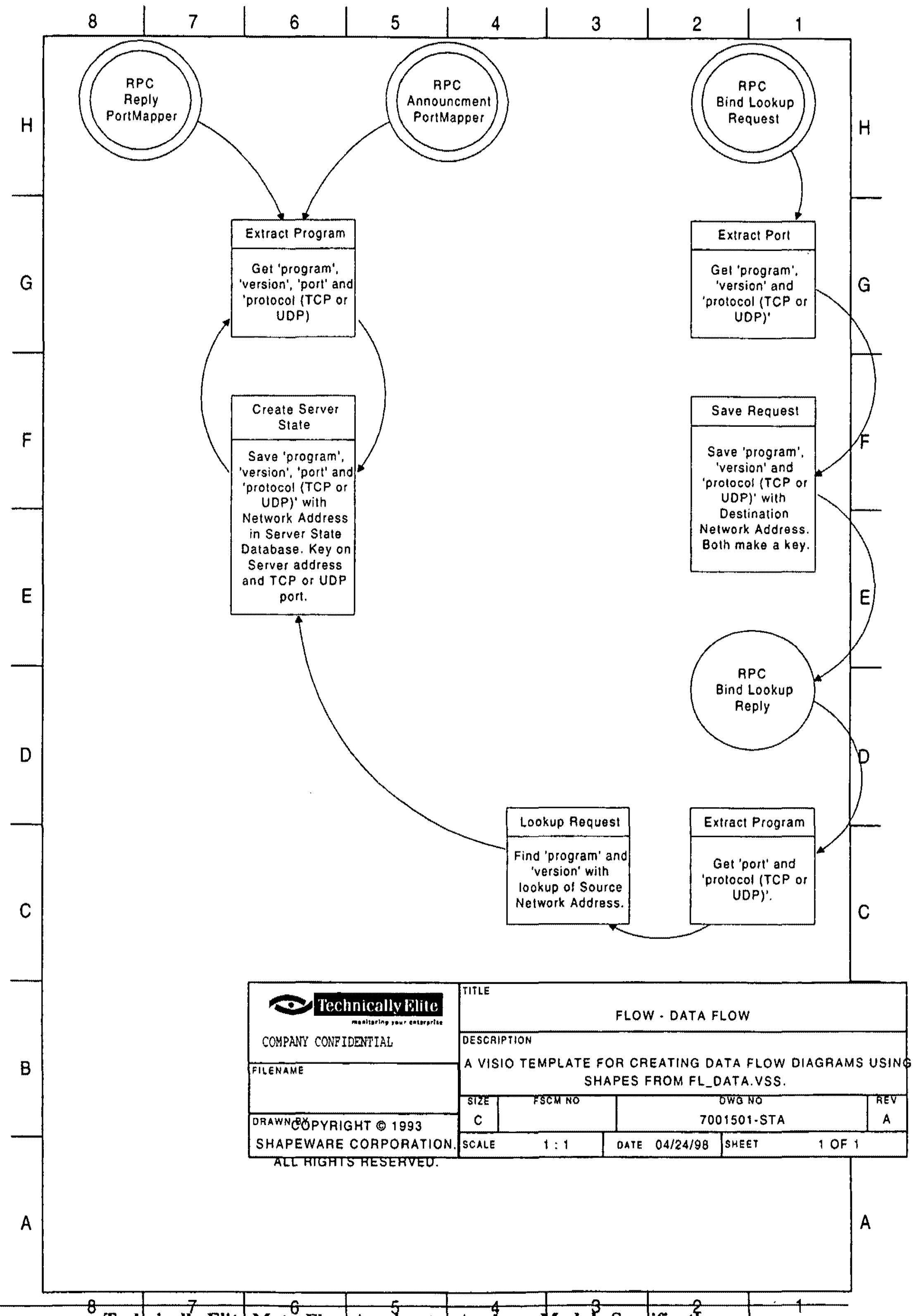
1) A client sends a TCP packet to Port 111, with an RPC Bind Lookup Request.

2) The server extracts the program identifier and version identifier from the request. The server also uses the fact that this packet came in the using the TCP transport.

3) The server sends a TCP packet to Port 111, with an RPC Bind Lookup Reply. The reply contains the specific ports on which future transactions will be accepted for the specific RPC program identifier.

11.3.2.2 Process for Sun RPC Analysis

1. Decode Sun RPC by TCP or UDP Port 111
2. Check RPC type field for Id
3. If value is PortMapper, save paired socket (i.e. dest for dest, src for src)
4. Decode ports and mapping, save ports with socket/addr key
5. There may be more than one pairing per mapper packet
6. Saving is complete


RPC Reply PortMapper
RPC Announcment PortMapper
RPC Bind Lookup Request
Extract Program
Get 'program', 'version', 'port' and 'protocol (TCP or UDP)
Extract Port
Get 'program', 'version' and 'protocol (TCP or UDP)'
Create Server State
Save 'program', 'version', 'port' and 'protocol (TCP or UDP)' with Network Address in Server State Database. Key on Server address and TCP or UDP port.
Save Request
Save 'program', 'version' and 'protocol (TCP or UDP)' with Destination Network Address. Both make a key.
RPC Bind Lookup Reply
Lookup Request
Find 'program' and 'version' with lookup of Source Network Address.
Extract Program
Get 'port' and 'protocol (TCP or UDP)'.
Technically Elite
COMPANY CONFIDENTIAL
FILENAME
DRAWN BY
COPYRIGHT © 1993 SHAPEWARE CORPORATION. ALL RIGHTS RESERVED.
TITLE
FLOW - DATA FLOW
DESCRIPTION
A VISIO TEMPLATE FOR CREATING DATA FLOW DIAGRAMS USING SHAPES FROM FL_DATA.VSS.
SIZE C
FSCM NO
DWG NO 7001501-STA
REV A
SCALE 1 : 1
DATE 04/24/98
SHEET 1 OF 1

[redacted]

PortMapper Protocol Specification (in RPC Language)

const PMAP_PORT = 111; /* portmapper port number */

A mapping of (program, version, protocol) to port number:

struct mapping { unsigned int prog; unsigned int vers; unsigned int prot; unsigned int port; };

Supported values for the "prot" field:

const IPPROTO_TCP = 6; /* protocol number for TCP/IP */ const IPPROTO_UDP = 17; /* protocol number for UDP/IP */ A list of mappings: struct *pmaplist { mapping map; pmaplist next; };

Arguments to callit: struct call_args { unsigned int prog; unsigned int vers; unsigned int proc; opaque args<>; }; Results of callit: struct call_result { unsigned int port; opaque res<>; }; Port mapper procedures: program PMAP_PROG { version PMAP_VERS { void PMAPPROC_NULL(void) = 0; bool PMAPPROC_SET(mapping) = 1; bool PMAPPROC_UNSET(mapping) = 2; unsigned int PMAPPROC_GETPORT(mapping) = 3; pmaplist PMAPPROC_DUMP(void) = 4; call_result PMAPPROC_CALLIT(call_args) = 5; } = 2; } = 100000;A.2 Port Mapper Operation The portmapper program currently supports two protocols (UDP and TCP). The portmapper is contacted by talking to it on assigned port number 111 (SUNRPC) on either of these protocols. The following is a description of each of the portmapper

Sun RPC Decode Process

1) Parse frame to TCP or UDP
2) Lookup paired sockets if no standard match
3) If RPC found, same new Key

The port mapper program maps RPC program and version numbers to transport-specific port numbers. This program makes dynamic binding of remote programs possible. This is desirable because the range of reserved port numbers is very small and the number of potential remote programs is very large. By running only the port mapper on a reserved port, the port numbers of other remote programs can be ascertained by querying the port mapper. The port mapper also aids in broadcast RPC. A given RPC program will usually have different port number bindings on different machines, so there is no way to directly broadcast to all of these programs. The port mapper, however, does have a fixed port number. So, to broadcast to a given program, the client actually sends its message to the port mapper located at the broadcast address. Each port mapper that picks up the broadcast then calls the local service specified by the client. When the port mapper gets the reply from the local service, it sends the reply on back to the client.

PortMapper Protocol Specification (in RPC Language)

const PMAP_PORT = 111; /* portmapper port number */

A mapping of (program, version, protocol) to port number:

struct mapping { unsigned int prog; unsigned int vers; unsigned int prot; unsigned int port; };

Supported values for the "prot" field:

const IPPROTO_TCP = 6; /* protocol number for TCP/IP */ const IPPROTO_UDP = 17; /* protocol number for UDP/IP */ A list of mappings: struct *pmaplist { mapping map; pmaplist next; };

Arguments to callit: struct call_args { unsigned int prog; unsigned int vers; unsigned int proc; opaque args<>; }; Results of callit: struct call_result { unsigned int port; opaque res<>; }; Port mapper procedures: program PMAP_PROG { version PMAP_VERS { void PMAPPROC_NULL(void) = 0; bool PMAPPROC_SET(mapping) = 1; bool PMAPPROC_UNSET(mapping) = 2; unsigned int PMAPPROC_GETPORT(mapping) = 3; pmaplist PMAPPROC_DUMP(void) = 4; call_result PMAPPROC_CALLIT(call_args) = 5; } = 2; } = 100000;

11.3.3 Port Mapper Operation

The portmapper program currently supports two protocols (UDP and TCP). The portmapper is contacted by talking to it on assigned port number 111 (SUNRPC) on either of these protocols. The following is a description of each of the portmapper procedures: PMAPPROC_NULL: This procedure does no work. By convention, procedure zero of any protocol takes no parameters and returns no results.

PMAPPROC_SET: When a program first becomes available on a machine, it registers itself with the port mapper program on the same machine. The program passes its program number "prog", version number "vers", transport protocol number "prot", and the port "port" on which it awaits service request. The procedure returns a boolean reply whose value is "TRUE" if the procedure successfully established the mapping and "FALSE" otherwise. The procedure refuses to establish a mapping if one already exists for the tuple "(prog, vers, prot)".

PMAPPROC_UNSET: When a program becomes unavailable, it should unregister itself with the port mapper program on the same machine. The parameters and results have meanings identical to those of "PMAPPROC_SET". The protocol and port number fields of the argument are ignored.

PMAPPROC_GETPORT: Given a program number "prog", version number "vers", and transport protocol number "prot", this procedure returns the port number on which the program is awaiting call requests. A port value of zeros means the program has not been registered. The "port" field of the argument is ignored. PMAPPROC_DUMP: This procedure enumerates all entries in the port mapper's database. The procedure takes no parameters and returns a list of program, version, protocol, and port values. PMAPPROC_CALLIT: This procedure allows a client to call another remote procedure on the same machine without knowing the remote procedure's port number. It is intended for supporting broadcasts to arbitrary remote programs via the well-known port mapper's port. The parameters "prog", "vers", "proc", and the bytes of "args" are the program number, version number, procedure number, and parameters of the remote procedure. Note: (1) This procedure only sends a reply if the procedure was successfully executed and is silent (no reply) otherwise. (2) The port mapper communicates with the remote program using UDP only. The procedure returns the remote program's port number, and the reply is the reply of the remote procedure.

11.3.4 Service Announcement

Service announcement method of the application recognition is very similar to server announcement. One specific difference in service announcement is that the announcements are made regularly and contain fixed information. Also, service announcement based applications only provide the key information for locating applications in each announcement. There is no capability to request a specific service. Each client must learn the key information required to access an application.

Novell's IPX SAP is a good example of service announcement oriented communications process. A Novell server will have many different services, which it may provide to clients on network. IPX uses service access points or SAP as a way to identify specific applications and services.

11.3.5 In-stream Recognition and Extraction

The process of identifying more of the business applications on networks today requires analysis of information in stream of the network data. Simply, this means that in order to contain the visibility to application traffic flow, a process must routinely analyze the network stream itself.

SMB is a protocol used to in networks today which has textual information during the data exchange that can be used to further determine the type of end-user application involved in communications. An SMB packet is usually transported above the NetBIOS session protocol. Inside the SMB header is a function code. This function code is one octet in length and assists in the classification of the type of SMB data in the payload.

11.3.5.1 Web-based Applications

The best example of applications requiring in-stream recognition mainly Web-based. These applications generally utilize two well-known ports for all conversations. Because of this, they can be considered multiplex ports. There is one big difference, the client and server have no well-known exchange mechanism outside of the normal data stream. Therefore, these applications require combining session tracking and in stream recognition to derive end-user application.

As discussed earlier, point cast is one of the most widely used Web based application. The steps required to detail point cast can be rid repeated for other Web based applications. This is also a good example for understanding the process used in combining session tracking with other recognition techniques.

The process begins when a client Web the browser initiates a request to a point cast Web server. This request

- **Exhibit A4:** Protocol Tracking Summary (Document MFAProtocolLayout.pdf)

		TCP/IP Access					Novell Access					Unix	DceRPC	Misc	
Description	Type	HTTP Content	HTTP Pattern	Well Known TCP	Well Known UDP	Other	Well Known IPX/SPX	SAP Mapped	Well Known VIP/VSPP	Well Known VIPC	Other	Port Mapped	Endpoint Mapped	New Parents	Other
2.0 Alternate HTTP	Well-Known			591,8008,8080											
14.0 Sybase Adaptive SQL Anywhere (ASA) Server	Well-Known			1498, 2638			0x80c5	0x0328							
14.2 Tabular Domain Stream (TDS)	Well-Known													Sybase,Msql	
14.2 Command Sequence (CmdSeq)	Well-Known													Sybase ASA	
15.0 Sybase Adaptive SQL Enterprise (ASE) Server	Well-Known			tbd			tbd	tbd	tbd	tbd					
16.0 Microsoft SQL Server	Well-Known			1433			*	core #11	tbd	tbd					
17.0 DB2	Well-Known			523,3700,3701			0x879e-87a2		tbd	tbd					
19.0 Cisco Dynamic ISL (DISL)	Well-Known													vsnap - 2000	
19.0 Cisco gateway Discovery Protocol (CDP)	Well-Known													vsnap - 2004	
20.0 Novell IPXWAN (RFC 1634)	Well-Known						0x9004							vsnap - 2004	
27.1 MS-Exchange – POP3 Mail	Well-Known			transparent			tbd	tbd	tbd	tbd					
27.1 MS-Exchange – SMTP Mail	Well-Known			transparent			tbd	tbd	tbd	tbd					
27.1 MS-Exchange – IMAP4 Mail	Well-Known			transparent			tbd	tbd	tbd	tbd					
27.1 MS-Exchange – LDAP	Well-Known			transparent			tbd	tbd	tbd	tbd					
27.1 MS-Exchange – ISO over TCP/IP	Well-Known			transparent			tbd	tbd	tbd	tbd					
27.1 MS-Exchange – X.400	Well-Known			transparent			tbd	tbd	tbd	tbd					
27.1 MS-Exchange – DCE Endpoint Mapping	Well-Known			transparent			tbd	tbd	tbd	tbd					
32.0 Vines Token-Ring (vtr)	Well-Known													llc - 0xbc	
33.0 SMTP over SSL	Well-Known			465											
33.0 NNTP over SSL	Well-Known			563											
33.0 Shell over SSL	Well-Known			615											
33.0 LDAP over SSL	Well-Known			636	636										
33.0 FTP-data over SSL	Well-Known			939											
33.0 FTP-control over SSL	Well-Known			990											
33.0 Telnet over SSL	Well-Known			992											
33.0 IMAP4 over SSL	Well-Known			993											
33.0 IRC over SSL	Well-Known			994	994										
33.0 POP3 over SSL	Well-Known			995	995										
34.0 VPN - PPTP	Well-Known			1723											
35.0 Cu-SeeMe	Well-Known			7648, 7649	7648,7649,24032										
37.0 PcAnywhere	Well-Known			5631	5632										
38.0 Timbuktu	Well-Known			1417-1420	407, 1419										
39.1 StreamWorks	Well-Known				1558										
40.0 VDOLive	Well-Known			7000	tbd										
41.0 FreeTell	Well-Known				21300-21303										
42.0 Microsoft System Mgmt Server	Well-Known			1761-1764	1761-1764		tbd	tbd							
43.0 Microsoft Message Queue Service	Well-Known			1801,2101,2103,2105	3527, 1801		tbd	tbd							
44.0 Distributed.net	Well-Known			2064											
45.0 OpenWindows	Well-Known			2000	2000										
46.0 X/Windows (X.11)	Well-Known			6000	6000	fix udp									
47.0 America Online (AOL)	Well-Known			1590-1593											
48.0 talk	Well-Known			517	517										
48.0 ntalk	Well-Known			518	518										
49.0 Internet Relay Chat (IRC)	Well-Known			6665 - 6669											
50.0 iChat	Well-Known			4020, 4080											
51.0 iVisit	Well-Known				9943, 9945, 56768										
52.0 The Palace	Well-Known			9992-9998	9992-9998										
53.0 Network Time Protocol (NTP)	Well-Known			123	123	fix tcp									
54.0 TACACS	Well-Known			49	49	fix udp									
55.0 Netbios SSN over TCP	Well-Known					tbd									
56.0 Netbios NM over UDP	Well-Known					tbd									
57.0 SMB over Netbios Session	Well-Known					tbd									
60.0 Quake/Quake-II	tbd					tbd - fix									
61.0 QuakeWorld	tbd					tbd									

Description	Type	TCP/IP Access					Novell Access					Unix	DceRPC	Misc	
		HTTP Content	HTTP Pattern	Well Known TCP	Well Known UDP	Other	Well Known IPX/SPX	SAP Mapped	Well Known VIP/VSPP	Well Known VIPC	Other	Port Mapped	Endpoint Mapped	New Parents	Other
3.0 RealAudio	State Based	•	•	7070	6970-7170	CORE #1									
4.0 Pointcast	State Based		•			core #1									
5.0 BackWeb	State Based		•		370	CORE #4									
6.0 Microsoft Media (formerly NetShow)	State Based	•		1755		CORE #7									
7.0 QuickTime	State Based	•				core #1									
8.0 VivoActive	State Based	•				core #1									
9.0 Shockwave	State Based	•				core #1									
10.0 PowerBuilder (Sybase/Powersoft)	State Based	•				core #1									
11.0 Web.SQL (Sybase)	State Based		•			core #1									
12.0 jConnect/JDBC (Sybase)	State Based		•			core #1									
18.2 Oracle - Transparent Network Substrate	State Based			1521,1526,1527			tbd	tbd	tbd	tbd					CORE #10
18.3 SQL*Net	State Based													Oracle	core #10
18.3 MS-ODBC	State Based													Oracle	core #10
18.3 PeopleSoft	State Based													tbd	core #10
18.3 SAP	State Based													tbd	core #10
21.0 IP-fragmentation	State Based					CORE #3									
22.0 SunRPC PortMapper	State Based					CORE #2									
23.0 Mount	State Based					core #2						•			
24.0 NFS	State Based				2049	core #2						•			
25.0 Yellow Pages	State Based					core #2						•			
25.0 db Session Manager	State Based					core #2						•			
25.0 pcNFS	State Based					core #2						•			
25.0 3270_mapper	State Based					core #2						•			
25.0 rje_mapper	State Based					core #2						•			
25.0 NIS+ (National Information Service)	State Based					core #2						•			
25.0 rstat	State Based					core #2						•			
26.0 Novell SAP	State Based							CORE #11							
27.2 MS-Exchange – Information Store	State Based												•		Core #8,9
27.2 MS-Exchange – Directory	State Based												•		Core #8,9
27.2 MS-Exchange – MTA	State Based												•		Core #8,9
28.6 DceRPC Endpoint Mapper (conn-less)	State Based				135		tbd	tbd	tbd	tbd					CORE #9
28.7 DceRPC Endpoint Mapper (conn-oriented)	State Based			135			tbd	tbd	tbd	tbd					CORE #8
29.0 Vines IPC-RDP	State Based										CORE #5				
30.0 Vines SMB over SPP	State Based										CORE #6				
31.0 Vines Print	State Based										tbd				
31.0 Vines Async	State Based										tbd				
36.0 Citrix	State Based			1494	1604	??? tcp - tbd	0x85ba-85bb	0x052d, 0x083d							core #11
58.0 Microsoft NetMeeting	State Based					tbd*****									
59.0 X.400	State Based														tbd

CORE #1 - HTTP Engine	TCP port 80, connection-oriented, STATE-BASED core for HTTP
CORE #2 - SunRPC Portmapper Engine	UDP port 111, connection-less, STATE-BASED core for PortMapper
CORE #3 - IP Fragmentation Engine	IP Fragmentation STATE-BASED core mapping fragments #2 - n to their flows from fragments #1
CORE #4 - BackWeb UDP Engine	UDP port 370, connection-less, STATE-BASED core to map responses to requests on port #370
CORE #5 - Vines IPC Engine	VIPC-RDP connection-oriented, STATE-BASED core to track VIPC sessions (for all sessions)
CORE #6 - Vines SPP Engine	VSPP connection-oriented, STATE-BASED core to track VSPP sessions (for non-well-known sockets)
CORE #7 - MS Media Engine	TCP port 1755, connection-oriented, STATE-BASED core to handle dynamic UDP Port Assignment
CORE #8 - DceRPC CO Mapper Engine	connection-oriented, STATE-BASED core for DCE RPC Endpoint Mapping
CORE #9 - DceRPC CL Mapper Engine	connection-less STATE-BASED core for DCE RPC Endpoint Mapping
CORE #10 - Oracle TNS Engine	TCP port 1521/1526/1527, connection oriented, STATE-BASED core for Oracle TNS session tracking
CORE #11 - Novell SAP Engine	connection-less, STATE-BASED core for Service Advertisement Program (SAP) mapping

Description	Type	HTTP Content Type	HTTP Pattern	Well Known TCP	Well Known UDP	Other	Well Known IPX/SPX	SAP Mapped	Well Known VIP/VSPP	Well Known VIPC	Other	Port Mapped	Endpoint Mapped	New Parents	Other
3.0 plain	State Based	text				core #1									
3.0 html	State Based	text				core #1									
3.0 tab-separated-values	State Based	text				core #1									
3.0 sgml	State Based	text				core #1									
3.0 rich-text	State Based	text				core #1									
3.0 enriched	State Based	text				core #1									
3.0 real-audio	State Based	text				core #1									
3.0 gif	State Based	image				core #1									
3.0 jpeg	State Based	image													
3.0 pict	State Based	image													
3.0 x-bitmap	State Based	image													
3.0 x-pixmap	State Based	image													
3.0 cgm	State Based	image													
3.0 group-3-fax	State Based	image													
3.0 png	State Based	image													
3.0 ief	State Based	image													
3.0 tiff	State Based	image													
3.0 real-audio	State Based	image													
3.0 quicktime	State Based	image													
3.0 basic-audio	State Based	audio													
3.0 AIFF	State Based	audio													
3.0 mpeg2-audio	State Based	audio													
3.0 WAV	State Based	audio													
3.0 real-audio	State Based	audio													
3.0 MIDI	State Based	audio													
3.0 x-windows-dump-image	State Based	audio													
3.0 mpeg-audio-3	State Based	audio													
3.0 vrml	State Based	x-world													
3.0 quicktime	State Based	applic													
3.0 mpeg2-video	State Based	applic													
3.0 sgi-video	State Based	applic													
3.0 real-audio	State Based	applic													
3.0 ms-media	State Based	applic													
3.0 avi	State Based	applic													
3.0 vivo-active	State Based	applic													
3.0 mac-binhex40	State Based	applic													
3.0 mac-stuffit	State Based	applic													
3.0 mac-binary	State Based	applic													
3.0 compress	State Based	applic													
3.0 zip	State Based	applic													
3.0 gzip	State Based	applic													
3.0 tar	State Based	applic													
3.0 posix-tar	State Based	applic													
3.0 gnu-tar	State Based	applic													
3.0 cpio	State Based	applic													
3.0 bcpio	State Based	applic													
3.0 c-shell	State Based	applic													
3.0 bourne-shell	State Based	applic													
3.0 tcl	State Based	applic													
3.0 octet-stream	State Based	applic													
3.0 javascript	State Based	applic													
3.0 mpeg-audio-3	State Based	applic													
3.0 rich-text	State Based	applic													
3.0 tex	State Based	applic													
3.0 latex	State Based	applic													
3.0 tex-dvi	State Based	applic													
3.0 gnu-texinfo	State Based	applic													
3.0 oda	State Based	applic													

3.0 edifact	State Based	applic
3.0 edi-x12	State Based	applic
3.0 edi-consent	State Based	applic
3.0 news	State Based	applic
3.0 microsoft-word	State Based	applic
3.0 microsoft-excel	State Based	applic
3.0 microsoft-powerpoint	State Based	applic
3.0 microsoft-project	State Based	applic
3.0 lotus-organizer	State Based	applic
3.0 lotus-freelance	State Based	applic
3.0 lotus-123	State Based	applic
3.0 lotus-approach	State Based	applic
3.0 lotus-wordpro	State Based	applic
3.0 troff	State Based	applic
3.0 wordperfect	State Based	applic
3.0 quattro-pro	State Based	applic
3.0 framemaker	State Based	applic
3.0 postscript	State Based	applic
3.0 visio	State Based	applic
3.0 sgml	State Based	applic
3.0 powerbuilder	State Based	applic
3.0 real-audio	State Based	applic
3.0 shockwave	State Based	applic
3.0 acrobat	State Based	applic
3.0 lotus-notes	State Based	applic

- **Exhibit B0** is a dated computer directory of test data and documents used therefore.

[redacted]Compiler-dir.txt
Directory of M:\aaa-----INVENTEK CLIENTS\Hifn\Patents\APPT-001-1-1 filed [redacted]\Proof of Reductn to Practice\[redacted] Compiler\

M:\aaa-----INVENTEK CLIENTS\Hifn\Patents\APPT-001-1-1 filed [redacted]Proof of Reductn to Practice\[redacted]Compiler\

big.cpl	548 KB	[redacted]	04:17:18 AM	a
bigfgc3.cpl	1164 KB	[redacted]	04:06:18 PM	a
bigfgpc.cpl	1164 KB	[redacted]	04:06:18 PM	a
bigfpayl.cpl	1051 KB	[redacted]	09:57:44 AM	a
bigfpayl2.cpl	1054 KB	[redacted]	10:17:18 AM	a
bigfpgrp.cpl	1159 KB	[redacted]	11:04:40 AM	a
bigfpgrp2.cpl	1163 KB	[redacted]	10:11:06 AM	a
bigfrag.cpl	995 KB	[redacted]	07:17:34 AM	a
bigfrag2.cpl	999 KB	[redacted]	10:21:52 AM	a
mfaptkey.txt	1 KB	[redacted]	03:05:54 PM	a
mfaptkey2.txt	1 KB	[redacted]	07:54:12 AM	a
mfaptpkt.txt	4 KB	[redacted]	03:07:00 PM	a
mfaptpkt2.txt	4 KB	[redacted]	01:52:42 AM	a
MFATEST.HEX	213 KB	[redacted]	02:53:04 PM	a
MFATEST.TXT	70 KB	[redacted]	03:00:48 PM	a
MFS-PDL-Reference.pdf	97 KB	[redacted]	04:10:18 AM	a
MFS-State-Classification.pdf	121 KB	[redacted]	04:11:28 AM	a
output.cpl	209 KB	[redacted]	08:45:34 AM	a
packets.txt	46 KB	[redacted]	09:29:04 AM	a
Protocols.cpl	204 KB	[redacted]	10:12:10 AM	a
short.cpl	150 KB	[redacted]	08:38:42 AM	a
shrtfpg2.cpl	290 KB	[redacted]	10:14:38 AM	a
shrtfps3.cpl	256 KB	[redacted]	02:25:12 PM	a
shrtfps4.cpl	86 KB	[redacted]	10:35:56 AM	a
shrtfps5.cpl	86 KB	[redacted]	10:35:56 AM	a
shrttunl.cpl	171 KB	[redacted]	12:21:42 PM	a

ΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛ
Total 0 folder(s); 26 file(s)

Total files size: 11 MB; 11315 KB; 11586502 Bytes

ΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛΛ

- **Exhibit B1:** Technically Elite MeterFlow Accelerator Modules Testbench Specification (Document MFATest.pdf in directory of Exhibit A0)

Technically Elite MeterFlow Accelerator Modules Testbench Specification

Not For External Release!

Revision History		
Version	**Date**	**Description**
0.1	[redacted]	Collect earlier documents. Format document.
0.9	[redacted]	Technically Elite review release.

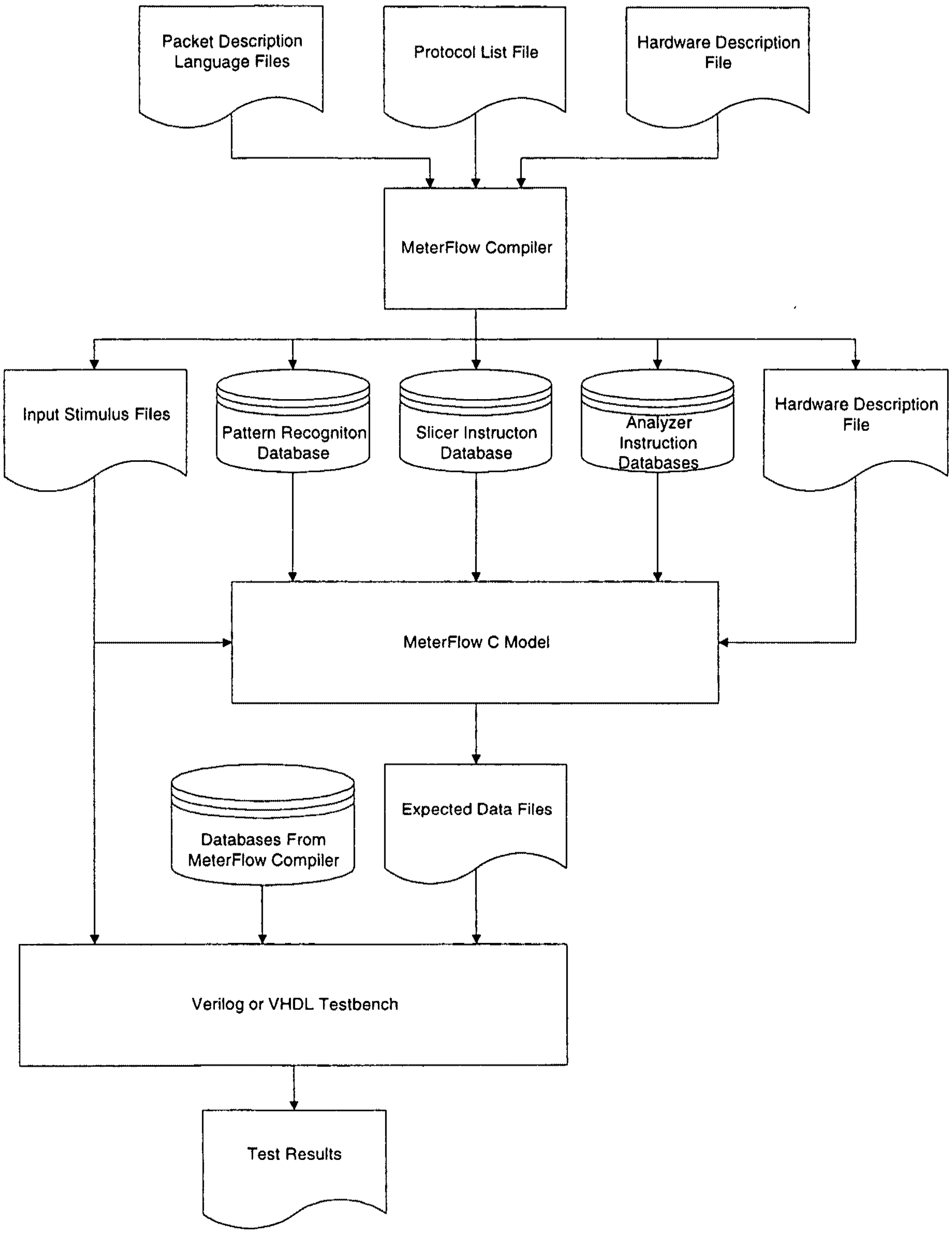
1 Introduction

This document describes the methodology to be used to build testbenches for the MeterFlow Accelerator Modules Verilog and VHDL implementations. The goal is to have fully automated testing. This means that the unit under tests (UUT) output is compared to expected data generated by the C model and the results can be reported as pass/fail. The input to the testbenches are files generated by the MeterFlow Compiler.

1.1 Technically Elite MeterFlow Accelerator Modules Testbench

- Written in both the Verilog and VHDL
- Asynchronous interfaces each have a separate clock
- Automated testing and result reporting
- The same input files read by the testbenches and the C model

2 Test Flow Chart



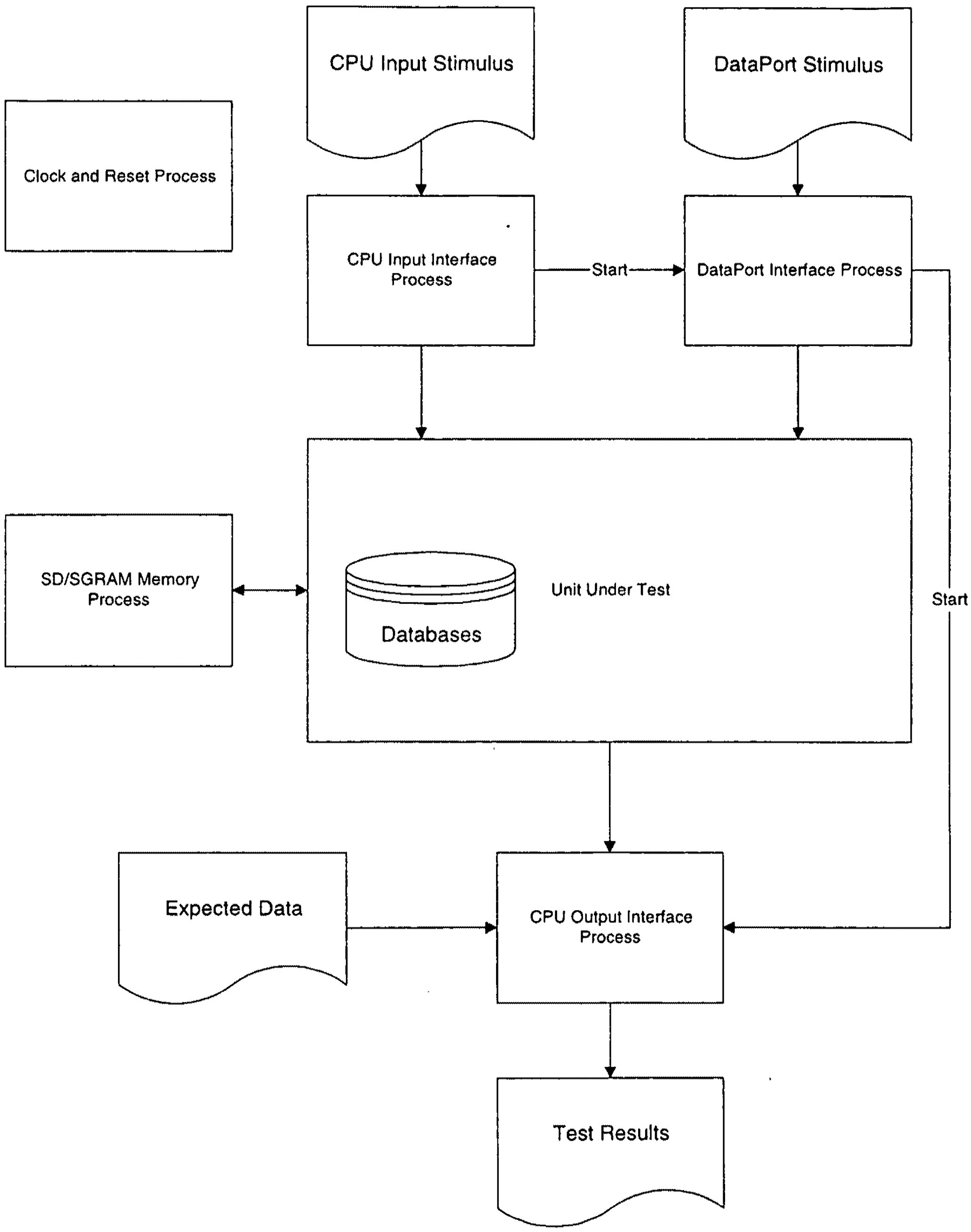
2.1 Test Flow Chart Description

The MeterFlow Compiler takes as it's input three sets of files. The first is the Protocol List file. This file describes the protocols this implementation of the hardware must recognize and process. The compiler will then lookup each of the protocols in the list for their Packet Description Language file. Each of these files describes how to recognize and process the protocol. Finally the compiler may be given a Hardware Description files that specifies the hardware resources available for this implementation. The compiler can also generate this file by determining the minimum resources required to implement all the protocols in the list.

The compiler outputs the databases used by the MeterFlow accelerator. These databases can be read into the C model, the UUT and the actual hardware (if it exists). It also outputs a set of input stimulus files for both the C model and the testbenches.

The C model emulates the functions of the UUT and produces expected data files. These files contain cycle by cycle data that the testbench uses to check the results of the test.

3 Testbench Block Diagram



3.1 Testbench Block Diagram Description

The testbenches are built up of separate processes run concurrently. The Clock and Reset Process generates the system clock and system reset signals. If a process requires a different clock, such as the SD/SGRAM Memory Process, it generates that clock itself.

The SD/SGRAM Memory Process instantiates the target memory the system is to use. An accurate model of the memory is required to assure valid results.

The three processes that are shown importing files, each instantiate memories to hold the data read from the files. These memories act as patterns to be either driven into the UUT or patterns the output of the UUT are compared against. Since the UUT must be programmed before testing can begin, there is a handshake between each of the three processes. This is shown in the diagram as the Start signals.

The test begins with the CPU Input Interface Process programming the UUT. Once the UUT is programmed, the CPU Input Interface Process raises it's Start output. This tells the DataPort Interface Process to begin sending packets into the UUT. After the packets are completed the DataPort Interface Process raises it's Start output. The CPU Output Interface Process then begins reading the flow database. It checks the flows against the expected data and writes the Test Results file.

- **Exhibit B2:** The first page of file big.cpl.

  The cpl files (big.cpl, bigfgc3.cpl, bigfgpc.cpl, bigfpayl.cpl, bigfpayl2.cpl, bigfpgrp.cpl, bigfpgrp2.cpl, bigfrag.cpl, bigfrag2.cpl, output.cpl, Protocols.cpl, short.cpl, shrtfpg2.cpl, shrtfps3.cpl, shrtfps4.cpl, shrtfps5.cpl, shrttunl.cpl) are files for the protocol compiler of all the actual protocols recognized by the system. These files include a description of the parser information for the parser to perform the parsing/extracting operation according to the protocol. They also contain the state processing states for the state operations of elements (d) and (e) of claim 54. The first page of one file is provided.

```
-- Generated on [redacted] 22:04:05

0x017C -- Total number of protocols (380 dec)

-- ****************************************************************************
--
-- Virtual Layer Decodes
--
-- ****************************************************************************

VirtualBase -- Text Name
0x00       -- InternalProtocolCode
0x01       -- HeaderLengthFixed (0x00 - no [computed], 0x01 - yes [fixed])
0x00       -- HeaderLengthElementSize (0x00 - byte, 0x01 nibble)
0x00       -- HeaderLengthWord (0x00 - byte count, 0x01 word count (32 bits))
0x01       -- HeaderLengthField (byte offset or nibble offset)
0x00       -- DLCLayerFlag ( NO )
0x00       -- DLCLayerDestOffset ( NULL )
0x00       -- DLCLayerDestMask ( NULL )
0x00       -- DLCLayerSrcOffset ( NULL )
0x00       -- DLCLayerSrcMask ( NULL )
0x00       -- NetLayerFlag ( NO )
0x00       -- NetLayerAddressSize ( NULL )
0x00       -- NetLayerDestOffset ( NULL )
0x00       -- NetLayerDestMask ( NULL )
0x00       -- NetLayerSrcOffset ( NULL )
0x00       -- NetLayerSrcMask ( NULL )
0x00       -- NetLayerFragments ( NULL )
0x00       -- TunnelLayerFlag ( NO )
0x00       -- TunnelLayerAddressSize ( NULL )
0x00       -- TunnelLayerDestOffset ( NULL )
0x00       -- TunnelLayerDestMask ( NULL )
0x00       -- TunnelLayerSrcOffset ( NULL )
0x00       -- TunnelLayerSrcMask ( NULL )
0x00       -- TunnelLayerFragments ( NULL )
0x00       -- ConnectionLayerFlag
0x00       -- ChildRecognitionTypeLengthFlag
0x01       -- ChildRecognitionIgnoreSource (0x00 - no, 0x01 - yes [ignore] )
0x01       -- ChildRecognitionSize
0x00       -- ChildRecognitionDestOffset
0x00       -- ChildRecognitionSrcOffset
0x01       -- NumChildren (1 children)
0x01      -- RecognitionCode
0x01      -- Ethernet Base

-- ****************************************************************************
--
-- DLC Layer Decodes
--
-- ****************************************************************************

-- DLC (base) Ethernet V2 Decodes

EtherType -- Text Name
0x01       -- InternalProtocolCode
0x01       -- HeaderLengthFixed (0x00 - no [computed], 0x01 - yes [fixed])
0x00       -- HeaderLengthElementSize (0x00 - byte, 0x01 nibble)
0x00       -- HeaderLengthWord (0x00 - byte count, 0x01 word count (32 bits))
0x0E       -- HeaderLengthField (byte offset or nibble offset)
0x01       -- DLCLayerFlag ( YES )
0x00       -- DLCLayerDestOffset ( 0 - 5 )
0xFF       -- DLCLayerDestMask ( All bits )
0x06       -- DLCLayerSrcOffset ( 6 - 11 )
```

- **Exhibit B3:** The file MFATEST.HEX that contains the actual packets captured by the packet acquisition device described in element (a) of claims 11 and 54, and corresponding to the contents of element (b), the input buffer memory of claim 29. The packet acquisition device for the experiment was a SUN workstation connected to a connection point of a network.

```
42
08 00 20 13 10 D2 00 A0 24 75 C7 78 08 00 45 00
00 30 B0 6C 40 00 80 06 94 14 59 06 06 03 59 07
FE 36 09 53 00 6E 1A 5D 8A 6E 50 DA 49 60 50 18
1D 4B BF 97 00 00 52 45 54 52 20 32 0D 0A FD 6E
9D F5
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
00 00 00 00 08 00 20 13 10 D2 00 A0 24 75 C7 78
00 08 59 07 FE 36 00 00 00 00 00 00 00 00 00 00
00 00 59 06 06 03 00 00 00 00 00 00 00 00 00 00
00 00 00 2B 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 47 00 6E 00 00 09 53 00 00
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
4B
00 A0 24 75 C7 78 08 00 20 13 10 D2 08 00 45 00
00 39 16 03 00 00 3C 06 B2 75 59 07 FE 36 59 06
06 03 00 6E 09 53 50 DA 49 60 1A 5D 8A 76 50 18
10 00 5D 6C 00 00 2B 4F 4B 20 31 33 35 31 20 6F
63 74 65 74 73 0D 0A CA E0 6A B1
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
00 00 00 00 00 A0 24 75 C7 78 08 00 20 13 10 D2
00 08 59 06 06 03 00 00 00 00 00 00 00 00 00 00
00 00 59 07 FE 36 00 00 00 00 00 00 00 00 00 00
00 00 00 2B 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 47 09 53 00 00 00 6E 00 00
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
40
08 00 20 13 10 D2 00 A0 24 75 C7 78 08 00 45 00
00 28 B1 6C 40 00 80 06 93 1C 59 06 06 03 59 07
FE 36 09 53 00 6E 1A 5D 8A 76 50 DA 49 71 50 10
1D 3A 93 73 00 00 00 00 00 00 00 00 02 03 21 C2
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
00 00 00 00 08 00 20 13 10 D2 00 A0 24 75 C7 78
00 08 59 07 FE 36 00 00 00 00 00 00 00 00 00 00
00 00 59 06 06 03 00 00 00 00 00 00 00 00 00 00
00 00 00 2B 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 47 00 6E 00 00 09 53 00 00
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      4      TO: 00A02475C778              [redacted]22:53:51<0.002> [redacted]
             FROM: 0800201310D2

Pkt: 4, Len: 1120/1390
0000    00 A0 24 75 C7 78 08 00   20 13 10 D2 08 00 45 00    ..$u.x.. .....E.
0010    05 5C 16 05 00 00 3C 06   AD 50 59 07 FE 36 59 06    .\....<..PY..6Y.
0020    06 03 00 6E 09 53 50 DA   49 71 1A 5D 8A 76 50 18    ...n.SP.Iq.].vP.
0030    10 00 53 11 00 00 52 65   74 75 72 6E 2D 50 61 74    ..S...Return-Pat
0040    68 3A 20 3C 6A 6D 65 74   7A 67 65 72 40 74 65 63    h: <jmetzger@tec
0050    65 6C 69 74 65 2E 63 6F   6D 3E 0D 0A 52 65 63 65    elite.com>..Rece
0060    69 76 65 64 3A 20 66 72   6F 6D 20 6E 61 74 61 64    ived: from natad
0070    6D 2E 74 65 63 65 6C 69   74 65 2E 63 6F 6D 20 62    m.tecelite.com b
0080    79 20 73 75 70 65 72 2E   74 65 63 65 6C 69 74 65    y super.tecelite
0090    2E 63 6F 6D 20 28 34 2E   31 2F 53 4D 49 2D 34 2E    .com (4.1/SMI-4.
00A0    31 29 0D 0A 09 69 64 20   41 41 32 38 34 30 38 3B    1)...id AA28408;
00B0    20 54 68 75 2C 20 31 30   20 53 65 70 20 39 38 20     Thu, 10 Sep 98
00C0    31 37 3A 33 37 3A 33 37   20 50 44 54 0D 0A 52 65    17:37:37 PDT..Re
00D0    63 65 69 76 65 64 3A 20   66 72 6F 6D 20 73 6D 74    ceived: from smt
00E0    70 6C 69 6E 6B 2E 74 65   63 65 6C 69 74 65 2E 63    plink.tecelite.c
00F0    6F 6D 20 28 73 6D 74 70   6C 69 6E 6B 20 5B 38 39    om (smtplink [89
0100    2E 37 2E 37 2E 31 30 30   5D 29 0D 0A 09 62 79 20    .7.7.100])...by
```

```
0110   6E 61 74 61 64 6D 2E 74   65 63 65 6C 69 74 65 2E   natadm.tecelite.
0120   63 6F 6D 20 28 38 2E 38   2E 37 2F 38 2E 38 2E 37   com (8.8.7/8.8.7
0130   29 20 77 69 74 68 20 53   4D 54 50 20 69 64 20 52   ) with SMTP id R
0140   41 41 31 37 32 34 35 3B   0D 0A 09 54 68 75 2C 20   AA17245;...Thu,
0150   31 30 20 53 65 70 20 31   39 39 38 20 31 37 3A 33   10 Sep 1998 17:3
0160   39 3A 30 34 20 2D 30 37   30 30 0D 0A 52 65 63 65   9:04 -0700..Rece
0170   69 76 65 64 3A 20 66 72   6F 6D 20 63 63 3A 4D 61   ived: from cc:Ma
0180   69 6C 20 62 79 20 73 6D   74 70 6C 69 6E 6B 2E 74   il by smtplink.t
0190   65 63 65 6C 69 74 65 2E   63 6F 6D 0D 0A 09 69 64   ecelite.com...id
01A0   20 41 41 39 30 35 34 37   34 38 32 39 20 54 68 75    AA905474829 Thu
01B0   2C 20 31 30 20 53 65 70   20 39 38 20 31 37 3A 34   , 10 Sep 98 17:4
01C0   37 3A 30 39 20 50 44 54   0D 0A 44 61 74 65 3A 20   7:09 PDT..Date:
01D0   54 68 75 2C 20 31 30 20   53 65 70 20 39 38 20 31   Thu, 10 Sep 98 1
01E0   37 3A 34 37 3A 30 39 20   50 44 54 0D 0A 46 72 6F   7:47:09 PDT..Fro
01F0   6D 3A 20 4A 6F 68 6E 20   4D 65 74 7A 67 65 72 20   m: John Metzger
0200   3C 6A 6D 65 74 7A 67 65   72 40 74 65 63 65 6C 69   <jmetzger@teceli
0210   74 65 2E 63 6F 6D 3E 0D   0A 45 6E 63 6F 64 69 6E   te.com>..Encodin
0220   67 3A 20 33 32 34 20 54   65 78 74 0D 0A 4D 65 73   g: 324 Text..Mes
0230   73 61 67 65 2D 49 64 3A   20 3C 39 38 30 38 31 30   sage-Id: <980810
0240   39 30 35 34 2E 41 41 39   30 35 34 37 34 38 32 39   9054.AA905474829
0250   40 73 6D 74 70 6C 69 6E   6B 2E 74 65 63 65 6C 69   @smtplink.teceli
0260   74 65 2E 63 6F 6D 3E 0D   0A 54 6F 3A 20 62 6C 65   te.com>..To: ble
0270   61 76 79 40 74 65 63 65   6C 69 74 65 2E 63 6F 6D   avy@tecelite.com
0280   2C 20 61 63 68 61 64 64   61 40 74 65 63 65 6C 69   , achadda@teceli
0290   74 65 2E 63 6F 6D 2C 20   64 61 76 65 63 40 74 65   te.com, davec@te
02A0   63 65 6C 69 74 65 2E 63   6F 6D 2C 0D 0A 20 20 20   celite.com,..
02B0   20 20 20 20 20 44 61 76   69 64 20 4C 75 6F 20 3C        David Luo <
02C0   64 6C 75 6F 40 74 65 63   65 6C 69 74 65 2E 63 6F   dluo@tecelite.co
02D0   6D 3E 2C 20 6C 6F 77 64   65 72 40 74 65 63 65 6C   m>, lowder@tecel
02E0   69 74 65 2E 63 6F 6D 2C   0D 0A 20 20 20 20 20 20   ite.com,..
02F0   20 20 65 77 68 65 65 6C   65 72 40 74 65 63 65 6C     ewheeler@tecel
0300   69 74 65 2E 63 6F 6D 2C   20 66 6E 6F 6F 6E 40 74   ite.com, fnoon@t
0310   65 63 65 6C 69 74 65 2E   63 6F 6D 2C 20 66 72 65   ecelite.com, fre
0320   64 6D 40 74 65 63 65 6C   69 74 65 2E 63 6F 6D 2C   dm@tecelite.com,
0330   0D 0A 20 20 20 20 20 20   20 20 6A 6D 61 69 78 6E   ..        jmaixn
0340   65 72 40 74 65 63 65 6C   69 74 65 2E 63 6F 6D 2C   er@tecelite.com,
0350   20 6A 6F 74 69 73 40 74   65 63 65 6C 69 74 65 2E    jotis@tecelite.
0360   63 6F 6D 2C 0D 0A 20 20   20 20 20 20 20 20 4B 69   com,..        Ki
0370   6D 20 44 61 76 69 73 20   3C 6B 64 61 76 69 73 40   m Davis <kdavis@
0380   74 65 63 65 6C 69 74 65   2E 63 6F 6D 3E 2C 20 72   tecelite.com>, r
0390   61 6D 40 74 65 63 65 6C   69 74 65 2E 63 6F 6D 2C   am@tecelite.com,
03A0   0D 0A 20 20 20 20 20 20   20 20 52 6F 62 20 52 69   ..        Rob Ri
03B0   74 7A 20 3C 72 72 69 74   7A 40 74 65 63 65 6C 69   tz <rritz@teceli
03C0   74 65 2E 63 6F 6D 3E 2C   20 72 73 64 69 65 74 7A   te.com>, rsdietz
03D0   40 74 65 63 65 6C 69 74   65 2E 63 6F 6D 2C 20 73   @tecelite.com, s
03E0   6B 69 70 40 74 65 63 65   6C 69 74 65 2E 63 6F 6D   kip@tecelite.com
03F0   0D 0A 53 75 62 6A 65 63   74 3A 20 4E 65 78 74 20   ..Subject: Next
0400   47 65 6E 65 72 61 74 69   6F 6E 20 50 72 6F 64 75   Generation Produ
0410   63 74 20 44 69 73 63 75   73 73 69 6F 6E 0D 0A 0D   ct Discussion...
0420   0A 0D 0A 53 75 62 6A 65   63 74 3A 20 4E 65 78 74   ...Subject: Next
0430   20 47 65 6E 65 72 61 74   69 6F 6E 20 50 72 6F 64    Generation Prod
0440   75 63 74 20 44 69 73 63   75 73 73 69 6F 6E 0D 0A   uct Discussion..
0450   0D 0A 49 20 77 6F 75 6C   64 20 6C 69 6B 65 20 74   ..I would like t
```

-

5 TO: 0800201310D2 [redacted]2:53:51<0.198>[redacted]
FROM: 00A02475C778

```
Pkt: 5, Len: 64/64
0000   08 00 20 13 10 D2 00 A0   24 75 C7 78 08 00 45 00   .. .....$u.x..E.
0010   00 28 B2 6C 40 00 80 06   92 1C 59 06 06 03 59 07   .(.l@.....Y...Y.
```

```
0020   FE 36 09 53 00 6E 1A 5D   8A 76 50 DA 4E A5 50 10   .6.S.n.].vP.N.P.
0030   22 38 89 41 00 00 00 00   00 00 00 00 BA 3C 6B D6   "8.A.........<k.

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      6       TO: 0800201310D2                    [redacted]2:53:53<2.556>[redacted]
              FROM: 00A02475C778

Pkt: 6, Len: 66/66
0000   08 00 20 13 10 D2 00 A0   24 75 C7 78 08 00 45 00   .. .....$u.x..E.
0010   00 30 B3 6C 40 00 80 06   91 14 59 06 06 03 59 07   .0.1@.....Y...Y.
0020   FE 36 09 53 00 6E 1A 5D   8A 76 50 DA 4E A5 50 18   .6.S.n.].vP.N.P.
0030   22 38 CB 6A 00 00 44 45   4C 45 20 32 0D 0A BC F6   "8.j..DELE 2....
0040   77 D9                                               w.

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      7       TO: 00A02475C778                    [redacted]2:53:53<0.001> [redacted]
              FROM: 0800201310D2

Pkt: 7, Len: 91/91
0000   00 A0 24 75 C7 78 08 00   20 13 10 D2 08 00 45 00   ..$u.x.. .....E.
0010   00 49 16 09 00 00 3C 06   B2 5F 59 07 FE 36 59 06   .I....<.._Y..6Y.
0020   06 03 00 6E 09 53 50 DA   4E A5 1A 5D 8A 7E 50 18   ...n.SP.N..].~P.
0030   10 00 3F 4C 00 00 2B 4F   4B 20 4D 65 73 73 61 67   ..?L..+OK Messag
0040   65 20 32 20 68 61 73 20   62 65 65 6E 20 64 65 6C   e 2 has been del
0050   65 74 65 64 2E 0D 0A 52   E8 E2 05                  eted...R...

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      8       TO: 0800201310D2                    [redacted]22:53:53<0.002> [redacted]
              FROM: 00A02475C778

Pkt: 8, Len: 64/64
0000   08 00 20 13 10 D2 00 A0   24 75 C7 78 08 00 45 00   .. .....$u.x..E.
0010   00 2E B4 6C 40 00 80 06   90 16 59 06 06 03 59 07   ...1@.....Y...Y.
0020   FE 36 09 53 00 6E 1A 5D   8A 7E 50 DA 4E C6 50 18   .6.S.n.].~P.N.P.
0030   22 17 E1 77 00 00 51 55   49 54 0D 0A 66 C7 F0 F5   "..w..QUIT..f...

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      9       TO: 00A02475C778                    [redacted]:53:53<0.029> [redacted]
              FROM: 0800201310D2

Pkt: 9, Len: 96/96
0000   00 A0 24 75 C7 78 08 00   20 13 10 D2 08 00 45 00   ..$u.x.. .....E.
0010   00 4E 16 0A 00 00 3C 06   B2 59 59 07 FE 36 59 06   .N....<..YY..6Y.
0020   06 03 00 6E 09 53 50 DA   4E C6 1A 5D 8A 84 50 18   ...n.SP.N..]..P.
0030   10 00 A0 4C 00 00 2B 4F   4B 20 50 6F 70 20 73 65   ...L..+OK Pop se
0040   72 76 65 72 20 61 74 20   73 75 70 65 72 20 73 69   rver at super si
0050   67 6E 69 6E 67 20 6F 66   66 2E 0D 0A 0D 7A D8 45   gning off....z.E

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      10      TO: 00A02475C778                    [redacted]22:53:53<0.003>[redacted]
              FROM: 0800201310D2

Pkt: 10, Len: 64/64
```

```
0000    00 A0 24 75 C7 78 08 00   20 13 10 D2 08 00 45 00    ..$u.x.. .....E.
0010    00 28 16 0B 00 00 3C 06   B2 7E 59 07 FE 36 59 06    .(....<..~Y..6Y.
0020    06 03 00 6E 09 53 50 DA   4E EC 1A 5D 8A 84 50 11    ...n.SP.N..]..P.
0030    10 00 9B 23 00 00 00 00   00 00 00 00 2B A5 6E 6A    ...#........+.nj

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      11      TO: 0800201310D2                  [redacted]:53:53<0.000> [redacted]
              FROM: 00A02475C778

Pkt: 11, Len: 64/64
0000    08 00 20 13 10 D2 00 A0   24 75 C7 78 08 00 45 00    .. .....$u.x..E.
0010    00 28 B5 6C 40 00 80 06   8F 1C 59 06 06 03 59 07    .(.l@.....Y...Y.
0020    FE 36 09 53 00 6E 1A 5D   8A 84 50 DA 4E ED 50 10    .6.S.n.]..P.N.P.
0030    21 F1 89 32 00 00 00 00   00 00 00 00 BA 06 A9 CC    !..2............

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      12      TO: 0800201310D2                  [redacted]2:53:53<0.031> [redacted]
              FROM: 00A02475C778

Pkt: 12, Len: 64/64
0000    08 00 20 13 10 D2 00 A0   24 75 C7 78 08 00 45 00    .. .....$u.x..E.
0010    00 28 B6 6C 40 00 80 06   8E 1C 59 06 06 03 59 07    .(.l@.....Y...Y.
0020    FE 36 09 53 00 6E 1A 5D   8A 84 50 DA 4E ED 50 11    .6.S.n.]..P.N.P.
0030    21 F1 89 31 00 00 00 00   00 00 00 00 1C BC F9 85    !..1............

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      13      TO: 00A02475C778                  [redacted]22:53:53<0.001> [redacted]
              FROM: 0800201310D2

Pkt: 13, Len: 64/64
0000    00 A0 24 75 C7 78 08 00   20 13 10 D2 08 00 45 00    ..$u.x.. .....E.
0010    00 28 16 0C 00 00 3C 06   B2 7D 59 07 FE 36 59 06    .(....<..}Y..6Y.
0020    06 03 00 6E 09 53 50 DA   4E ED 1A 5D 8A 85 50 10    ...n.SP.N..]..P.
0030    10 00 9B 22 00 00 00 00   00 00 00 00 E0 F4 BC B0    ..."............

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      14      TO: 006008C0D710                  [redacted]2:53:58<4.755> [redacted]
              FROM: 00A076A010F2

Pkt: 14, Len: 64/64
0000    00 60 08 C0 D7 10 00 A0   76 A0 10 F2 08 00 45 00    .`......v.....E.
0010    00 2C 5F FE 40 00 80 06   B9 0B 59 59 18 06 59 4B    .,_.@.....YY..YK
0020    17 18 05 B4 00 8B 01 BE   3A 7E 00 00 00 00 60 02    ........:~....`.
0030    20 00 53 E9 00 00 02 04   05 B4 20 00 D9 FB 20 4D     .S....... ... M

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      15      TO: 00A076A010F2                  [redacted]2:53:58<0.001> [redacted]
              FROM: 006008C0D710

Pkt: 15, Len: 64/64
0000    00 A0 76 A0 10 F2 00 60   08 C0 D7 10 08 00 45 00    ..v....`......E.
0010    00 2C 49 7C 40 00 80 06   CF 8D 59 4B 17 18 59 59    .,I|@.....YK..YY
0020    18 06 00 8B 05 B4 1E CD   51 3B 01 BE 3A 7F 60 12    ........Q;..:.`.
```

- **Exhibit B4**: The file packets.txt that describes the nature of the packets in MFATEST.HEX.

```
***** Packet ID: 1 *****
ETHERNET=====================================================
Destination Address: 080020-1310d2 (super)
Source Address: 006097-9d6b1d (embedded-pc)
Ethernet Type: 08-00 (IP)
IP===========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 44 (0x2c)
Identification: 56918 (0xde56)
Reserved: 0
Don't Fragment (DF): Don't Fragment(1)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 32 (0x20)
Protocol: TCP(6)
Header Checksum: 7B B5
Source IP: 89.76.80.54 (embedded-pc)
Destination IP: 89.7.254.54 (super)
TCP==========================================================
Source Port: (1427)
Destination Port: POP3(110)
Sequence Number: 16058242 (0xf50782)
Acknowledgement Number: 0
Data Offset: 6 (0x6)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 0
Push Function (PSH): 0
Reset Connection (RST): 0
Synchronize Sequence (SYN): 1
No More Data (FIN): 0
Window Size: 8192 (0x2000)
Checksum: 68 EE
Urgent Pointer: 0


***** Packet ID: 2 *****
ETHERNET=====================================================
Destination Address: 006097-9d6b1d (embedded-pc)
Source Address: 080020-1310d2 (super)
Ethernet Type: 08-00 (IP)
IP===========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 44 (0x2c)
Identification: 1630 (0x65e)
Reserved: 0
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
```

```
Time to Live (TTL): 60 (0x3c)
Protocol: TCP(6)
Header Checksum: 77 AE
Source IP: 89.7.254.54 (super)
Destination IP: 89.76.80.54 (embedded-pc)
TCP=========================================================
Source Port: POP3(110)
Destination Port: (1427)
Sequence Number: 1240192000 (0x49ebd400)
Acknowledgement Number: 16058243 (0xf50783)
Data Offset: 6 (0x6)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 0
Reset Connection (RST): 0
Synchronize Sequence (SYN): 1
No More Data (FIN): 0
Window Size: 4096 (0x1000)
Checksum: 5A F1
Urgent Pointer: 0


***** Packet ID: 3 *****
ETHERNET====================================================
Destination Address: 080020-1310d2 (super)
Source Address: 006097-9d6b1d (embedded-pc)
Ethernet Type: 08-00 (IP)
IP==========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 40 (0x28)
Identification: 57174 (0xdf56)
Reserved: 0
Don't Fragment (DF): Don't Fragment(1)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 32 (0x20)
Protocol: TCP(6)
Header Checksum: 7A B9
Source IP: 89.76.80.54 (embedded-pc)
Destination IP: 89.7.254.54 (super)
TCP=========================================================
Source Port: (1427)
Destination Port: POP3(110)
Sequence Number: 16058243 (0xf50783)
Acknowledgement Number: 1240192001 (0x49ebd401)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 0
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 8760 (0x2238)
Checksum: 60 76
Urgent Pointer: 0
```

```
***** Packet ID: 4 *****
ETHERNET=================================================
Destination Address: 006097-9d6b1d (embedded-pc)
Source Address: 080020-1310d2 (super)
Ethernet Type: 08-00 (IP)
IP=======================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 120 (0x78)
Identification: 1649 (0x671)
Reserved: 0
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 60 (0x3c)
Protocol: TCP(6)
Header Checksum: 77 4F
Source IP: 89.7.254.54 (super)
Destination IP: 89.76.80.54 (embedded-pc)
TCP======================================================
Source Port: POP3(110)
Destination Port: (1427)
Sequence Number: 1240192001 (0x49ebd401)
Acknowledgement Number: 16058243 (0xf50783)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 4096 (0x1000)
Checksum: BA 88
Urgent Pointer: 0
DATA=====================================================
Data:
0000 -- 2B 4F 4B 20 51 55 41 4C 43 4F  +OK QUALCO
0010 -- 4D 4D 20 50 6F 70 20 73 65 72  MM Pop ser
0020 -- 76 65 72 20 64 65 72 69 76 65  ver derive
0030 -- 64 20 66 72 6F 6D 20 55 43 42  d from UCB
0040 -- 20 28 76 65 72 73 69 6F 6E 20   (version
0050 -- 32 2E 31 2E 34 2D 52 33 29 20  2.1.4-R3)
0060 -- 61 74 20 73 75 70 65 72 20 73  at super s
0070 -- 74 61 72 74 69 6E 67 2E 0D 0A  tarting...

***** Packet ID: 5 *****
ETHERNET=================================================
Destination Address: 080020-1310d2 (super)
Source Address: 006097-9d6b1d (embedded-pc)
Ethernet Type: 08-00 (IP)
IP=======================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
```

```
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 55 (0x37)
Identification: 57430 (0xe056)
Reserved: 0
Don't Fragment (DF): Don't Fragment(1)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 32 (0x20)
Protocol: TCP(6)
Header Checksum: 79 AA
Source IP: 89.76.80.54 (embedded-pc)
Destination IP: 89.7.254.54 (super)
TCP=====================================================
Source Port: (1427)
Destination Port: POP3(110)
Sequence Number: 16058243 (0xf50783)
Acknowledgement Number: 1240192081 (0x49ebd451)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 8680 (0x21e8)
Checksum: E4 02
Urgent Pointer: 0
DATA====================================================
Data:
0000 -- 55 53 45 52 20 6A 6D 61 69 78  USER jmaix
0010 -- 6E 65 72 0D 0A        ner..

***** Packet ID: 6 *****
ETHERNET================================================
Destination Address: 006097-9d6b1d (embedded-pc)
Source Address: 080020-1310d2 (super)
Ethernet Type: 08-00 (IP)
IP======================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 77 (0x4d)
Identification: 1650 (0x672)
Reserved: 0
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 60 (0x3c)
Protocol: TCP(6)
Header Checksum: 77 79
Source IP: 89.7.254.54 (super)
Destination IP: 89.76.80.54 (embedded-pc)
TCP=====================================================
Source Port: POP3(110)
```

```
Destination Port: (1427)
Sequence Number: 1240192081 (0x49ebd451)
Acknowledgement Number: 16058258 (0xf50792)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 4096 (0x1000)
Checksum: C1 E2
Urgent Pointer: 0
DATA=====================================================
Data:
0000 -- 2B 4F 4B 20 50 61 73 73 77 6F  +OK Passwo
0010 -- 72 64 20 72 65 71 75 69 72 65  rd require
0020 -- 64 20 66 6F 72 20 6A 6D 61 69  d for jmai
0030 -- 78 6E 65 72 2E 0D 0A     xner...


***** Packet ID: 7 *****
ETHERNET=================================================
Destination Address: 080020-1310d2 (super)
Source Address: 006097-9d6b1d (embedded-pc)
Ethernet Type: 08-00 (IP)
IP=======================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 55 (0x37)
Identification: 57686 (0xe156)
Reserved: 0
Don't Fragment (DF): Don't Fragment(1)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 32 (0x20)
Protocol: TCP(6)
Header Checksum: 78 AA
Source IP: 89.76.80.54 (embedded-pc)
Destination IP: 89.7.254.54 (super)
TCP======================================================
Source Port: (1427)
Destination Port: POP3(110)
Sequence Number: 16058258 (0xf50792)
Acknowledgement Number: 1240192118 (0x49ebd476)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 8643 (0x21c3)
Checksum: DB 04
Urgent Pointer: 0
DATA=====================================================
```

```
Data:
0000 -- 50 41 53 53 20 6A 6D 61 69 78  PASS jmaix
0010 -- 6E 65 72 0D 0A       ner..


***** Packet ID: 8 *****
ETHERNET=====================================================
Destination Address: 006097-9d6b1d (embedded-pc)
Source Address: 080020-1310d2 (super)
Ethernet Type: 08-00 (IP)
IP===========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 40 (0x28)
Identification: 1651 (0x673)
Reserved: 0
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 60 (0x3c)
Protocol: TCP(6)
Header Checksum: 77 9D
Source IP: 89.7.254.54 (super)
Destination IP: 89.76.80.54 (embedded-pc)
TCP==========================================================
Source Port: POP3(110)
Destination Port: (1427)
Sequence Number: 1240192118 (0x49ebd476)
Acknowledgement Number: 16058273 (0xf507a1)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 0
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 4096 (0x1000)
Checksum: 72 1B
Urgent Pointer: 0


***** Packet ID: 9 *****
ETHERNET=====================================================
Destination Address: 006097-9d6b1d (embedded-pc)
Source Address: 080020-1310d2 (super)
Ethernet Type: 08-00 (IP)
IP===========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 83 (0x53)
Identification: 1654 (0x676)
Reserved: 0
```

```
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 60 (0x3c)
Protocol: TCP(6)
Header Checksum: 77 6F
Source IP: 89.7.254.54 (super)
Destination IP: 89.76.80.54 (embedded-pc)
TCP==========================================================
Source Port: POP3(110)
Destination Port: (1427)
Sequence Number: 1240192118 (0x49ebd476)
Acknowledgement Number: 16058273 (0xf507a1)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 4096 (0x1000)
Checksum: 40 BC
Urgent Pointer: 0
DATA=========================================================
Data:
0000 -- 2B 4F 4B 20 6A 6D 61 69 78 6E  +OK jmaixn
0010 -- 65 72 20 68 61 73 20 30 20 6D  er has 0 m
0020 -- 65 73 73 61 67 65 28 73 29 20  essage(s)
0030 -- 28 30 20 6F 63 74 65 74 73 29  (0 octets)
0040 -- 2E 0D 0A           ...


***** Packet ID: 10 *****
ETHERNET=====================================================
Destination Address: 080020-1310d2 (super)
Source Address: 006097-9d6b1d (embedded-pc)
Ethernet Type: 08-00 (IP)
IP===========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 46 (0x2e)
Identification: 57942 (0xe256)
Reserved: 0
Don't Fragment (DF): Don't Fragment(1)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 32 (0x20)
Protocol: TCP(6)
Header Checksum: 77 B3
Source IP: 89.76.80.54 (embedded-pc)
Destination IP: 89.7.254.54 (super)
TCP==========================================================
Source Port: (1427)
Destination Port: POP3(110)
Sequence Number: 16058273 (0xf507a1)
Acknowledgement Number: 1240192161 (0x49ebd4a1)
Data Offset: 5 (0x5)
```

```
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 8600 (0x2198)
Checksum: BE 97
Urgent Pointer: 0
DATA=====================================================
Data: 53 54 41 54 0D 0A     STAT..

***** Packet ID: 11 *****
ETHERNET=================================================
Destination Address: 006097-9d6b1d (embedded-pc)
Source Address: 080020-1310d2 (super)
Ethernet Type: 08-00 (IP)
IP=======================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 49 (0x31)
Identification: 1655 (0x677)
Reserved: 0
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 60 (0x3c)
Protocol: TCP(6)
Header Checksum: 77 90
Source IP: 89.7.254.54 (super)
Destination IP: 89.76.80.54 (embedded-pc)
TCP======================================================
Source Port: POP3(110)
Destination Port: (1427)
Sequence Number: 1240192161 (0x49ebd4a1)
Acknowledgement Number: 16058279 (0xf507a7)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 4096 (0x1000)
Checksum: 91 3C
Urgent Pointer: 0
DATA=====================================================
Data: 2B 4F 4B 20 30 20 30 0D 0A  +OK 0 0..

***** Packet ID: 12 *****
ETHERNET=================================================
Destination Address: 080020-1310d2 (super)
Source Address: 006097-9d6b1d (embedded-pc)
Ethernet Type: 08-00 (IP)
```

```
IP=========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 46 (0x2e)
Identification: 58198 (0xe356)
Reserved: 0
Don't Fragment (DF): Don't Fragment(1)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 32 (0x20)
Protocol: TCP(6)
Header Checksum: 76 B3
Source IP: 89.76.80.54 (embedded-pc)
Destination IP: 89.7.254.54 (super)
TCP========================================================
Source Port: (1427)
Destination Port: POP3(110)
Sequence Number: 16058279 (0xf507a7)
Acknowledgement Number: 1240192170 (0x49ebd4aa)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 8591 (0x218f)
Checksum: B8 90
Urgent Pointer: 0
DATA=======================================================
Data: 51 55 49 54 0D 0A     QUIT..


***** Packet ID: 13 *****
ETHERNET===================================================
Destination Address: 006097-9d6b1d (embedded-pc)
Source Address: 080020-1310d2 (super)
Ethernet Type: 08-00 (IP)
IP=========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 78 (0x4e)
Identification: 1656 (0x678)
Reserved: 0
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 60 (0x3c)
Protocol: TCP(6)
Header Checksum: 77 72
Source IP: 89.7.254.54 (super)
Destination IP: 89.76.80.54 (embedded-pc)
```

```
TCP=======================================================
Source Port: POP3(110)
Destination Port: (1427)
Sequence Number: 1240192170 (0x49ebd4aa)
Acknowledgement Number: 16058285 (0xf507ad)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 4096 (0x1000)
Checksum: 76 DD
Urgent Pointer: 0
DATA======================================================
Data:
0000 -- 2B 4F 4B 20 50 6F 70 20 73 65  +OK Pop se
0010 -- 72 76 65 72 20 61 74 20 73 75  rver at su
0020 -- 70 65 72 20 73 69 67 6E 69 6E  per signin
0030 -- 67 20 6F 66 66 2E 0D 0A        g off...


***** Packet ID: 14 *****
ETHERNET==================================================
Destination Address: 080020-1310d2 (super)
Source Address: 006097-9d6b1d (embedded-pc)
Ethernet Type: 08-00 (IP)
IP========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 40 (0x28)
Identification: 58454 (0xe456)
Reserved: 0
Don't Fragment (DF): Don't Fragment(1)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 32 (0x20)
Protocol: TCP(6)
Header Checksum: 75 B9
Source IP: 89.76.80.54 (embedded-pc)
Destination IP: 89.7.254.54 (super)
TCP=======================================================
Source Port: (1427)
Destination Port: POP3(110)
Sequence Number: 16058285 (0xf507ad)
Acknowledgement Number: 1240192208 (0x49ebd4d0)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 0
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 1
Window Size: 8553 (0x2169)
Checksum: 60 4B
```

```
Urgent Pointer: 0

***** Packet ID: 15 *****
ETHERNET==================================================
Destination Address: 006097-9d6b1d (embedded-pc)
Source Address: 080020-1310d2 (super)
Ethernet Type: 08-00 (IP)
IP========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 40 (0x28)
Identification: 1657 (0x679)
Reserved: 0
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 60 (0x3c)
Protocol: TCP(6)
Header Checksum: 77 97
Source IP: 89.7.254.54 (super)
Destination IP: 89.76.80.54 (embedded-pc)
TCP=======================================================
Source Port: POP3(110)
Destination Port: (1427)
Sequence Number: 1240192208 (0x49ebd4d0)
Acknowledgement Number: 16058286 (0xf507ae)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 0
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 4096 (0x1000)
Checksum: 71 B4
Urgent Pointer: 0

***** Packet ID: 16 *****
ETHERNET==================================================
Destination Address: 006097-9d6b1d (embedded-pc)
Source Address: 080020-1310d2 (super)
Ethernet Type: 08-00 (IP)
IP========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 40 (0x28)
Identification: 1658 (0x67a)
Reserved: 0
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
```

```
Fragment Offset: 0
Time to Live (TTL): 60 (0x3c)
Protocol: TCP(6)
Header Checksum: 77 96
Source IP: 89.7.254.54 (super)
Destination IP: 89.76.80.54 (embedded-pc)
TCP=====================================================
Source Port: POP3(110)
Destination Port: (1427)
Sequence Number: 1240192208 (0x49ebd4d0)
Acknowledgement Number: 16058286 (0xf507ae)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 0
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 1
Window Size: 4096 (0x1000)
Checksum: 71 B3
Urgent Pointer: 0


***** Packet ID: 17 *****
ETHERNET================================================
Destination Address: 080020-1310d2 (super)
Source Address: 006097-9d6b1d (embedded-pc)
Ethernet Type: 08-00 (IP)
IP======================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 40 (0x28)
Identification: 58710 (0xe556)
Reserved: 0
Don't Fragment (DF): Don't Fragment(1)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 32 (0x20)
Protocol: TCP(6)
Header Checksum: 74 B9
Source IP: 89.76.80.54 (embedded-pc)
Destination IP: 89.7.254.54 (super)
TCP=====================================================
Source Port: (1427)
Destination Port: POP3(110)
Sequence Number: 16058286 (0xf507ae)
Acknowledgement Number: 1240192209 (0x49ebd4d1)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 0
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 8553 (0x2169)
Checksum: 60 4A
```

```
Urgent Pointer: 0

***** Packet ID: 18 *****
ETHERNET===================================================
Destination Address: 080020-0dddf9 (c3po)
Source Address: 0000a3-b0022a (TecElite-N.b0022a)
Ethernet Type: 08-00 (IP)
IP=========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 40 (0x28)
Identification: 37459 (0x9253)
Reserved: 0
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 61 (0x3d)
Protocol: TCP(6)
Header Checksum: 15 3D
Source IP: 192.190.175.254 (ftp)
Destination IP: 89.111.12.20 (c3po)
TCP========================================================
Source Port: TELNET(23)
Destination Port: (32779)
Sequence Number: 3652221321 (0xd9b07989)
Acknowledgement Number: 4022713487 (0xefc5bc8f)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 0
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 32736 (0x7fe0)
Checksum: DA 01
Urgent Pointer: 0

***** Packet ID: 19 *****
ETHERNET===================================================
Destination Address: 0000a3-b0022a (TecElite-N.b0022a)
Source Address: 080020-0dddf9 (c3po)
Ethernet Type: 08-00 (IP)
IP=========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 40 (0x28)
Identification: 21585 (0x5451)
Reserved: 0
Don't Fragment (DF): Don't Fragment(1)
More Fragment (MF): Last Fragment(0)
```

```
Fragment Offset: 0
Time to Live (TTL): 255 (0xff)
Protocol: TCP(6)
Header Checksum: 51 3E
Source IP: 89.111.12.20 (c3po)
Destination IP: 192.190.175.254 (ftp)
TCP=====================================================
Source Port: (32779)
Destination Port: TELNET(23)
Sequence Number: 4022713487 (0xefc5bc8f)
Acknowledgement Number: 3652221322 (0xd9b0798a)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 0
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 8760 (0x2238)
Checksum: 37 A9
Urgent Pointer: 0


***** Packet ID: 20 *****
ETHERNET================================================
Destination Address: aa0004-000104 (DEC.000104)
Source Address: 0000e8-061f15 (smtplink)
Ethernet Type: 08-00 (IP)
IP======================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 44 (0x2c)
Identification: 3736 (0xe98)
Reserved: 0
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 64 (0x40)
Protocol: TCP(6)
Header Checksum: 9B 0C
Source IP: 89.7.7.100 (smtplink)
Destination IP: 192.190.175.254 (ftp)
TCP=====================================================
Source Port: (11348)
Destination Port: SMTP(25)
Sequence Number: 104679649 (0x63d48e1)
Acknowledgement Number: 1 (0x1)
Data Offset: 6 (0x6)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 0
Push Function (PSH): 0
Reset Connection (RST): 0
Synchronize Sequence (SYN): 1
No More Data (FIN): 0
Window Size: 2048 (0x800)
Checksum: 47 0E
```

```
Urgent Pointer: 0

***** Packet ID: 21 *****
ETHERNET====================================================
Destination Address: 0000e8-061f15 (smtplink)
Source Address: 0000a3-b0022a (TecElite-N.b0022a)
Ethernet Type: 08-00 (IP)
IP==========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 44 (0x2c)
Identification: 37475 (0x9263)
Reserved: 0
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 62 (0x3e)
Protocol: TCP(6)
Header Checksum: 19 41
Source IP: 192.190.175.254 (ftp)
Destination IP: 89.7.7.100 (smtplink)
TCP=========================================================
Source Port: SMTP(25)
Destination Port: (11348)
Sequence Number: 3878102034 (0xe7272412)
Acknowledgement Number: 104679650 (0x63d48e2)
Data Offset: 6 (0x6)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 0
Reset Connection (RST): 0
Synchronize Sequence (SYN): 1
No More Data (FIN): 0
Window Size: 32736 (0x7fe0)
Checksum: C3 E3
Urgent Pointer: 0

***** Packet ID: 22 *****
ETHERNET====================================================
Destination Address: aa0004-000104 (DEC.000104)
Source Address: 0000e8-061f15 (smtplink)
Ethernet Type: 08-00 (IP)
IP==========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 40 (0x28)
Identification: 3737 (0xe99)
Reserved: 0
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
```

```
Fragment Offset: 0
Time to Live (TTL): 64 (0x40)
Protocol: TCP(6)
Header Checksum: 9B 0F
Source IP: 89.7.7.100 (smtplink)
Destination IP: 192.190.175.254 (ftp)
TCP=========================================================
Source Port: (11348)
Destination Port: SMTP(25)
Sequence Number: 104679650 (0x63d48e2)
Acknowledgement Number: 3878102035 (0xe7272413)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 0
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 2048 (0x800)
Checksum: 4F E5
Urgent Pointer: 0


***** Packet ID: 23 *****
ETHERNET====================================================
Destination Address: 0000e8-061f15 (smtplink)
Source Address: 0000a3-b0022a (TecElite-N.b0022a)
Ethernet Type: 08-00 (IP)
IP==========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 44 (0x2c)
Identification: 37476 (0x9264)
Reserved: 0
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 62 (0x3e)
Protocol: TCP(6)
Header Checksum: 19 40
Source IP: 192.190.175.254 (ftp)
Destination IP: 89.7.7.100 (smtplink)
TCP=========================================================
Source Port: (12998)
Destination Port: (113)
Sequence Number: 2356842160 (0x8c7a8eb0)
Acknowledgement Number: 0
Data Offset: 6 (0x6)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 0
Push Function (PSH): 0
Reset Connection (RST): 0
Synchronize Sequence (SYN): 1
No More Data (FIN): 0
Window Size: 512 (0x200)
Checksum: 76 9C
```

```
Urgent Pointer: 0

***** Packet ID: 24 *****
ETHERNET==================================================
Destination Address: aa0004-000104 (DEC.000104)
Source Address: 0000e8-061f15 (smtplink)
Ethernet Type: 08-00 (IP)
IP========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 40 (0x28)
Identification: 3738 (0xe9a)
Reserved: 0
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 64 (0x40)
Protocol: TCP(6)
Header Checksum: 9B 0E
Source IP: 89.7.7.100 (smtplink)
Destination IP: 192.190.175.254 (ftp)
TCP=======================================================
Source Port: (113)
Destination Port: (12998)
Sequence Number: 0
Acknowledgement Number: 2356842161 (0x8c7a8eb1)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 1
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 0
Checksum: 90 3D
Urgent Pointer: 0

***** Packet ID: 25 *****
ETHERNET==================================================
Destination Address: 0000e8-061f15 (smtplink)
Source Address: 0000a3-b0022a (TecElite-N.b0022a)
Ethernet Type: 08-00 (IP)
IP========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 125 (0x7d)
Identification: 37477 (0x9265)
Reserved: 0
Don't Fragment (DF): Don't Fragment(1)
More Fragment (MF): Last Fragment(0)
```

```
Fragment Offset: 0
Time to Live (TTL): 62 (0x3e)
Protocol: TCP(6)
Header Checksum: D8 ED
Source IP: 192.190.175.254 (ftp)
Destination IP: 89.7.7.100 (smtplink)
TCP=====================================================
Source Port: SMTP(25)
Destination Port: (11348)
Sequence Number: 3878102035 (0xe7272413)
Acknowledgement Number: 104679650 (0x63d48e2)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 32736 (0x7fe0)
Checksum: FC 9F
Urgent Pointer: 0
DATA====================================================
Data:
0000 -- 32 32 30 20 6E 61 74 61 64 6D  220 natadm
0010 -- 2E 74 65 63 65 6C 69 74 65 2E  .tecelite.
0020 -- 63 6F 6D 20 45 53 4D 54 50 20  com ESMTP
0030 -- 53 65 6E 64 6D 61 69 6C 20 38  Sendmail 8
0040 -- 2E 38 2E 37 2F 38 2E 38 2E 37  .8.7/8.8.7
0050 -- 3B 20 54 68 75 2C 20 31 30 20  ; Thu, 10
0060 -- 53 65 70 20 31 39 39 38 20 31  Sep 1998 1
0070 -- 37 3A 32 38 3A 31 30 20 2D 30  7:28:10 -0
0080 -- 37 30 30 0D 0A       700..

***** Packet ID: 26 *****
ETHERNET================================================
Destination Address: aa0004-000104 (DEC.000104)
Source Address: 0000e8-061f15 (smtplink)
Ethernet Type: 08-00 (IP)
IP======================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 68 (0x44)
Identification: 3739 (0xe9b)
Reserved: 0
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 64 (0x40)
Protocol: TCP(6)
Header Checksum: 9A F1
Source IP: 89.7.7.100 (smtplink)
Destination IP: 192.190.175.254 (ftp)
TCP=====================================================
Source Port: (11348)
Destination Port: SMTP(25)
Sequence Number: 104679650 (0x63d48e2)
```

```
Acknowledgement Number: 3878102120 (0xe7272468)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 1963 (0x7ab)
Checksum: 84 C7
Urgent Pointer: 0
DATA=====================================================
Data:
0000 -- 48 45 4C 4F 20 73 6D 74 70 6C  HELO smtpl
0010 -- 69 6E 6B 2E 74 65 63 65 6C 69  ink.teceli
0020 -- 74 65 2E 63 6F 6D 0D 0A        te.com..


***** Packet ID: 27 *****
ETHERNET=================================================
Destination Address: 0000e8-061f15 (smtplink)
Source Address: 0000a3-b0022a (TecElite-N.b0022a)
Ethernet Type: 08-00 (IP)
IP=======================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 114 (0x72)
Identification: 37478 (0x9266)
Reserved: 0
Don't Fragment (DF): Don't Fragment(1)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 62 (0x3e)
Protocol: TCP(6)
Header Checksum: D8 F7
Source IP: 192.190.175.254 (ftp)
Destination IP: 89.7.7.100 (smtplink)
TCP======================================================
Source Port: SMTP(25)
Destination Port: (11348)
Sequence Number: 3878102120 (0xe7272468)
Acknowledgement Number: 104679678 (0x63d48fe)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 32736 (0x7fe0)
Checksum: EA FB
Urgent Pointer: 0
DATA=====================================================
Data:
0000 -- 32 35 30 20 6E 61 74 61 64 6D  250 natadm
0010 -- 2E 74 65 63 65 6C 69 74 65 2E  .tecelite.
```

```
0020 -- 63 6F 6D 20 48 65 6C 6C 6F 20  com Hello
0030 -- 73 6D 74 70 6C 69 6E 6B 20 5B  smtplink [
0040 -- 38 39 2E 37 2E 37 2E 31 30 30  89.7.7.100
0050 -- 5D 2C 20 70 6C 65 61 73 65 64  ], pleased
0060 -- 20 74 6F 20 6D 65 65 74 20 79   to meet y
0070 -- 6F 75 0D 0A        ou..

***** Packet ID: 28 *****
ETHERNET=================================================
Destination Address: aa0004-000104 (DEC.000104)
Source Address: 0000e8-061f15 (smtplink)
Ethernet Type: 08-00 (IP)
IP=======================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 88 (0x58)
Identification: 3740 (0xe9c)
Reserved: 0
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 64 (0x40)
Protocol: TCP(6)
Header Checksum: 9A DC
Source IP: 89.7.7.100 (smtplink)
Destination IP: 192.190.175.254 (ftp)
TCP======================================================
Source Port: (11348)
Destination Port: SMTP(25)
Sequence Number: 104679678 (0x63d48fe)
Acknowledgement Number: 3878102194 (0xe72724b2)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 1889 (0x761)
Checksum: 6B B6
Urgent Pointer: 0
DATA=====================================================
Data:
0000 -- 4D 41 49 4C 20 46 52 4F 4D 3A  MAIL FROM:
0010 -- 3C 44 6F 75 67 20 46 65 6C 64  <Doug Feld
0020 -- 65 72 20 3C 64 66 65 6C 64 65  er <dfelde
0030 -- 72 40 74 65 63 65 6C 69 74 65  r@tecelite
0040 -- 2E 63 6F 6D 3E 3E 0D 0A        .com>>..

***** Packet ID: 29 *****
ETHERNET=================================================
Destination Address: 0000e8-061f15 (smtplink)
Source Address: 0000a3-b0022a (TecElite-N.b0022a)
Ethernet Type: 08-00 (IP)
IP=======================================================
```

```
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 40 (0x28)
Identification: 37481 (0x9269)
Reserved: 0
Don't Fragment (DF): Don't Fragment(1)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 62 (0x3e)
Protocol: TCP(6)
Header Checksum: D9 3E
Source IP: 192.190.175.254 (ftp)
Destination IP: 89.7.7.100 (smtplink)
TCP=====================================================
Source Port: SMTP(25)
Destination Port: (11348)
Sequence Number: 3878102194 (0xe72724b2)
Acknowledgement Number: 104679726 (0x63d492e)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 0
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 32736 (0x7fe0)
Checksum: D7 19
Urgent Pointer: 0


***** Packet ID: 30 *****
ETHERNET================================================
Destination Address: 0000e8-061f15 (smtplink)
Source Address: 0000a3-b0022a (TecElite-N.b0022a)
Ethernet Type: 08-00 (IP)
IP======================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 95 (0x5f)
Identification: 37482 (0x926a)
Reserved: 0
Don't Fragment (DF): Don't Fragment(1)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 62 (0x3e)
Protocol: TCP(6)
Header Checksum: D9 06
Source IP: 192.190.175.254 (ftp)
Destination IP: 89.7.7.100 (smtplink)
TCP=====================================================
Source Port: SMTP(25)
Destination Port: (11348)
```

```
Sequence Number: 3878102194 (0xe72724b2)
Acknowledgement Number: 104679726 (0x63d492e)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 32736 (0x7fe0)
Checksum: DB 0A
Urgent Pointer: 0
DATA=======================================================
Data:
0000 -- 32 35 30 20 3C 44 6F 75 67 20  250 <Doug
0010 -- 46 65 6C 64 65 72 20 3C 64 66  Felder <df
0020 -- 65 6C 64 65 72 40 74 65 63 65  elder@tece
0030 -- 6C 69 74 65 2E 63 6F 6D 3E 3E  lite.com>>
0040 -- 2E 2E 2E 20 53 65 6E 64 65 72  ... Sender
0050 -- 20 6F 6B 0D 0A                ok..


***** Packet ID: 31 *****
ETHERNET===================================================
Destination Address: aa0004-000104 (DEC.000104)
Source Address: 0000e8-061f15 (smtplink)
Ethernet Type: 08-00 (IP)
IP=========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 70 (0x46)
Identification: 3741 (0xe9d)
Reserved: 0
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 64 (0x40)
Protocol: TCP(6)
Header Checksum: 9A ED
Source IP: 89.7.7.100 (smtplink)
Destination IP: 192.190.175.254 (ftp)
TCP========================================================
Source Port: (11348)
Destination Port: SMTP(25)
Sequence Number: 104679726 (0x63d492e)
Acknowledgement Number: 3878102249 (0xe72724e9)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 1834 (0x72a)
Checksum: 43 E8
Urgent Pointer: 0
```

```
DATA=========================================================
Data:
0000 -- 52 43 50 54 20 54 4F 3A 3C 6B  RCPT TO:<k
0010 -- 61 68 6D 69 6E 2E 74 65 68 40  ahmin.teh@
0020 -- 61 6D 64 2E 63 6F 6D 3E 0D 0A  amd.com>..


***** Packet ID: 32 *****
ETHERNET=====================================================
Destination Address: 0000e8-061f15 (smtplink)
Source Address: 0000a3-b0022a (TecElite-N.b0022a)
Ethernet Type: 08-00 (IP)
IP===========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 40 (0x28)
Identification: 37485 (0x926d)
Reserved: 0
Don't Fragment (DF): Don't Fragment(1)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 62 (0x3e)
Protocol: TCP(6)
Header Checksum: D9 3A
Source IP: 192.190.175.254 (ftp)
Destination IP: 89.7.7.100 (smtplink)
TCP==========================================================
Source Port: SMTP(25)
Destination Port: (11348)
Sequence Number: 3878102249 (0xe72724e9)
Acknowledgement Number: 104679756 (0x63d494c)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 0
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 32736 (0x7fe0)
Checksum: D6 C4
Urgent Pointer: 0


***** Packet ID: 33 *****
ETHERNET=====================================================
Destination Address: 0000e8-061f15 (smtplink)
Source Address: 0000a3-b0022a (TecElite-N.b0022a)
Ethernet Type: 08-00 (IP)
IP===========================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 82 (0x52)
```

```
Identification: 37493 (0x9275)
Reserved: 0
Don't Fragment (DF): Don't Fragment(1)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 62 (0x3e)
Protocol: TCP(6)
Header Checksum: D9 08
Source IP: 192.190.175.254 (ftp)
Destination IP: 89.7.7.100 (smtplink)
TCP=====================================================
Source Port: SMTP(25)
Destination Port: (11348)
Sequence Number: 3878102249 (0xe72724e9)
Acknowledgement Number: 104679756 (0x63d494c)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 32736 (0x7fe0)
Checksum: AF 57
Urgent Pointer: 0
DATA====================================================
Data:
0000 -- 32 35 30 20 3C 6B 61 68 6D 69  250 <kahmi
0010 -- 6E 2E 74 65 68 40 61 6D 64 2E  n.teh@amd.
0020 -- 63 6F 6D 3E 2E 2E 2E 20 52 65  com>... Re
0030 -- 63 69 70 69 65 6E 74 20 6F 6B  cipient ok
0040 -- 0D 0A                          ..

***** Packet ID: 34 *****
ETHERNET================================================
Destination Address: aa0004-000104 (DEC.000104)
Source Address: 0000e8-061f15 (smtplink)
Ethernet Type: 08-00 (IP)
IP======================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 47 (0x2f)
Identification: 3742 (0xe9e)
Reserved: 0
Don't Fragment (DF): May Fragment(0)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 64 (0x40)
Protocol: TCP(6)
Header Checksum: 9B 03
Source IP: 89.7.7.100 (smtplink)
Destination IP: 192.190.175.254 (ftp)
TCP=====================================================
Source Port: (11348)
Destination Port: SMTP(25)
Sequence Number: 104679756 (0x63d494c)
```

```
Acknowledgement Number: 3878102291 (0xe7272513)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
No More Data (FIN): 0
Window Size: 1792 (0x700)
Checksum: 8C DC
Urgent Pointer: 0
DATA=====================================================
Data: 44 41 54 41 20 0D 0A    DATA ..


***** Packet ID: 35 *****
ETHERNET=================================================
Destination Address: 0000e8-061f15 (smtplink)
Source Address: 0000a3-b0022a (TecElite-N.b0022a)
Ethernet Type: 08-00 (IP)
IP=======================================================
Version: 4
Header Length: 5 (0x5)
Type of Service: 00
TOS Precedence: Routine(0)
TOS Delay: Normal Delay(0)
TOS Throughput: Normal Throughput(0)
TOS Reliability: Normal Reliability(0)
Total Length: 90 (0x5a)
Identification: 37494 (0x9276)
Reserved: 0
Don't Fragment (DF): Don't Fragment(1)
More Fragment (MF): Last Fragment(0)
Fragment Offset: 0
Time to Live (TTL): 62 (0x3e)
Protocol: TCP(6)
Header Checksum: D8 FF
Source IP: 192.190.175.254 (ftp)
Destination IP: 89.7.7.100 (smtplink)
TCP======================================================
Source Port: SMTP(25)
Destination Port: (11348)
Sequence Number: 3878102291 (0xe7272513)
Acknowledgement Number: 104679763 (0x63d4953)
Data Offset: 5 (0x5)
Reserved: 0
Urgent Field (URG): 0
Acknowledgement field (ACK): 1
Push Function (PSH): 1
Reset Connection (RST): 0
Synchronize Sequence (SYN): 0
N
```

- **Exhibit B5:** The contents of files mfaptpkt.txt and mfaptpkt2.txt that are files that contain the elements that were extracted by the parsing/extracting of

42
08 00 20 13 10 D2 00 A0 24 75 C7 78 08 00 45 00
00 30 B0 6C 40 00 80 06 94 14 59 06 06 03 59 07
FE 36 09 53 00 6E 1A 5D 8A 6E 50 DA 49 60 50 18
1D 4B BF 97 00 00 52 45 54 52 20 32 0D 0A FD 6E
9D F5
_
4B
00 A0 24 75 C7 78 08 00 20 13 10 D2 08 00 45 00
00 39 16 03 00 00 3C 06 B2 75 59 07 FE 36 59 06
06 03 00 6E 09 53 50 DA 49 60 1A 5D 8A 76 50 18
10 00 5D 6C 00 00 2B 4F 4B 20 31 33 35 31 20 6F
63 74 65 74 73 0D 0A CA E0 6A B1
_
40
08 00 20 13 10 D2 00 A0 24 75 C7 78 08 00 45 00
00 28 B1 6C 40 00 80 06 93 1C 59 06 06 03 59 07
FE 36 09 53 00 6E 1A 5D 8A 76 50 DA 49 71 50 10
1D 3A 93 73 00 00 00 00 00 00 00 00 02 03 21 C2
_
460
00 A0 24 75 C7 78 08 00 20 13 10 D2 08 00 45 00
05 5C 16 05 00 00 3C 06 AD 50 59 07 FE 36 59 06
06 03 00 6E 09 53 50 DA 49 71 1A 5D 8A 76 50 18
10 00 53 11 00 00 52 65 74 75 72 6E 2D 50 61 74
68 3A 20 3C 6A 6D 65 74 7A 67 65 72 40 74 65 63
65 6C 69 74 65 2E 63 6F 6D 3E 0D 0A 52 65 63 65
69 76 65 64 3A 20 66 72 6F 6D 20 6E 61 74 61 64
6D 2E 74 65 63 65 6C 69 74 65 2E 63 6F 6D 20 62
79 20 73 75 70 65 72 2E 74 65 63 65 6C 69 74 65
2E 63 6F 6D 20 28 34 2E 31 2F 53 4D 49 2D 34 2E
31 29 0D 0A 09 69 64 20 41 41 32 38 34 30 38 3B
20 54 68 75 2C 20 31 30 20 53 65 70 20 39 38 20
31 37 3A 33 37 3A 33 37 20 50 44 54 0D 0A 52 65
63 65 69 76 65 64 3A 20 66 72 6F 6D 20 73 6D 74
70 6C 69 6E 6B 2E 74 65 63 65 6C 69 74 65 2E 63
6F 6D 20 28 73 6D 74 70 6C 69 6E 6B 20 5B 38 39
2E 37 2E 37 2E 31 30 30 5D 29 0D 0A 09 62 79 20
6E 61 74 61 64 6D 2E 74 65 63 65 6C 69 74 65 2E
63 6F 6D 20 28 38 2E 38 2E 37 2F 38 2E 38 2E 37
29 20 77 69 74 68 20 53 4D 54 50 20 69 64 20 52
41 41 31 37 32 34 35 3B 0D 0A 09 54 68 75 2C 20
31 30 20 53 65 70 20 31 39 39 38 20 31 37 3A 33
39 3A 30 34 20 2D 30 37 30 30 0D 0A 52 65 63 65
69 76 65 64 3A 20 66 72 6F 6D 20 63 63 3A 4D 61
69 6C 20 62 79 20 73 6D 74 70 6C 69 6E 6B 2E 74
65 63 65 6C 69 74 65 2E 63 6F 6D 0D 0A 09 69 64
20 41 41 39 30 35 34 37 34 38 32 39 20 54 68 75
2C 20 31 30 20 53 65 70 20 39 38 20 31 37 3A 34
37 3A 30 39 20 50 44 54 0D 0A 44 61 74 65 3A 20
54 68 75 2C 20 31 30 20 53 65 70 20 39 38 20 31
37 3A 34 37 3A 30 39 20 50 44 54 0D 0A 46 72 6F
6D 3A 20 4A 6F 68 6E 20 4D 65 74 7A 67 65 72 20
3C 6A 6D 65 74 7A 67 65 72 40 74 65 63 65 6C 69
74 65 2E 63 6F 6D 3E 0D 0A 45 6E 63 6F 64 69 6E
67 3A 20 33 32 34 20 54 65 78 74 0D 0A 4D 65 73
73 61 67 65 2D 49 64 3A 20 3C 39 38 30 38 31 30
39 30 35 34 2E 41 41 39 30 35 34 37 34 38 32 39
40 73 6D 74 70 6C 69 6E 6B 2E 74 65 63 65 6C 69
74 65 2E 63 6F 6D 3E 0D 0A 54 6F 3A 20 62 6C 65
61 76 79 40 74 65 63 65 6C 69 74 65 2E 63 6F 6D
2C 20 61 63 68 61 64 64 61 40 74 65 63 65 6C 69
74 65 2E 63 6F 6D 2C 20 64 61 76 65 63 40 74 65

63 65 6C 69 74 65 2E 63 6F 6D 2C 0D 0A 20 20 20
20 20 20 20 20 44 61 76 69 64 20 4C 75 6F 20 3C
64 6C 75 6F 40 74 65 63 65 6C 69 74 65 2E 63 6F
6D 3E 2C 20 6C 6F 77 64 65 72 40 74 65 63 65 6C
69 74 65 2E 63 6F 6D 2C 0D 0A 20 20 20 20 20 20
20 20 65 77 68 65 65 6C 65 72 40 74 65 63 65 6C
69 74 65 2E 63 6F 6D 2C 20 66 6E 6F 6F 6E 40 74
65 63 65 6C 69 74 65 2E 63 6F 6D 2C 20 66 72 65
64 6D 40 74 65 63 65 6C 69 74 65 2E 63 6F 6D 2C
0D 0A 20 20 20 20 20 20 20 20 6A 6D 61 69 78 6E
65 72 40 74 65 63 65 6C 69 74 65 2E 63 6F 6D 2C
20 6A 6F 74 69 73 40 74 65 63 65 6C 69 74 65 2E
63 6F 6D 2C 0D 0A 20 20 20 20 20 20 20 20 4B 69
6D 20 44 61 76 69 73 20 3C 6B 64 61 76 69 73 40
74 65 63 65 6C 69 74 65 2E 63 6F 6D 3E 2C 20 72
61 6D 40 74 65 63 65 6C 69 74 65 2E 63 6F 6D 2C
0D 0A 20 20 20 20 20 20 20 20 52 6F 62 20 52 69
74 7A 20 3C 72 72 69 74 7A 40 74 65 63 65 6C 69
74 65 2E 63 6F 6D 3E 2C 20 72 73 64 69 65 74 7A
40 74 65 63 65 6C 69 74 65 2E 63 6F 6D 2C 20 73
6B 69 70 40 74 65 63 65 6C 69 74 65 2E 63 6F 6D
0D 0A 53 75 62 6A 65 63 74 3A 20 4E 65 78 74 20
47 65 6E 65 72 61 74 69 6F 6E 20 50 72 6F 64 75
63 74 20 44 69 73 63 75 73 73 69 6F 6E 0D 0A 0D
0A 0D 0A 53 75 62 6A 65 63 74 3A 20 4E 65 78 74
20 47 65 6E 65 72 61 74 69 6F 6E 20 50 72 6F 64
75 63 74 20 44 69 73 63 75 73 73 69 6F 6E 0D 0A
0D 0A 49 20 77 6F 75 6C 64 20 6C 69 6B 65 20 74
_
40
08 00 20 13 10 D2 00 A0 24 75 C7 78 08 00 45 00
00 28 B2 6C 40 00 80 06 92 1C 59 06 06 03 59 07
FE 36 09 53 00 6E 1A 5D 8A 76 50 DA 4E A5 50 10
22 38 89 41 00 00 00 00 00 00 00 00 BA 3C 6B D6
_

05
42
08 00 20 13 10 D2 00 A0 24 75 C7 78 08 00 45 00
00 30 B0 6C 40 00 80 06 94 14 59 06 06 03 59 07
FE 36 09 53 00 6E 1A 5D 8A 6E 50 DA 49 60 50 18
1D 4B BF 97 00 00 52 45 54 52 20 32 0D 0A FD 6E
9D F5
4B
00 A0 24 75 C7 78 08 00 20 13 10 D2 08 00 45 00
00 39 16 03 00 00 3C 06 B2 75 59 07 FE 36 59 06
06 03 00 6E 09 53 50 DA 49 60 1A 5D 8A 76 50 18
10 00 5D 6C 00 00 2B 4F 4B 20 31 33 35 31 20 6F
63 74 65 74 73 0D 0A CA E0 6A B1
40
08 00 20 13 10 D2 00 A0 24 75 C7 78 08 00 45 00
00 28 B1 6C 40 00 80 06 93 1C 59 06 06 03 59 07
FE 36 09 53 00 6E 1A 5D 8A 76 50 DA 49 71 50 10
1D 3A 93 73 00 00 00 00 00 00 00 00 02 03 21 C2
0460
00 A0 24 75 C7 78 08 00 20 13 10 D2 08 00 45 00
05 5C 16 05 00 00 3C 06 AD 50 59 07 FE 36 59 06
06 03 00 6E 09 53 50 DA 49 71 1A 5D 8A 76 50 18
10 00 53 11 00 00 52 65 74 75 72 6E 2D 50 61 74
68 3A 20 3C 6A 6D 65 74 7A 67 65 72 40 74 65 63
65 6C 69 74 65 2E 63 6F 6D 3E 0D 0A 52 65 63 65
69 76 65 64 3A 20 66 72 6F 6D 20 6E 61 74 61 64
6D 2E 74 65 63 65 6C 69 74 65 2E 63 6F 6D 20 62
79 20 73 75 70 65 72 2E 74 65 63 65 6C 69 74 65
2E 63 6F 6D 20 28 34 2E 31 2F 53 4D 49 2D 34 2E
31 29 0D 0A 09 69 64 20 41 41 32 38 34 30 38 3B
20 54 68 75 2C 20 31 30 20 53 65 70 20 39 38 20
31 37 3A 33 37 3A 33 37 20 50 44 54 0D 0A 52 65
63 65 69 76 65 64 3A 20 66 72 6F 6D 20 73 6D 74
70 6C 69 6E 6B 2E 74 65 63 65 6C 69 74 65 2E 63
6F 6D 20 28 73 6D 74 70 6C 69 6E 6B 20 5B 38 39
2E 37 2E 37 2E 31 30 30 5D 29 0D 0A 09 62 79 20
6E 61 74 61 64 6D 2E 74 65 63 65 6C 69 74 65 2E
63 6F 6D 20 28 38 2E 38 2E 37 2F 38 2E 38 2E 37
29 20 77 69 74 68 20 53 4D 54 50 20 69 64 20 52
41 41 31 37 32 34 35 3B 0D 0A 09 54 68 75 2C 20
31 30 20 53 65 70 20 31 39 39 38 20 31 37 3A 33
39 3A 30 34 20 2D 30 37 30 30 0D 0A 52 65 63 65
69 76 65 64 3A 20 66 72 6F 6D 20 63 63 3A 4D 61
69 6C 20 62 79 20 73 6D 74 70 6C 69 6E 6B 2E 74
65 63 65 6C 69 74 65 2E 63 6F 6D 0D 0A 09 69 64
20 41 41 39 30 35 34 37 34 38 32 39 20 54 68 75
2C 20 31 30 20 53 65 70 20 39 38 20 31 37 3A 34
37 3A 30 39 20 50 44 54 0D 0A 44 61 74 65 3A 20
54 68 75 2C 20 31 30 20 53 65 70 20 39 38 20 31
37 3A 34 37 3A 30 39 20 50 44 54 0D 0A 46 72 6F
6D 3A 20 4A 6F 68 6E 20 4D 65 74 7A 67 65 72 20
3C 6A 6D 65 74 7A 67 65 72 40 74 65 63 65 6C 69
74 65 2E 63 6F 6D 3E 0D 0A 45 6E 63 6F 64 69 6E
67 3A 20 33 32 34 20 54 65 78 74 0D 0A 4D 65 73
73 61 67 65 2D 49 64 3A 20 3C 39 38 30 38 31 30
39 30 35 34 2E 41 41 39 30 35 34 37 34 38 32 39
40 73 6D 74 70 6C 69 6E 6B 2E 74 65 63 65 6C 69
74 65 2E 63 6F 6D 3E 0D 0A 54 6F 3A 20 62 6C 65
61 76 79 40 74 65 63 65 6C 69 74 65 2E 63 6F 6D
2C 20 61 63 68 61 64 64 61 40 74 65 63 65 6C 69
74 65 2E 63 6F 6D 2C 20 64 61 76 65 63 40 74 65
63 65 6C 69 74 65 2E 63 6F 6D 2C 0D 0A 20 20 20
20 20 20 20 20 44 61 76 69 64 20 4C 75 6F 20 3C

64 6C 75 6F 40 74 65 63 65 6C 69 74 65 2E 63 6F
6D 3E 2C 20 6C 6F 77 64 65 72 40 74 65 63 65 6C
69 74 65 2E 63 6F 6D 2C 0D 0A 20 20 20 20 20 20
20 20 65 77 68 65 65 6C 65 72 40 74 65 63 65 6C
69 74 65 2E 63 6F 6D 2C 20 66 6E 6F 6F 6E 40 74
65 63 65 6C 69 74 65 2E 63 6F 6D 2C 20 66 72 65
64 6D 40 74 65 63 65 6C 69 74 65 2E 63 6F 6D 2C
0D 0A 20 20 20 20 20 20 20 20 6A 6D 61 69 78 6E
65 72 40 74 65 63 65 6C 69 74 65 2E 63 6F 6D 2C
20 6A 6F 74 69 73 40 74 65 63 65 6C 69 74 65 2E
63 6F 6D 2C 0D 0A 20 20 20 20 20 20 20 20 4B 69
6D 20 44 61 76 69 73 20 3C 6B 64 61 76 69 73 40
74 65 63 65 6C 69 74 65 2E 63 6F 6D 3E 2C 20 72
61 6D 40 74 65 63 65 6C 69 74 65 2E 63 6F 6D 2C
0D 0A 20 20 20 20 20 20 20 20 52 6F 62 20 52 69
74 7A 20 3C 72 72 69 74 7A 40 74 65 63 65 6C 69
74 65 2E 63 6F 6D 3E 2C 20 72 73 64 69 65 74 7A
40 74 65 63 65 6C 69 74 65 2E 63 6F 6D 2C 20 73
6B 69 70 40 74 65 63 65 6C 69 74 65 2E 63 6F 6D
0D 0A 53 75 62 6A 65 63 74 3A 20 4E 65 78 74 20
47 65 6E 65 72 61 74 69 6F 6E 20 50 72 6F 64 75
63 74 20 44 69 73 63 75 73 73 69 6F 6E 0D 0A 0D
0A 0D 0A 53 75 62 6A 65 63 74 3A 20 4E 65 78 74
20 47 65 6E 65 72 61 74 69 6F 6E 20 50 72 6F 64
75 63 74 20 44 69 73 63 75 73 73 69 6F 6E 0D 0A
0D 0A 49 20 77 6F 75 6C 64 20 6C 69 6B 65 20 74
40
08 00 20 13 10 D2 00 A0 24 75 C7 78 08 00 45 00
00 28 B2 6C 40 00 80 06 92 1C 59 06 06 03 59 07
FE 36 09 53 00 6E 1A 5D 8A 76 50 DA 4E A5 50 10
22 38 89 41 00 00 00 00 00 00 00 00 BA 3C 6B D6

- **Exhibit B6:** The contents of files mfaptkey.txt and mfaptkey2.txt that are files that contain the keys that were generated from the extracted data (Exhibit B4) and used for looking up the flow-entry database per element (c) of method claims 11 and 54, which are operations carried out by the lookup engine of element (e) of claim 29.

```
00 00 00 00 08 00 20 13 10 D2 00 A0 24 75 C7 78
00 08 59 07 FE 36 00 00 00 00 00 00 00 00 00
00 00 59 06 06 03 00 00 00 00 00 00 00 00 00
00 00 00 2B 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 47 00 6E 00 00 09 53 00 00
*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*
00 00 00 00 00 A0 24 75 C7 78 08 00 20 13 10 D2
00 08 59 06 06 03 00 00 00 00 00 00 00 00 00
00 00 59 07 FE 36 00 00 00 00 00 00 00 00 00
00 00 00 2B 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 47 09 53 00 00 00 6E 00 00
*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*
00 00 00 00 08 00 20 13 10 D2 00 A0 24 75 C7 78
00 08 59 07 FE 36 00 00 00 00 00 00 00 00 00
00 00 59 06 06 03 00 00 00 00 00 00 00 00 00
00 00 00 2B 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 47 00 6E 00 00 09 53 00 00
*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*
00 00 00 00 00 A0 24 75 C7 78 08 00 20 13 10 D2
00 08 59 06 06 03 00 00 00 00 00 00 00 00 00
00 00 59 07 FE 36 00 00 00 00 00 00 00 00 00
00 00 00 2B 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 47 09 53 00 00 00 6E 00 00
*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*
00 00 00 00 08 00 20 13 10 D2 00 A0 24 75 C7 78
00 08 59 07 FE 36 00 00 00 00 00 00 00 00 00
00 00 59 06 06 03 00 00 00 00 00 00 00 00 00
00 00 00 2B 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 47 00 6E 00 00 09 53 00 00
*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*
```

05

```
00 00 00 00 08 00 20 13 10 D2 00 A0 24 75 C7 78
00 08 59 07 FE 36 00 00 00 00 00 00 00 00 00 00
00 00 59 06 06 03 00 00 00 00 00 00 00 00 00 00
00 00 00 2B 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 47 00 6E 00 00 09 53 00 00

00 00 00 00 00 A0 24 75 C7 78 08 00 20 13 10 D2
00 08 59 06 06 03 00 00 00 00 00 00 00 00 00 00
00 00 59 07 FE 36 00 00 00 00 00 00 00 00 00 00
00 00 00 2B 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 47 09 53 00 00 00 6E 00 00

00 00 00 00 08 00 20 13 10 D2 00 A0 24 75 C7 78
00 08 59 07 FE 36 00 00 00 00 00 00 00 00 00 00
00 00 59 06 06 03 00 00 00 00 00 00 00 00 00 00
00 00 00 2B 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 47 00 6E 00 00 09 53 00 00

00 00 00 00 00 A0 24 75 C7 78 08 00 20 13 10 D2
00 08 59 06 06 03 00 00 00 00 00 00 00 00 00 00
00 00 59 07 FE 36 00 00 00 00 00 00 00 00 00 00
00 00 00 2B 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 47 09 53 00 00 00 6E 00 00

00 00 00 00 08 00 20 13 10 D2 00 A0 24 75 C7 78
00 08 59 07 FE 36 00 00 00 00 00 00 00 00 00 00
00 00 59 06 06 03 00 00 00 00 00 00 00 00 00 00
00 00 00 2B 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 47 00 6E 00 00 09 53 00 00
```

- **Exhibit B7:** The contents of file MFATEST.TXT that includes the decoded packets that were generated by operation of the method that includes the elements of each of method claims 11 and 54, by an apparatus that includes the elements of claim 29.

```
HP ProbeView/SNMP File Contains 148 Packets

      1       TO: 0800201310D2                    [redacted]2:53:50<0.000> [redacted]
              FROM: 00A02475C778

Pkt: 1, Len: 66/66
Ethernet:    (00a02475c778 ->   Sun 1310d2)  type: IP(0x800)
Internet:           89.6.6.3 -> 89.7.254.54     hl: 5   ver: 4
   tos: 0  len: 48   id: 0xb06c fragoff: 0   flags: 0x2 ttl: 128
   prot: TCP(6)  xsum: 0x9414
TCP:              2387 -> POP-3(110)     seq: 1a5d8a6e
   ack: 50da4960  win: 7499   hl: 5   xsum: 0xbf97 urg: 0
   flags: <ACK><PUSH>
 data (8/8):

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      2       TO: 00A02475C778                    [redacted]2:53:50<0.002>[redacted]
              FROM: 0800201310D2

Pkt: 2, Len: 75/75
Ethernet:    ( Sun 1310d2 -> 00a02475c778)  type: IP(0x800)
Internet:        89.7.254.54 -> 89.6.6.3        hl: 5   ver: 4
   tos: 0  len: 57   id: 0x1603 fragoff: 0   flags: 00 ttl: 60
   prot: TCP(6)  xsum: 0xb275
TCP:       POP-3(110) -> 2387            seq: 50da4960
   ack: 1a5d8a76  win: 4096   hl: 5   xsum: 0x5d6c urg: 0
   flags: <ACK><PUSH>
 data (17/17):

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      3       TO: 0800201310D2                    [redacted]2:53:51<0.185> [redacted]
              FROM: 00A02475C778

Pkt: 3, Len: 64/64
Ethernet:    (00a02475c778 ->   Sun 1310d2)  type: IP(0x800)
Internet:           89.6.6.3 -> 89.7.254.54     hl: 5   ver: 4
   tos: 0  len: 40   id: 0xb16c fragoff: 0   flags: 0x2 ttl: 128
   prot: TCP(6)  xsum: 0x931c
TCP:              2387 -> POP-3(110)     seq: 1a5d8a76
   ack: 50da4971  win: 7482   hl: 5   xsum: 0x9373 urg: 0
   flags: <ACK>


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      4       TO: 00A02475C778                    [redacted] 22:53:51<0.002>[redacted]
              FROM: 0800201310D2

Pkt: 4, Len: 1120/1390
Ethernet:    ( Sun 1310d2 -> 00a02475c778)  type: IP(0x800)
Internet:        89.7.254.54 -> 89.6.6.3        hl: 5   ver: 4
   tos: 0  len: 1372 id: 0x1605 fragoff: 0   flags: 00 ttl: 60
   prot: TCP(6)  xsum: 0xad50
TCP:       POP-3(110) -> 2387            seq: 50da4971
   ack: 1a5d8a76  win: 4096   hl: 5   xsum: 0x5311 urg: 0
   flags: <ACK><PUSH>
```

```
 data (60/1332):

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      5      TO: 0800201310D2                [redacted]22:53:51<0.198> [redacted]
             FROM: 00A02475C778

Pkt: 5, Len: 64/64
Ethernet:    (00a02475c778 ->   Sun 1310d2)  type: IP(0x800)
Internet:         89.6.6.3 -> 89.7.254.54     hl: 5   ver: 4
   tos: 0  len: 40   id: 0xb26c fragoff: 0   flags: 0x2 ttl: 128
   prot: TCP(6)  xsum: 0x921c
TCP:             2387 -> POP-3(110)      seq: 1a5d8a76
   ack: 50da4ea5  win: 8760   hl: 5   xsum: 0x8941 urg: 0
   flags: <ACK>


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      6      TO: 0800201310D2                [redacted]2:53:53<2.556>[redacted]
             FROM: 00A02475C778

Pkt: 6, Len: 66/66
Ethernet:    (00a02475c778 ->   Sun 1310d2)  type: IP(0x800)
Internet:         89.6.6.3 -> 89.7.254.54     hl: 5   ver: 4
   tos: 0  len: 48   id: 0xb36c fragoff: 0   flags: 0x2 ttl: 128
   prot: TCP(6)  xsum: 0x9114
TCP:             2387 -> POP-3(110)      seq: 1a5d8a76
   ack: 50da4ea5  win: 8760   hl: 5   xsum: 0xcb6a urg: 0
   flags: <ACK><PUSH>
 data (8/8):


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      7      TO: 00A02475C778                [redacted]2:53:53<0.001> [redacted]
             FROM: 0800201310D2

Pkt: 7, Len: 91/91
Ethernet:    (  Sun 1310d2 -> 00a02475c778)  type: IP(0x800)
Internet:      89.7.254.54 -> 89.6.6.3        hl: 5   ver: 4
   tos: 0  len: 73   id: 0x1609 fragoff: 0   flags: 00 ttl: 60
   prot: TCP(6)  xsum: 0xb25f
TCP:       POP-3(110) -> 2387            seq: 50da4ea5
   ack: 1a5d8a7e  win: 4096   hl: 5   xsum: 0x3f4c urg: 0
   flags: <ACK><PUSH>
 data (33/33):


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      8      TO: 0800201310D2                [redacted]2:53:53<0.002> [redacted]
             FROM: 00A02475C778

Pkt: 8, Len: 64/64
Ethernet:    (00a02475c778 ->   Sun 1310d2)  type: IP(0x800)
Internet:         89.6.6.3 -> 89.7.254.54     hl: 5   ver: 4
   tos: 0  len: 46   id: 0xb46c fragoff: 0   flags: 0x2 ttl: 128
   prot: TCP(6)  xsum: 0x9016
TCP:             2387 -> POP-3(110)      seq: 1a5d8a7e
   ack: 50da4ec6  win: 8727   hl: 5   xsum: 0xe177 urg: 0
```

```
   flags: <ACK><PUSH>
 data (6/6):


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      9      TO: 00A02475C778                  [redacted]2:53:53<0.029> [redacted]
             FROM: 0800201310D2

Pkt: 9, Len: 96/96
Ethernet:   (  Sun 1310d2 -> 00a02475c778)  type: IP(0x800)
Internet:      89.7.254.54 -> 89.6.6.3        hl: 5   ver: 4
   tos: 0  len: 78   id: 0x160a fragoff: 0   flags: 00 ttl: 60
   prot: TCP(6)  xsum: 0xb259
TCP:       POP-3(110) -> 2387           seq: 50da4ec6
   ack: 1a5d8a84  win: 4096   hl: 5   xsum: 0xa04c urg: 0
   flags: <ACK><PUSH>
 data (38/38):


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      10     TO: 00A02475C778                  [redacted]2:53:53<0.003> [redacted]
             FROM: 0800201310D2

Pkt: 10, Len: 64/64
Ethernet:   (  Sun 1310d2 -> 00a02475c778)  type: IP(0x800)
Internet:      89.7.254.54 -> 89.6.6.3        hl: 5   ver: 4
   tos: 0  len: 40   id: 0x160b fragoff: 0   flags: 00 ttl: 60
   prot: TCP(6)  xsum: 0xb27e
TCP:       POP-3(110) -> 2387           seq: 50da4eec
   ack: 1a5d8a84  win: 4096   hl: 5   xsum: 0x9b23 urg: 0
   flags: <ACK><FIN>


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      11     TO: 0800201310D2                  [redacted]22:53:53<0.000> [redacted]
             FROM: 00A02475C778

Pkt: 11, Len: 64/64
Ethernet:   (00a02475c778 ->   Sun 1310d2)  type: IP(0x800)
Internet:         89.6.6.3 -> 89.7.254.54     hl: 5   ver: 4
   tos: 0  len: 40   id: 0xb56c fragoff: 0   flags: 0x2 ttl: 128
   prot: TCP(6)  xsum: 0x8f1c
TCP:             2387 -> POP-3(110)     seq: 1a5d8a84
   ack: 50da4eed  win: 8689   hl: 5   xsum: 0x8932 urg: 0
   flags: <ACK>


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      12     TO: 0800201310D2                  [redacted]:53:53<0.031> [redacted]
             FROM: 00A02475C778

Pkt: 12, Len: 64/64
Ethernet:   (00a02475c778 ->   Sun 1310d2)  type: IP(0x800)
Internet:         89.6.6.3 -> 89.7.254.54     hl: 5   ver: 4
   tos: 0  len: 40   id: 0xb66c fragoff: 0   flags: 0x2 ttl: 128
   prot: TCP(6)  xsum: 0x8e1c
TCP:             2387 -> POP-3(110)     seq: 1a5d8a84
```

```
   ack: 50da4eed  win: 8689   hl: 5   xsum: 0x8931 urg: 0
   flags: <ACK><FIN>


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      13      TO: 00A02475C778                   [redacted] 22:53:53<0.001> [redacted]
              FROM: 0800201310D2

Pkt: 13, Len: 64/64
Ethernet:    (  Sun 1310d2 -> 00a02475c778)  type: IP(0x800)
Internet:       89.7.254.54 -> 89.6.6.3         hl: 5   ver: 4
   tos: 0  len: 40   id: 0x160c fragoff: 0   flags: 00 ttl: 60
   prot: TCP(6)  xsum: 0xb27d
TCP:        POP-3(110) -> 2387          seq: 50da4eed
   ack: 1a5d8a85  win: 4096   hl: 5   xsum: 0x9b22 urg: 0
   flags: <ACK>


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      14      TO: 006008C0D710                   [redacted] 22:53:58<4.755> [redacted]
              FROM: 00A076A010F2

Pkt: 14, Len: 64/64
Ethernet:    (00a076a010f2 -> 006008c0d710)  type: IP(0x800)
Internet:         89.89.24.6 -> 89.75.23.24     hl: 5   ver: 4
   tos: 0  len: 44   id: 0x5ffe fragoff: 0   flags: 0x2 ttl: 128
   prot: TCP(6)  xsum: 0xb90b
TCP:              1460 -> netb-ses(139)  seq: 01be3a7e
   ack:   ----    win: 8192   hl: 6   xsum: 0x53e9 urg: 0
   flags: <SYN>  mss: 1460


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      15      TO: 00A076A010F2                   [redacted] 2:53:58<0.001> [redacted]
              FROM: 006008C0D710

Pkt: 15, Len: 64/64
Ethernet:    (006008c0d710 -> 00a076a010f2)  type: IP(0x800)
Internet:       89.75.23.24 -> 89.89.24.6       hl: 5   ver: 4
   tos: 0  len: 44   id: 0x497c fragoff: 0   flags: 0x2 ttl: 128
   prot: TCP(6)  xsum: 0xcf8d
TCP:    netb-ses(139) -> 1460           seq: 1ecd513b
   ack: 01be3a7f  win: 8760   hl: 6   xsum: 0xe197 urg: 0
   flags: <ACK><SYN>  mss: 1460


*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

      16      TO: 006008C0D710                   [redacted] 22:53:58<0.000> [redacted]
              FROM: 00A076A010F2

Pkt: 16, Len: 64/64
Ethernet:    (00a076a010f2 -> 006008c0d710)  type: IP(0x800)
Internet:         89.89.24.6 -> 89.75.23.24     hl: 5   ver: 4
   tos: 0  len: 40   id: 0x60fe fragoff: 0   flags: 0x2 ttl: 128
   prot: TCP(6)  xsum: 0xb80f
```

- **Exhibit B8:** Protocol Definition Language (PDL) Reference Guide (the document MFS-PDL-Reference.pdf) that provides a reference to the protocol definition language used in cpl files.

MeterFlow™
Traffic Classification System

Protocol Definition Language (PDL)
Reference Guide

Version A0.02

[redacted]

DRAFT - 02

Technically Elite

monitoring your enterprise

NOTICE

Restricted Rights Legend

Government Use

Document revision A.02-004, format revision 1.00-000.

Technically Elite, Inc.
6330 San Ignacio Ave.
San Jose, CA 95119-1209, USA
Phone: +1.408.574.2300
Fax: +1.408.629.8300
E-mail: mworks@tecelite.com
URL: http://www.tecelite.com

Contents

1. Introduction

The *MeterFlow* Protocol Definition Language (PDL) is a special purpose language used to describe network protocols and all the fields within the protocol headers.

Within this document, protocol descriptions (PDL files) are referred to as *PDL* or *rules* when there in no risk of confusion with other types of descriptions.

PDL uses both form and organization similar to the data structure definition part of the C programming language and the PERL scripting language. Since PDL was derived from a language used to decode network packet contact, the authors have mixed the language format with the requirements of packet decoding. This results in an expressive language that is very familiar and comfortable for describing packet content and the details required representing a flow.

1.1 Summary

MeterFlow PDL is a non-procedural Forth Generation language (4GL). This means is describes *what* needs to be done without describing *how* to do it. The details of *how* are hidden in the compiler and the Compiled Protocol Layout (CPL) optimization utility.

In addition, it is used to describe network flows by defining which fields are the address fields, which are the protocol type fields, etc.

Once a PDL file is written, it is compiled using the Netscope compiler (**nsc**), which produces the *MeterFlow* database (MeterFlow.db) and the Netscope database (Netscope.db). The MeterFlow database contains the flow definitions and the Netscope database contains the protocol header definitions.

These databases are used by programs like: **mfkeys**, which produces flow keys; **mfcpl**, which produces flow definitions in CPL format; **mfpkts** which produces sample packets of all known protocols; and **netscope**, which decodes Sniffer™ and tcpdump files.

Due to its size, electronic media copies of the documentation are not provided but can be made available if necessary.

1.2 Document Conventions

The following conventions will be used throughout this document:

Small `courier` typeface indicates C code examples or function names. Functions are written with parentheses after them [**`function()`**], variables are written just as their names [**`variables`**], and structure names are written prefixed with "**`struct`**" [**`struct packet`**].

Italics indicate a filename (for instance, *mworks/base/h/base.h*). Filenames will usually be written relative to the root directory of the distribution.

Constants are expressed in decimal, unless written "`0x...`", the C language notation for hexadecimal numbers.

2. Language Structure

TBD

3. Program Structure

A *MeterFlow* PDL decodes and flow set is a non-empty sequence of statements.

There are four basic types of statements or definitions available in *MeterFlow* PDL:

FIELD,
GROUP,
PROTOCOL and
FLOW.

3.1 FIELD Definitions

The FIELD definition is used to define a specific string of bits or bytes in the packet. The FIELD definition has the following format:

```
Name    FIELD
        SYNTAX Type [ { Enums } ]
        DISPLAY-HINT "FormatString"
        LENGTH "Expression"
        FLAGS FieldFlags
        ENCAP FieldName [ , FieldName2 ]
        LOOKUP LookupType [ Filename ]
        ENCODING EncodingType
        DEFAULT "value"
        DESCRIPTION "Description"
```

Where only the **FIELD** and **SYNTAX** lines are required. All the other lines are attribute lines, which define special characteristics about the **FIELD**. Attribute lines are optional and may appear in any order. Each of the attribute lines are described in detail below:

3.1.1 SYNTAX Type [{ Enums }]

This attribute defines the type and, if the type is an INT, BYTESTRING, BITSTRING, or SNMPSEQUENCE type, the enumerated values for the FIELD. The currently defined types are:

INT(*numBits*)	Integer that is *numBits* bits long.
UNSIGNED INT(*numBits*)	Unsigned integer that is *numBits* bits long.

BYTESTRING(*numBytes*)	String that is *numBytes* bytes long.
BYTESTRING(*R1..R2*)	String that ranges in size from *R1* to *R2* bytes.
BITSTRING(*numBits*)	String that is *numBits* bits long.
LSTRING(*lenBytes*)	String with *lenBytes* header.
NSTRING	Null terminated string.
DNSSTRING	DNS encoded string.
SNMPOID	SNMP Object Identifier.
SNMPSEQUENCE	SNMP Sequence.
SNMPTIMETICKS	SNMP TimeTicks.
COMBO *field1 field2*	Combination pseudo field.

3.1.2 DISPLAY-HINT "FormatString"

This attribute is for specifying how the value of the FIELD is displayed. The currently supported formats are:

Numx	Print as a num byte hexidecimal number.
Numd	Print as a num byte decimal number.
Numo	Print as a num byte octal number.
Numb	Print as a num byte binary number.
Numa	Print num bytes in ASCII format.
Text	Print as ASCII text.
HexDump	Print in hexdump format.

3.1.3 LENGTH "Expression"

This attribute defines an expression for determining the FIELD's length. Expressions are arithmetic and can refer to the value of other FIELD's in the packet by adding a $ to the referenced field's name. For example, "($tcpHeaderLen *4) – 20" is a valid expression if tcpHeaderLen is another field defined for the current packet.

3.1.4 FLAGS FieldFlags

The attribute defines some special flags for a FIELD. The currently supported FieldFlags are:

SAMELAYER	Display field on the same layer as the previous field.
NOLABEL	Don't display the field name with the value.

NOSHOW	Decode the field but don't display it.
SWAPPED	The integer value is swapped.

3.1.5 ENCAP FieldName [, FieldName2]

This attribute defines how one packet is encapsulated inside another. Which packet is determined by the value of the FieldName field. If no packet is found using FieldName then FieldName2 is tried.

3.1.6 LOOKUP LookupType [Filename]

This attribute defines how to lookup the name for a particular FIELD value. The currently supported LookupTypes are:

SERVICE	Use getservbyport().
HOSTNAME	Use gethostbyaddr().
MACADDRESS	Use $METERFLOW/conf/mac2ip.cf.
FILE *file*	Use *file* to lookup value.

3.1.7 ENCODING EncodingType

This attribute defines how a FIELD is encoded. Currently, the only supported EncodingType is BER (for Basic Encoding Rules defined by ASN.1).

3.1.8 DEFAULT "value"

This attribute defines the default value to be used for this field when generating sample packets of this protocol.

3.1.9 DESCRIPTION "Description"

This attribute defines the description of the FIELD. It is used for informational purposes only.

3.2 GROUP Definitions

The GROUP definition is used to tie several related FIELDs together. The GROUP definition has the following format:

```
Name    GROUP
        LENGTH "Expression"
        OPTIONAL "Condition"
        SUMMARIZE "Condition" : "FormatString" [
        "Condition" : "FormatString"... ]
        DESCRIPTION "Description"
        ::= { Name=FieldOrGroup [ ,
        Name=FieldOrGroup... ] }
```

Where only the GROUP and ::= lines are required. All the other lines are attribute lines, which define special characteristics for the GROUP. Attribute lines are optional and may appear in any order. Each attribute line is described in detail below:

3.2.1 LENGTH "Expression"

This attribute defines an expression for determining the GROUP's length. Expressions are arithmetic and can refer to the value of other FIELD's in the packet by adding a $ to the referenced field's name. For example, "($tcpHeaderLen *4) – 20" is a valid expression if tcpHeaderLen is another field defined for the current packet.

3.2.2 OPTIONAL "Condition"

This attribute defines a condition for determining whether a GROUP is present or not. Valid conditions are defined in the Conditions section below.

3.2.3 SUMMARIZE "Condition" : "FormatString" ["Condition" : "FormatString"...]

This attribute defines how a GROUP will be displayed in Detail mode. A different format (FormatString) can be specified for each condition (Condition). Valid conditions are defined in the Conditions section below. Any FIELD's value can be referenced within the FormatString by proceeding the FIELD's name with a $. In addition to FIELD names there are several other special $ keywords:

$LAYER	Displays the current protocol layer.
$GROUP	Displays the entire GROUP as a table.
$LABEL	Displays the GROUP label.
$*field*	Displays the *field* value (use enumerated name if available).
$:*field*	Displays the *field* value (in raw format).

3.2.4 DESCRIPTION "Description"

This attribute defines the description of the GROUP. It is used for informational purposes only.

3.2.5 ::= { Name=FieldOrGroup [, Name=FieldOrGroup...] }

This defines the order of the fields and subgroups within the GROUP.

3.3 PROTOCOL Definitions

The PROTOCOL definition is used to define the order of the FIELDs and GROUPs within the protocol header. The PROTOCOL definition has the following format:

```
Name    PROTOCOL
        SUMMARIZE "Condition" : "FormatString" [
        "Condition" : "FormatString"... ]
        DESCRIPTION "Description"
        REFERENCE "Reference"
        ::= { Name=FieldOrGroup [ ,
        Name=FieldOrGroup... ] }
```

Where only the PROTOCOL and ::= lines are required. All the other lines are attribute lines, which define special characteristics for the PROTOCOL. Attribute lines are optional and may appear in any order. Each attribute line is described in detail below:

3.3.1 SUMMARIZE "Condition" : "FormatString" ["Condition" : "FormatString"...]

This attribute defines how a PROTOCOL will be displayed in Summary mode. A different format (FormatString) can be specified for each condition (Condition). Valid conditions are defined in the Conditions section below. Any FIELD's value can be referenced within the FormatString by proceeding the FIELD's name with a $. In addition to FIELD names there are several other special $ keywords:

$LAYER	Displays the current protocol layer.
$VARBIND	Displays the entire SNMP VarBind list.
$*field*	Displays the *field* value (use enumerated name if available).
$:*field*	Displays the *field* value (in raw format).
$#*field*	Counts all occurrences of *field*.
$**field*	Lists all occurrences of *field*.

3.3.2 DESCRIPTION "Description"

This attribute defines the description of the PROTOCOL. It is used for informational purposes only.

3.3.3 REFERENCE "Reference"

This attribute defines the reference material used to determine the protocol format. It is used for informational purposes only.

3.3.4 ::= { Name=FieldOrGroup [, Name=FieldOrGroup...] }

This defines the order of the FIELDs and GROUPs within the PROTOCOL.

3.4 FLOW Definitions

The FLOW definition is used to define a network flow by describing where the address, protocol type, and port numbers are in a packet. The FLOW definition has the following format:

```
Name    FLOW
        HEADER { Option [, Option...] }
        DLC-LAYER { Option [, Option...] }
        NET-LAYER { Option [, Option...] }
        CONNECTION { Option [, Option...] }
        PAYLOAD { Option [, Option...] }
        CHILDREN { Option [, Option...] }
        STATE-BASED
        STATES "Definitions"
```

Where only the FLOW line is required. All the other lines are attribute lines, which define special characteristics for the FLOW. Attribute lines are optional and may appear in any order. However, at least one attribute line must be present. Each attribute line is described in detail below:

3.4.1 HEADER { Option [, Option...] }

This attribute is used to describe the length of the protocol header. The currently supported Options are:

LENGTH=*number*	Header is a fixed length of size *number*.
LENGTH=*field*	Header is variable length determined by value of *field*.
IN-WORDS	The units of the header length are in 32-bit words rather than bytes.

3.4.2 DLC-LAYER { Option [, Option...] }

If the protocol is a data link layer protocol, this attribute describes it. The currently supported Options are:

DESTINATION=*field*	Indicates which *field* is the DLC destination address.
SOURCE=*field*	Indicates which *field* is the DLC source address.
PROTOCOL	Indicates this is a data link layer protocol.
TUNNELING	Indicates this is a tunneling protocol.

3.4.3 NET-LAYER { Option [, Option...] }

If the protocol is a network layer protocol, then this attribute describes it. The currently supported Options are:

DESTINATION=*field*	Indicates which *field* is the network destination address.

SOURCE=*field*	Indicates which *field* is the network source address.
TUNNELING	Indicates this is a tunneling protocol.
FRAGMENTATION=*type*	Indicates this protocol supports fragmentation. There are currently two fragmentation types: IPV4 and IPV6.

3.4.4 CONNECTION { Option [, Option...] }

If the protocol is a connection-oriented protocol, then this attribute describes how connections are established and torn down. The currently supported Options are:

IDENTIFIER=*field*	Indicates the connection identifier *field.*
CONNECT-START="*flag*"	Indicates when a connection is being initiated.
CONNECT-COMPLETE="*flag*"	Indicates when a connection has been established.
DISCONNECT-START="*flag*"	Indicates when a connection is being torn down.
DISCONNECT-COMPLETE="*flag*"	Indicates when a connection has been torn down.
INHERITED	Indicates this is a connection-oriented protocol but the parent protocol is where the connection is established.

3.4.5 PAYLOAD { Option [, Option...] }

This attribute describes how much of the payload from a packet of this type should be stored for later use during analysis. The currently supported Options are:

INCLUDE-HEADER	Indicates that the protocol header should be included.
LENGTH=*number*	Indicates how many bytes of the payload should be stored.
DATA=*field*	Indicates which *field* contains the payload.

3.4.6 CHILDREN { Option [, Option...] }

This attribute describes how children protocols are determined. The currently supported Options are:

DESTINATION=*field*	Indicates which *field* is the destination port.
SOURCE=*field*	Indicates which *field* is the source port.
LLCCHECK=*flow*	Indicates that if the DESTINATION field is less than 0x05DC then use *flow* instead of the current flow definition.

3.4.7 STATE-BASED

This attribute indicates that the flow is a state-based flow.

3.4.8 STATES "Definitions"

This attribute describes how children flows of this protocol are determined using states. See the State Definitions section below for how these states are defined.

3.5 CONDITIONS

Conditions are used with the OPTIONAL and SUMMARIZE attributes and may consist of the following:

Value1 == Value2	Value1 equals Value2. Works with string values.
Value1 != Value2	Value1 does not equal Value2. Works with string values.
Value1 <= Value2	Value1 is less than or equal to Value2.
Value1 >= Value2	Value1 is greater than or equal to Value2.
Value1 < Value2	Value1 is less than Value2.
Value1 > Value2	Value1 is greater than Value2.
Field m/regex/	Field matches the regular expression regex.

Where *Value1* and *Value2* can be either FIELD references (field names preceded by a $) or constant values. Note that compound conditional statements (using AND and OR) are not currently supported.

3.6 STATE DEFINITIONS

Many applications running over data networks utilize complex methods of classifying traffic through the use of multiple states. State definitions are used for managing and maintaining learned states from traffic derived from the network.

The basic format of a state definition is:

```
StateName:  Operand Parameters [ Operand Parameters...]
```

The various states of a particular flow are described using the following operands:

3.6.1 CHECKCONNECT, *operand*

Checks for connection. Once connected executes *operand.*

3.6.2 GOTO *state*

Goes to *state*, using the current packet.

3.6.3 NEXT *state*

Goes to *state*, using the next packet.

3.6.4 DEFAULT *operand*

Executes *operand* when all other operands fail.

3.6.5 CHILD *protocol*

Jump to child *protocol* and perform state-based processing (if any) in the child.

3.6.6 WAIT *numPackets, operand1, operand2*

Waits the specified number of packets. Executes *operand1* when the specified number of packets have been received. Executes *operand2* when a packet is received but it is less than the number of specified packets.

3.6.7 MATCH *'string' weight offset LF-offset range LF-range, operand*

Searches for a *string* in the packet, executes *operand* if found.

3.6.8 CONSTANT *number offset range, operand*

Checks for a constant in a packet, executes *operand* if found.

3.6.9 EXTRACTIP *offset destination, operand*

Extracts an IP address from the packet and then executes *operand.*

3.6.10 EXTRACTPORT *offset destination, operand*

Extracts a port number from the packet and then executes *operand.*

3.6.11 CREATEREDIRECTEDFLOW, *operand*

Creates a redirected flow and then executes *operand*.

4. Example PDL Rules

The following section contains several examples of PDL Rule files.

4.1 Ethernet

The following is an example of the PDL for Ethernet:

```
MacAddress FIELD
           SYNTAX          BYTESTRING(6)
           DISPLAY-HINT    "1x:"
           LOOKUP          MACADDRESS
           DESCRIPTION
                "MAC layer physical address"

etherType  FIELD
           SYNTAX          INT(16)
           DISPLAY-HINT    "1x:"
           LOOKUP          FILE "EtherType.cf"
           DESCRIPTION
                "Ethernet type field"

etherData  FIELD
           SYNTAX          BYTESTRING(46..1500)
           ENCAP           etherType
           DISPLAY-HINT    "HexDump"
           DESCRIPTION
               "Ethernet data"

ethernet   PROTOCOL
           DESCRIPTION
               "Protocol format for an Ethernet frame"
           REFERENCE  "RFC 894"
::= { MacDest=macAddress, MacSrc=macAddress,
EtherType=etherType,
     Data=etherData }

ethernet   FLOW
           HEADER { LENGTH=14 }
           DLC-LAYER {
               SOURCE=MacSrc,
               DESTINATION=MacDest,
               TUNNELING,
               PROTOCOL
           }
           CHILDREN { DESTINATION=EtherType, LLC-CHECK=llc }
```

4.2 IP Version 4

Here is an example of the PDL for the IP protocol:

```
ipAddress  FIELD
           SYNTAX          BYTESTRING(4)
           DISPLAY-HINT    "1d."
           LOOKUP          HOSTNAME
           DESCRIPTION
               "IP address"

ipVersion  FIELD
           SYNTAX     INT(4)
           DEFAULT    "4"

ipHeaderLength  FIELD
                SYNTAX     INT(4)

ipTypeOfService FIELD
                SYNTAX     BITSTRING(8) { minCost(1),
                           maxReliability(2), maxThruput(3),
                           minDelay(4) }

ipLength        FIELD
                SYNTAX     UNSIGNED INT(16)

ipFlags         FIELD
                SYNTAX     BITSTRING(3) { moreFrags(0),
dontFrag(1) }

IpFragmentOffset     FIELD
                     SYNTAX     INT(13)

ipProtocol FIELD
           SYNTAX     INT(8)
           LOOKUP     FILE "IpProtocol.cf"

ipData     FIELD
           SYNTAX          BYTESTRING(0..1500)
           ENCAP           ipProtocol
           DISPLAY-HINT    "HexDump"

ip    PROTOCOL
      SUMMARIZE
      "$FragmentOffset != 0":
           "IPFragment ID=$Identification
      Offset=$FragmentOffset"
      "Default" :
           "IP Protocol=$Protocol"
      DESCRIPTION
          "Protocol format for the Internet Protocol"
      REFERENCE  "RFC 791"
::= { Version=ipVersion, HeaderLength=ipHeaderLength,
      TypeOfService=ipTypeOfService, Length=ipLength,
```

```
     Identification=UInt16, IpFlags=ipFlags,
     FragmentOffset=ipFragmentOffset, TimeToLive=Int8,
     Protocol=ipProtocol, Checksum=ByteStr2,
     IpSrc=ipAddress, IpDest=ipAddress, Options=ipOptions,
     Fragment=ipFragment, Data=ipData }

ip   FLOW
     HEADER { LENGTH=HeaderLength, IN-WORDS }
     NET-LAYER {
         SOURCE=IpSrc,
         DESTINATION=IpDest,
         FRAGMENTATION=IPV4,
         TUNNELING
     }
     CHILDREN { DESTINATION=Protocol }

ipFragData FIELD
          SYNTAX          BYTESTRING(1..1500)
          LENGTH          "ipLength - ipHeaderLength * 4"
          DISPLAY-HINT    "HexDump"

ipFragment GROUP
          OPTIONAL   "$FragmentOffset != 0"
::= { Data=ipFragData }

ipOptionCode     FIELD
                 SYNTAX     INT(8) { ipRR(0x07),
ipTimestamp(0x44),
                            ipLSRR(0x83), ipSSRR(0x89) }
                 DESCRIPTION
                     "IP option code"

ipOptionLength   FIELD
                 SYNTAX     UNSIGNED INT(8)
                 DESCRIPTION
                     "Length of IP option"

ipOptionData     FIELD
                 SYNTAX          BYTESTRING(0..1500)
                 ENCAP           ipOptionCode
                 DISPLAY-HINT    "HexDump"

ipOptions  GROUP
          LENGTH          "(ipHeaderLength * 4) - 20"
::= { Code=ipOptionCode, Length=ipOptionLength, Pointer=UInt8,
     Data=ipOptionData }
```

4.3 TCP

Here is an example of the PDL for the TCP protocol:

```
tcpPort FIELD
      SYNTAX     UNSIGNED INT(16)
      LOOKUP     FILE "TcpPort.cf"

tcpHeaderLen FIELD
      SYNTAX     INT(4)

tcpFlags FIELD
      SYNTAX     BITSTRING(12) { fin(0), syn(1), rst(2), psh(3),
                                 ack(4), urg(5) }

tcpData FIELD
      SYNTAX     BYTESTRING(0..1564)
      LENGTH     "($ipLength-($ipHeaderLength*4))-
($tcpHeaderLen*4)"
      ENCAP      tcpPort
      DISPLAY-HINT    "HexDump"

tcp   PROTOCOL
      SUMMARIZE
          "Default" :
            "TCP ACK=$Ack WIN=$WindowSize"
      DESCRIPTION
          "Protocol format for the Transmission Control
Protocol"
      REFERENCE  "RFC 793"
::= { SrcPort=tcpPort, DestPort=tcpPort, SequenceNum=UInt32,
      Ack=UInt32, HeaderLength=tcpHeaderLen, TcpFlags=tcpFlags,
      WindowSize=UInt16, Checksum=ByteStr2,
      UrgentPointer=UInt16, Options=tcpOptions, Data=tcpData }

tcp   FLOW
      HEADER { LENGTH=HeaderLength, IN-WORDS }
      CONNECTION {
           IDENTIFIER=SequenceNum,
           CONNECT-START="TcpFlags:1",
           CONNEGT-COMPLETE="TcpFlags:4",
           DISCONNECT-START="TcpFlags:0",
           DISCONNECT-COMPLETE="TcpFlags:4"
       }
      PAYLOAD { INCLUDE-HEADER }
      CHILDREN { DESTINATION=DestPort, SOURCE=SrcPort }

tcpOptionKind   FIELD
           SYNTAX     UNSIGNED INT(8) { tcpOptEnd(0), tcpNop(1),
                            tcpMSS(2), tcpWscale(3),
                            tcpTimestamp(4) }
           DESCRIPTION
               "Type of TCP option"
```

```
tcpOptionData   FIELD
          SYNTAX     BYTESTRING(0..1500)
          ENCAP      tcpOptionKind
          FLAGS      SAMELAYER
          DISPLAY-HINT    "HexDump"

tcpOptions GROUP
          LENGTH          "($tcpHeaderLen * 4) - 20"
::= { Option=tcpOptionKind, OptionLength=UInt8,
      OptionData=tcpOptionData }

tcpMSS    PROTOCOL
::= { MaxSegmentSize=UInt16 }
```

4.4 HTTP (with State)

Here is an example of the PDL for the HTTP protocol:

```
httpData FIELD
   SYNTAX  BYTESTRING(1..1500)
   LENGTH  "($ipLength - ($ipHeaderLength * 4)) - ($tcpHeaderLen
* 4)"
   DISPLAY-HINT "Text"
   FLAGS        NOLABEL

http    PROTOCOL
        SUMMARIZE
            "$httpData m/^GET|^HTTP|^HEAD|^POST/" :
                "HTTP $httpData"
            "$httpData m/^[Dd]ate|^[Ss]erver|^[Ll]ast-
[Mm]odified/" :
                "HTTP $httpData"
            "$httpData m/^[Cc]ontent-/" :
                "HTTP $httpData"
            "$httpData m/^<HTML>/" :
                "HTTP [HTML document]"
            "$httpData m/^GIF/" :
                "HTTP [GIF image]"
            "Default" :
                "HTTP [Data]"
        DESCRIPTION
            "Protocol format for HTTP."
::= { Data=httpData }

http FLOW
     HEADER { LENGTH=0 }
     CONNECTION { INHERITED }
     PAYLOAD { INCLUDE-HEADER, DATA=Data, LENGTH=256 }
     STATES
           "S0: CHECKCONNECT, GOTO S1
                DEFAULT NEXT S0

            S1: WAIT 2, GOTO S2, NEXT S1
                DEFAULT NEXT S0

            S2: MATCH
                  '\n\r\n'           900 0 0 255 0, NEXT S3
                  '\n\n'             900 0 0 255 0, NEXT S3
                  'POST /tds?'        50 0 0 127 1, CHILD
sybaseWebsql
                  '.hts HTTP/1.0'     50 4 0 127 1, CHILD
sybaseJdbc
                  'jdbc:sybase:Tds'   50 4 0 127 1, CHILD
sybaseTds
                  'PCN-The Poin'     500 4 1 255 0, CHILD
pointcast
                  't: BW-C-'         100 4 1 255 0, CHILD backweb
                  DEFAULT NEXT S3
```

```
          S3: MATCH
                '\n\r\n'           50 0 0    0 0, NEXT S3
                '\n\n'             50 0 0    0 0, NEXT S3
                'Content-Type:'   800 0 0 255 0, CHILD mime
                'PCN-The Poin'    500 4 1 255 0, CHILD
pointcast
                't: BW-C-'        100 4 1 255 0, CHILD backweb
                DEFAULT NEXT S0"

sybaseWebsql   FLOW
               STATE-BASED

sybaseJdbc     FLOW
               STATE-BASED

sybaseTds      FLOW
               STATE-BASED

pointcast      FLOW
               STATE-BASED

backweb        FLOW
               STATE-BASED

mime           FLOW
               STATE-BASED
               STATES
                 "S0:   MATCH
                   'application' 900 0 0   1 0, CHILD
                mimeApplication
                   'audio'       900 0 0   1 0, CHILD mimeAudio
                   'image'        50 0 0   1 0, CHILD mimeImage
                   'text'         50 0 0   1 0, CHILD mimeText
                   'video'        50 0 0   1 0, CHILD mimeVideo
                   'x-world'     500 4 1 255 0, CHILD
              mimeXworld
                DEFAULT GOTO S0"

mimeApplication FLOW
                STATE-BASED

mimeAudio   FLOW
            STATE-BASED
            STATES
             "S0: MATCH
                  'basic'              100 0 0 1 0, CHILD
pdBasicAudio
                  'midi'               100 0 0 1 0, CHILD pdMidi
                  'mpeg'               100 0 0 1 0, CHILD
pdMpeg2Audio
                  'vnd.rn-realaudio'   100 0 0 1 0, CHILD
pdRealAudio
                  'wav'                100 0 0 1 0, CHILD pdWav
                  'x-aiff'             100 0 0 1 0, CHILD pdAiff
```

```
                    'x-midi'               100 0 0 1 0, CHILD pdMidi
                    'x-mpeg'               100 0 0 1 0, CHILD
pdMpeg2Audio
                    'x-mpgurl'             100 0 0 1 0, CHILD
pdMpeg3Audio
                    'x-pn-realaudio'       100 0 0 1 0, CHILD
pdRealAudio
                    'x-wav'                100 0 0 1 0, CHILD pdWav
                   DEFAULT GOTO S0"

mimeImage   FLOW
            STATE-BASED

mimeText    FLOW
            STATE-BASED

mimeVideo   FLOW
            STATE-BASED

mimeXworld FLOW
            STATE-BASED

pdBasicAudio FLOW
           STATE-BASED

pdMidi        FLOW
           STATE-BASED

pdMpeg2Audio FLOW
           STATE-BASED

pdMpeg3Audio FLOW
           STATE-BASED

pdRealAudio  FLOW
           STATE-BASED

pdWav         FLOW
           STATE-BASED

pdAiff        FLOW
           STATE-BASED
```

5. Supplemental Products

MeterWorks is supported by an optional Simple Network Management Protocol (SNMP) implementation. Envoy simplifies the process of porting *MeterWorks* to any platform. The *MeterWorks* SNMP Method Routines are written to directly interoperate with the Envoy product.

In addition, Envoy is supported by Emissary, Epilogue's optional MIB Compiler product. The MIB Compiler is a tool that greatly simplifies the creation and maintenance of proprietary MIB extensions. *MeterWorks* also takes direct advantage of the Emissary MIB compiler for making changes in the RMON groups that are supported.

Attaché, Epilogue's Portable UDP/IP protocol stack, complements *MeterWorks* and Envoy in environments that do not already provide a protocol stack for use by SNMP. Attaché version 3.0 provides full integration with *MeterWorks* version 4.00 and Envoy version 5.2, and fully implements MIB II (RFC 1213).

Index

- **Exhibit B9:** State-based Sub-Classification Overview (document MFS-State-Classification.pdf) that describes the states of some of the protocols that are supported.

MeterFlow™
Traffic Classification System

State-based Sub-Classification Overview

Version A0.03

[redacted]

DRAFT - 03

Technically Elite
monitoring your enterprise

NOTICE

Document revision A.03-004, format revision 1.00-000.

Technically Elite, Inc.
6330 San Ignacio Ave.
San Jose, CA 95119-1209, USA
Phone: +1.408.574.2300
Fax: +1.408.629.8300
E-mail: `mworks@tecelite.com`
URL: `http://www.tecelite.com`

Contents

1. Introduction

MeterFlow allows for a very rich set of protocol classification and sub-classification in the process of analyzing and interpreting Network Traffic. MeterFlow accomplishes this by combining the maintenance of state information with a robust ability to interpret network data streams.

Without the ability to maintain state, an increasingly large amount of Network Traffic will be misclassified, partially classified, or not classified at all with present traffic analysis and interpretation technologies. Pattern matching parser techniques employed in many such contemporary technologies provide little help here given the growing complexity of today's Network Traffic.

Misclassification is frequent given the practice of using assigned (or otherwise well-known) port/socket numbers as ephemeral ports/sockets. This has become especially noteworthy with the increasing proliferation of Web Browsers and MS WinSock. For example, BackWeb push-technology and Streamworks or VDOLive multimedia clients can use UDP ports that are either assigned to or used as defacto standards by other Network Applications such as Citrix, H.323 Gatekeeper, RealAudio, etc.

Partial classification is a common limitation in traffic analysis when the scope of interpretation is limited to a single packet. For example, one could see TCP Port #1527 referenced in a network packet and know that is was an Oracle TNS Packet. Without having interpreted the initial Oracle TNS protocol exchange spanning multiple packets, one could not have known that it was indeed PeopleSoft running over SQL*Net running over Oracle TNS.

Another example is of partial classification is simple "IP Fragmentation". Decoding the first fragment of an IP Datagram could easily determine that it further contained NFS over SunRPC over UDP. However, since subsequent fragments do not contain the UDP or SunRPC headers, they cannot be sub-classified for these protocols without having retained state and decoding information from the original (or first) fragment.

The inability to classify is becoming increasingly common as Network Applications use dynamic mechanisms to allocate and assign resources to various applications. There are a number of ways this can happen.

- In many cases, connections are established on a "truly" well-known port/socket of a server. The exchange on this connection serves to negotiate services requested/available and the address/port at which those services can be accessed. A second connection on the allocated/assigned address and port (almost always ephemeral) carries the bulk or volume of the data in the overall Network Session. Without the ability to interpret and analyze "data" in such allocation/assignment protocols connections, the volume traffic on the secondary connections cannot be distinguished from any other "un-interpretable" traffic. Microsoft's Endpoint-Mapper, SunRPC's Portmapper, and Oracle TNS are examples of such protocols.

- In other cases, available services and their locations (addresses and ports/sockets) are periodically announced. Without having interpreted and remembered the content of such announcements, traffic to/from them cannot be classified. Novell SAP and Apple's Name Binding Protocol (NBP) are examples of such announcement-based approaches.

The art of traffic classification becomes further complicated when a multitude of the underlying challenges described above occurs for the same Network Data events. For example, NFS version 1 is transferring one of its typical 32-Kbyte blocks of data in a single IP Datagram and is hence fragmenting it (partial classification scenario). This transfer is occurring on an "ephemeral" UDP port of the server that was allocated via an initial exchange with the SunRPC Portmapper protocol (no classification scenario). Or, even worse, the "ephemeral" UDP port on the server turns out to be the same as one of the defacto standard UDP ports that "RealAudio" uses (mis-classification scenario).

MeterFlow surmounts these challenges to provide accurate and thorough network traffic classification. This document discusses the currently supported and in-progress traffic classification capabilities of MeterFlow. It also presents the how MeterFlow may be extended to support further sub-classifications.

2. Protocol Classification

2.1 Current Protocol Classifications

MeterFlow currently includes support for classification and sub-classification of the following protocols, each of which are described in more detail in Section 3.0. In-progress developments to enhance or extend the sub-classification capabilities for these protocols are also described in Section 3.0.

1. IP Fragmentation
2. Microsoft Endpoint-Mapper
3. SunRPC Portmapper
4. Oracle 6/7 Transparent Network Substrate (TNS)
5. H.323 Video Conferencing
6. HTTP
7. BackWeb

2.2 In-Progress Protocol Classifications

Several new state-based protocol classification and sub-classification capabilities are under development. These protocols include the following and are described further in Section 4.0.

1. Real-Time Streaming Protocol (RTSP)
2. Novell Service Advertising Protocol (SAP)
3. MS Media
4. Streamworks and VDOLive

3. Current Protocol Classification

3.1 IP/IPIP/IPIP4 Fragmentation

3.1.1 Features

Fragmentation support for the Internet IP protocol is implemented in three MeterFlow sub-engines, each of which supports state maintenance and sub-classification retention for network packet fragments associated with the following protocols:

1. IP - Internet Protocol Version 4 datagram fragments
2. IPIP - IPIP datagram fragments Tunneled over IP
3. IPIP4 - IPIP4 datagram fragments Tunneled over IP

Key capabilities of these sub-engines include:

1. Tracking fragments for their corresponding protocols
2. Passing on 1st fragments through normal decoding and state-based decoding
3. Retaining complete 1st fragment sub-classification information for datagrams which are not further classified as state based (e.g. NFS Version 2 over UDP on well-known port 2049) and applying this information to all subsequent fragments components.
4. Retaining flow references for 1st fragment sub-classifications that further classify as state-based (e.g. Oracle TNS over TCP on a redirected, ephemeral port) and updating such flows for all subsequent fragment components.
5. Supporting concurrent fragmentation of data across multiple layers of Tunneling (e.g. IPIP4 fragments contained in IP fragments).

3.1.2 Sub-classifications

These sub-engines don't really "classify" or "sub-classify" underlying protocols contained in fragments beyond that normally done by the standard IP Version 4 decoding of the "protocol type". They do however retain "sub-classification" information or flow references as described above in Section 3.1.1.

3.1.3 Extensibility

By the nature of their scope, these sub-engines are not extensible beyond that which may be applied to standard IP Version 4 decoding with respect to the addition of new "Protocol Types".

3.1.4 Planned Developments

The "Internet Fragmentation Sub-Engines" will eventually add support for IP Version 6.

3.2 Microsoft Endpoint-Mapper

3.2.1 Features

The Microsoft Endpoint-Mapper **actually** supports the Endpoint-Mapper protocol defined by the "*Distributed Computing Environment (DCE) 1.1 – Remote Procedure Call*" specification. The key node point in the protocol directory for this protocol, and related applications determined by its mappings, is "*endpoint-mapper*".

Key capabilities of this sub-engine include:

1. Tracking connections to and exchange within the well-known Endpoint-Mapper.
2. Distinguishing such "mapping" traffic from traffic on application connections subsequently "mapped".
3. Detecting assignments of server application access assignments to various hosts and/or ports and creating sub-classifications for these access points.
4. When traffic to these access points is seen, it will be classified
   a) By the appropriate application under "*endpoint-mapper*", if the server application identifier in the mapping exchange is a **known** sub-application.
   b) Minimally as "*endpoint-mapper*", if the server application is unknown.
5. Allowing **known** sub-applications to be specified with respect to flow reporting with two levels of identification
   a) Level 1 – Endpoint Mapped "Application Group"
   b) Level 2 – Sub-application within the Application Group
6. Supporting the "connection-oriented" mode of Endpoint-Mapper operations.

3.2.2 Sub-classifications

Sub-classifications under "endpoint-mapper" include the following in both the "*tcp*" and "*udp*" protocol subtrees:

endpoint-mapper → dcerpc-mapper *(DCE RPC – Endpoint Mapping)*
→ ms-exchange → directory *(MS-Exchange Directory)*
→ information-store *(MS-Exhange Information Store)*
→ mta *(MS-Exchange MS-Mail MTA)*

3.2.3 Extensibility

New Sub-classifications are easily added as new entries in the DCE RPC Sub-Engine's "Sub-Protocol Info" table, if the Universally Unique IDs (UUIDs) of the corresponding applications are known.

3.2.4 Planned Developments

Certainly there are more applications out there other than MS-EXCHANGE using DCE-RPC (also known as MS-RPC or Microsoft RPC since Microsoft adopted this RPC standard as opposed to SunRPC). As more notable applications are identified along with their assigned UUIDs, they will be added to the MeterFlow implementation.

Microsoft Exchange under the UDP instance of "endpoint-mapper" probably isn't really a valid possibility. Accordingly, it will probably be removed from this instance.

Support for the "connection-less" mode of Endpoint Mapper operation could become a candidate for implementation when and if it can be determined that someone is indeed using it in the real world.

3.3 SunRPC Portmapper

3.3.1 Features

The SunRPC Portmapper protocol is defined by the "RPC: Remote Procedure Call Specification Version 2 (RFC 1831)" standard. The key node point in the protocol directory for this protocol, and related applications determined by its mappings, is "sunrpc".

Key capabilities of this sub-engine include:

1. Tracking exchanges with the well-known SunRPC Portmapper.
2. Distinguishing such "mapping" traffic from traffic on application connections subsequently "mapped".
3. Detecting assignments of server application access assignments to various hosts and/or ports and creating sub-classifications for these access points.
4. When traffic to these access points is seen, it will be classified
   (a) By the appropriate application under "*sunrpc*", if the server application identifier in the mapping exchange is a **known** sub-application.
   (b) Minimally as "*sunrpc*", if the server application is unknown.
5. Allowing known sub-applications to be specified with respect to flow reporting with a single levels of identification
   (a) Level 1 – Portmapped "Application"

3.3.2 Sub-classifications

Sub-classifications under "sunrpc" include the following in both the "tcp" and "udp" protocol subtrees:

sunrpc	→ *portmapper*	*(SunRPC – Port Mapping)*
	→ *rstat*	*(remote statistics)*
	→ *nfs*	*(network file service)*
	→ *ypserv*	*(yellow pages – server)*
	→ *ypbind*	*(yellow pages – bindings)*
	→ *ypupdated*	*(yellow pages – update daemon)*
	→ *ypxferd*	*(yellow pages – transfer daemon)*
	→ *mount*	*(remote file system mount)*
	→ *3270-mapper*	*(3270 terminal session mapper)*
	→ *rje-mapper*	*(remote job entry session mapper)*
	→ *nis*	*(next generation yellow pages)*
	→ *pcnfsd*	*(pcNFS daemon)*

3.3.3 Extensibility

New Sub-classifications are easily added as new entries in the SunRPC Sub-Engine's "Sub-Protocol Info" table, if the SunRPC Program Number of the corresponding applications are known.

3.3.4 Planned Developments

Certainly there are still other applications using SunRPC. As more notable applications are identified along with their assigned SunRPC Program Numbers, they will be added to the MeterFlow implementation.

Enhancement of the SunRPC Sub-Engine to additionally support SET, UNSET, DUMP, and/or CALLIT SunRPC Portmapper primitives could become a candidate for implementation when and if it can be determined that someone is indeed using them in the real world.

Improved understanding of whether the supported SunRPC sub-applications run over just UDP or TCP will enable the "sunrpc" sub-classifications to be more accurately categorized with respect to the protocol it actually runs over. For example, "portmapper", "nfs", and "pcnfsd" all operate only over UDP. Accordingly, their presence in the TCP subtree for "sunrpc" is unnecessary.

3.4 Oracle 6/7 Transparent Network Substrate (TNS)

3.4.1 Features

The Transparent Network Substrate (TNS) protocol is defined by Oracle Corporation and is used as the underlying networks access framework for its Oracle Version 6 and Oracle Version 7 database product offerings. The key node points in the protocol directory for this protocol and applications determined by its mappings are "oracl-tns", "oracl-tns2", "oracl-tns-srv". These three node points reflect the three different "well-known" ports that serve to support initial access to Oracle TNS on Oracle Database servers. The first is a defacto, Oracle standard use. The next two access points (TCP ports) are assigned to Oracle by IANA.

Key capabilities of this sub-engine include:

1. Tracking connections to and exchanges in well-known Oracle TNS port traffic.
2. Learning the client application attempting to access the Oracle Database (e.g. PeopleSoft, Oracle Forms, etc.) to further classify traffic on the well-known Oracle TNS connections.
3. Detecting "redirections" of connections to various hosts and/or ports and creating sub-classifications for these access points. Such "redirections" inherit the sub-classifications of the initial connections to the well-known Oracle TNS service.
4. When traffic to these access points is seen or when TNS sessions are "accepted" on the well-know TNS service port, it will be classified

   (a) By the appropriate client application under "*oracle-tns*" (or "*oracl-tns2*" or "*oracl-tns-srv*), if the client application identifier is a **known** sub-application.
   (b) Minimally as "*oracle-tns*" (or "*oracl-tns2*" or "*oracl-tns-srv*), if the server application is unknown.

5. Allowing known sub-applications to be specified with respect to flow reporting with two levels of identification

   (a) Level 1 – Oracle client's "Application Group"
   (b) Level 2 – Sub-application within the Application Group

3.4.2 Sub-classifications

Sub-classifications under "oracle-tns" include the following in the "tcp" subtree. Note that the same sub-classification occurs under the "oracl-tns2" and "oracl-tns-srv" nodes as well.

oracle-tns	→ *ms-odbc*	*(Microsoft ODBC)*
	→ *ms-ole*	*(Microsoft OLE)*
	→ *oracle-sqlplus*	*(Oracle SQLPlus)*
	→ *oracle-forms*	*(Oracle FORMS)*
	→ *peoplesoft*	*(PeopleSoft)*

3.4.3 Extensibility

New Sub-classifications are easily added as new entries in the Oracle TNS Sub-Engine's "Sub-Protocol Info" table, if the Program Names (or names of the client programs' executables) of the corresponding client applications are known.

3.4.4 Planned Developments

Further sub-classification of "PeopleSoft" is highly desired. Namely, breaking "peoplesoft" down into component applications.

Certainly there are still other native, client applications using Oracle TNS. As more notable applications are identified along with their assigned Program/Executable Names, they will be added to the MeterFlow implementation.

"SAP R/3" and "Baan", in particular, are high priority applications to establish such additional support for.

A major enhancement to the Oracle TNS sub-engine will be to further build upon the application sub-classification capabilities presently supported. This will allow the sub-engine to further delve into the SQL*Net content to determine the actual client applications riding atop 4GL tools (such as Oracle FORMs) and access APIs (such as MS ODBC, and MS OLE).

The three Oracle TNS subtrees ("oracle-tns", "oracl-tns2", and "oracl-tns-srv) will most likely be consolidated under a single subtree under TCP ("oracle-tns").

3.5 H.323 Videoconferencing

3.5.1 Features

H.323 is an umbrella standard published by the International Telecommunication Union (ITU, formerly CCITT) for videoconferencing. H.323 entails one of the most complicated traffic classification challenges of today's networking protocols. This arises form its inherent multi-tier connection/data-stream architecture.

In H.323, connections are initially established on a well-known service port. The Q.931 protocol is used on this "H.323 Call Setup" connection to set-up a second connection on an ephemeral port. The second "H.323 Call Control" connection uses the H.245 protocol to negotiate audio and video capabilities (codecs) as well as to further set-up RTP/RTCP audio and video data streams over ephemeral UDP ports.

Key capabilities of this sub-engine include:

1. Tracking connections to and exchanges on well-known H.323-host-call port (Q.931 protocol) traffic.
2. Detecting assignments of H.245 access points to various hosts and/or ports and creating H.245 sub-classifications for these access points.
3. Tracking connections to and exchanges with such assigned H.245 access points.
4. Detecting the assignment of RTP/RTCP audio and video, UDP datastreams access points as well as the audio and video "codecs" negotiated for use on them and creating RTP/RTCP sub-classifications for these access points.
5. When traffic to these RTP/RTCP access points are seen it will be classified
   (a) By the appropriate "codec" under "*rtp*", if the negotiated codec is a **known** audio/video stream type.
   (b) Minimally as "*rtp*", if the negotiated codec is unknown.
6. Allowing known sub-applications (audio/video datastreams) to be specified with respect to flow reporting with three levels of identification
   (a) Level 1 – Datastream Class (e.g. audio, video, other…)
   (b) Level 2 – Datastream Type within the Datastream Class
   (c) Level 3 – Datastream Sub-Type within the Datastream Type
7. Supporting the Q.931 "normal mode" of operation for "H.323 Call Setup connections".

3.5.2 Sub-classifications

"H.323 Call Setup" Sub-classifications under "h323-host-call" include the following in the "tcp" subtree.

h323-host-call	→ *q931*	*(H.323 Call Setup)*
	→ *q931-fast-start*	*(H.323 Combined Setup and Control)*

"H.323 Call Control" Sub-classifications under "h323-host-control" include the following in the "tcp" subtree.

h323-host-control	→ *h245*	*(H.323 Call Control)*

Audio and video datastream Sub-classifications under "RTP/RTCP" include the following in the "udp" subtree.

RTCP →				*(Audio/Video Stream Control sub-channel)*
RTP	→ *audio*	→ *G.711*		*(Audio Transfer sub-channel)*
		→ *G.722*		
		→ *G.728*		
		→ *G.729*		
		→ *MPEG1-audio*		
		→ *G.723*		
		→ *GSM*		
	→ *video*	→ *H.261*	→ *QCIF*	*(Video Transfer sub-channel)*
			→ *CIF*	
		→ *H.263*	→ *SQCIF*	
			→ *QCIF*	
			→ *CIF*	
			→ *4CIF*	
			→ *16CIF*	
		→ *MRV*		

Standards for the audio stream sub-classifications indicated above are:

G.711 - 64 Kbps, 8K samples/sec, 8-bit companded PCM (A-law or μ -law), high quality, low complexity. Required for H.320 and H.323.

G.722 - ADPCM audio encode/decode (64 kbit/s, 7 kHz) .

G.723 - Speech coder at 6.3 and 5.3 Kbps data rate. Medium complexity. Required for H.324; Optional for H.323.

G.728 - 16 Kbps, LD-CELP, high quality speech coder, very high complexity. Optional for H.320 and H.323.

G.729 - 8Kbps, LD-CELP, high quality speech coder, medium complexity. G.DSVD is an interoperable subset.

GSM - Group Special Mobile -- European telephony standard, not ITU. Used by ProShare Video Conferencing software versions 1.0-1.8. 13Kbps, medium quality for voice only, low complexity.

Standards for the **video** stream sub-classifications indicated above are:

H.261 - Supports 352x288 (CIF or FCIF) and 176x144 (QCIF). DCT-based algorithm tuned for 2B to 6B ISDN communication. Required for H.320, H.323, and H.324.

H.263 - Much-improved derivative of H.261, tuned for POTS data rates. Mostly aimed at QCIF and Sub-QCIF (128x96 -- SQCIF). Optional for H.323 and H.324, although industry is focusing on it for POTS. Being added as an option to H.261.

MRV - Intel Indeo® video compression technology tuned for ISDN and LAN data rates.

3.5.3 Extensibility

New Sub-classifications are easily added as new entries in the H.323 Sub-Engine's "Sub-Protocol Info" table, if the Audio/Video Capability Identifiers of the corresponding audio/video datastream are known.

3.5.4 Planned Developments

There are still more audio/video datastream formats. As others are identified along with their assigned capability identifiers, they will be added to the MeterFlow implementation.

There is a mode of H.323 operation defined called "Q.931 Fast Start". In this mode, "H.323 Call Control" operations (normally performed under their own H.245 connection) are piggybacked over Q.931 in the "H.323 Call Setup" connection. The use of this mode of operation has historically been rare and infrequent in contemporary videoconferencing products. The H.323 sub-engine will be enhanced to support this mode of operation.

3.6 HTTP

3.6.1 Features

The HTTP Protocol is the basis of common, present-day Web Browsers and has become a fundamental transport mechanism for many Internet applications. HTTP operates over TCP connections. Traditional/typical use of HTTP involves the establishment/tear-down of an individual HTTP connection for each element of exchange in a given user session activity (e.g. a web page will involve many TCP connections to effect the transfer of the various components of the activity).

There are two ways to distinguish the nature of the application information involved in an HTTP exchange.

1. HTTP content type
2. Interpretation of various fields in the HTTP data

Key capabilities of this sub-engine include:

1. Tracking connections to and exchanges in well-known HTTP Port traffic.
2. Learning the nature of the application data being transferred or accessed to further classify traffic on such well-known HTTP connections.
3. Learning the nature of the application by virtue of analyzing selected HTTP fields.
4. Allowing known sub-applications to be specified with respect to flow reporting with two levels of identification
    (a) Level 1 – HTTP sub-application group (e.g. database, application, video, etc…)
    (b) Level 2 – sub-application within the sub-application group
5. Classifying HTTP traffic
    (a) By the appropriate sub-application within the sub-application group, if the sub-application identifier is **known**.
    (b) Minimally by the sub-application group, if the negotiated sub-application identifier is unknown.

3.6.2 Sub-classifications

Sub-classifications under "*www-http*" include the following in the "*tcp*" subtree. Note that the same sub-classification occurs under the "*alternate-http*" node as well.

www-http → *database* → *sybase-web-sql*
→ *sybase-tunneled-tds*

→ *jdbc* → *odbc-bridge*
→ *ibm-db2*
→ *gupta-jdbc*
→ *sybase-jdbc*

→ *application* → *pointcast*
→ *backweb*
→ *datawindow*
→ *edi-content*
→ *edi-x12*
→ *edifact*
→ *excel*
→ *macbinhex40*
→ *mp3*
→ *mspowerpoint*
→ *msword*
→ *news-message-id*
→ *news-transmission*
→ *octet-stream*
→ *oda*
→ *pdf*
→ *postscript*
→ *powerbuilder*
→ *quattro-pro*
→ *rtf*
→ *sgml*
→ *vnd-framemaker*
→ *vnd-lotus-1-2-3*
→ *vnd-lotus-approach*
→ *vnd-lotus-freelance*
→ *vnd-lotus-organizer*
→ *vnd-lotus-wordpro*
→ *vnd-mif*
→ *vnd-ms-excel*
→ *vnd-ms-powerpoint*
→ *vnd-ms-project*
→ *vnd-ms-word*
→ *vnd-werbuilder*
→ *vnd-rn-realplyer*
→ *vnd-visio*
→ *wordperfect*
→ *x-bcpio*
→ *x-compress*
→ *x-cpio*
→ *x-csh*
→ *x-director*
→ *x-dvi*
→ *x-gtar*
→ *x-gzip*
→ *x-javascript*
→ *x-latex*
→ *x-lotus-notes*
→ *x-macbinary*
→ *x-mif*
→ *x-pncmd*
→ *x-pn-realaudio*
→ *x-powerpoint*
→ *x-sh*
→ *x-stuffit*
→ *x-tar*
→ *x-tcl*

www-http (cont.) → *application (cont.)*
- → *x-tex*
- → *x-troff*
- → *x-ustar*
- → *x-zip-compressed*
- → *xpp5*
- → *zip-archive*
- → *x-netcdf*

→ *audio*
- → *basic*
- → *midi*
- → *mpeg*
- → *vnd-rn-realaudio*
- → *wav*
- → *x-aiff*
- → *x-midi*
- → *x-mpeg*
- → *x-mpgurl*
- → *x-pn-realaudio*
- → *x-wav*

→ *image*
- → *cgm*
- → *g3fax*
- → *gif*
- → *ief*
- → *jpeg*
- → *pict*
- → *png*
- → *tiff*
- → *vnd-rn-realflash*
- → *vnd-rn-realpix*
- → *x-bitmap*
- → *x-pixmap*
- → *x-quicktime*
- → *x-windowdump*
- → *x-xbm*

→ *text*
- → *enriched*
- → *html*
- → *plain*
- → *richtext*
- → *sgml*
- → *tab-separated-value*
- → *vnd-rn-text*
- → *css*

→ *video*
- → *avi*
- → *mpeg*
- → *msvideo*
- → *ms-video*
- → *quicktime*
- → *vnd-rn-realvideo*
- → *vnd-vivo*
- → *x-ls-asf*
- → *x-ls-asx*
- → *x-mpeg*
- → *x-ms-asf*
- → *x-ms-asx*
- → *x-msvideo*
- → *x-sgi-movie*

→ *x-world*
- → *x-vrml*

3.6.3 Extensibility

New Sub-classifications require much more thought and analysis when being added to HTTP. This arises from the following factors:

1. HTTP is a "text" based protocol
2. To support "minimum" execution overhead, when searching the HTTP Sub-Engine's "Sub-Protocol Info" database, a rather robust set of sequentially indexed, look-aside tables are employed.
   (a) The challenge here is to take a string from an HTTP packet (e.g. Content Type) and match it with any one of approximately 110+ well known (as is the case with Content Type)
   (b) And to do so within an embedded environment that is trying to keep up with the network packet rate at line speed.
   (c) The supported search mechanism can identify a single match candidate sub-string by looking at typically no more than 3 to 5 characters of the sub-string from the HTTP packet.
3. Adding a sub-classification to the HTTP "Sub-Protocol Info" Database is simply a matter of adding a new entry if the "Content Type" or "JDBC URL Component" is known.
4. Updating and/or extending the "look-aside" tables requires extreme caution and accuracy.

Extensibility for this sub-engine will be **tremendously improved** when the PDL compiler is incorporated for this sub-engine.

3.6.4 Planned Developments

New "Content Types" are springing up almost every week. As new applications are identified along with their designated Content Types, they will be added to the MeterFlow implementation.

WebNFS from Sun Microsystems, Inc. tunnels NFS file access over HTTP and is a clear candidate for inclusion into this sub-engine.

There are many other JDBC packages from various database manufactures and technology suppliers that are integrated with WWW. Oracle's being the most noted at this time. As more are identified along with their designated JDBC URL Selectors, they will be added to the MeterFlow implementation.

3.7 BackWeb

3.7.1 Features

BackWeb (BackWeb Technologies, Inc.) is a news/broadcast application. It may be configured to operate in either of 2 modes:

a) HTTP only (see Section 3.6 above)

b) UDP for access to BackWeb Servers & HTTP to access to 3rd party channels (polite mode)

BackWeb operates over UDP in what it calls its "Polite Client" mode. In this mode, BackWeb has an unusual mechanism of exchange that makes traffic in one direction very easy to see (well-known), but difficult to classify in the other direction.

The BackWeb sub-engine has been implemented specifically for BackWeb's UDP (Polite Mode) access protocol.

Key capabilities of this sub-engine include:

1. Tracking exchanges with BackWeb Servers in well-known BackWeb Server port traffic.
2. Remembering the access points of traffic from BackWeb Clients and creating sub-classifications for these access points.
3. When traffic to these access points are seen, it will be classified
   (a) as "backweb"

3.7.2 Sub-classifications

BackWeb (Polite Mode) traffic is classified as "backweb" in the "udp" subtree. No further sub-classifications for BackWeb are supported.

3.7.3 Extensibility

There are no known Sub-Classifications to be supported for BackWeb at this time.

3.7.4 Planned Developments

No further development efforts are currently planned for the BackWeb sub-engine.

4. In-Progress Protocol Classification

4.1 Real-Time Streaming Protocol (RTSP)

4.1.1 Features

The "Real-Time Streaming Protocol" is defined in RFC 2326. Like HTTP it is a "text" based protocol. Unlike HTTP, its principle purpose is to enable the controlled, on-demand delivery of real-time data, such as audio and video.

In function it acts similar to H.323's "Call Setup" and "Call Control" services, however, in a single connection on a well-known port. Ultimately, it serves to set up RTP/RTCP datastreams over UDP.

Key capabilities of this sub-engine include:

1. Tracking exchanges with the well-known RTSP server.
2. Detecting the assignment of RTP/RTCP audio and video, UDP datastreams access points as well as the audio and video "codecs" negotiated for use on them and creating RTP/RTCP sub-classifications for these access points.
3. When traffic to these RTP/RTCP access points are seen it will be classified

    (a) By the appropriate "codec" under "*rtp*", if the negotiated codec is a **known** audio/video stream type.
    (b) Minimally as "*rtp*", if the negotiated codec is unknown.

4. Allowing known sub-applications (audio/video datastreams) to be specified with respect to flow reporting with three levels of identification

    (a) Level 1 – Datastream Class (e.g. audio, video, other...)
    (b) Level 2 – Datastream Type within the Datastream Class
    (a) Level 3 – Datastream Sub-Type within the Datastream Type

4.1.2 Sub-classifications

RTSP traffic will be classified as "rtsp" in the "tcp" subtree. No further sub-classifications for RTSP are supported.

NEW Audio and video datastream Sub-classifications under "RTP" will include the following in the "udp" subtree.

RTP	→ *audio*	→ *1016*	*(Audio Transfer sub-channel)*
		→ *DVI4*	
		→ *L8*	
		→ *L16*	
		→ *LPC*	
		→ *MPA*	
		→ *VDVI*	
		→ *AIFF-C*	
	→ *video*	→ *CelB*	*(Video Transfer sub-channel)*
		→ *JPEG*	
		→ *MPV*	
		→ *MP2T*	
		→ *nv*	

Standards for the audio stream sub-classifications indicated above are:

1016 - frame based encoding using code-excited linear prediction (CELP) and is specified in Federal Standard FED-STD 1016

DVI4 - IMA ADPCM wave type, "IMA Recommended Practices for Enhancing Digital Audio Compatibility in Multimedia Systems (version 3.0)"

L8 - L8 denotes linear audio data, using 8-bits of precision with an offset of 128, that is, the most negative signal is encoded as zero.

L16 - L16 denotes uncompressed audio data, using 16-bit signed representation with 65535 equally divided steps between minimum and maximum signal level, ranging from -32768 to 32767. The value is represented in two's complement notation and network byte order.

LPC - LPC designates an experimental linear predictive encoding contributed by Ron Frederick, Xerox PARC, which is based on an implementation written by Ron Zuckerman, Motorola, posted to the Usenet group comp.dsp on June 26, 1992.

MPA - MPA denotes MPEG-I or MPEG-II audio encapsulated as elementary streams. The encoding is defined in ISO standards ISO/IEC 11172-3 and 13818-3. The encapsulation is specified in work in progress.

VDVI - VDVI is a variable-rate version of DVI4, yielding speech bit rates of between 10 and 25 kb/s. It is specified for single-channel operation only.

AIFF-c - Apple Computer, "Audio interchange file format AIFF-C," Aug. 1991. (also ftp://ftp.sgi.com/sgi/aiff-c.9.26.91.ps.Z).

Standards for the **video** stream sub-classifications indicated above are:

CelB - The CELL-B encoding is a proprietary encoding proposed by Sun Microsystems. "RTP payload format of CellB video encoding," Work in Progress, Internet Engineering Task Force, Aug. 1995.

JPEG - The encoding is specified in ISO Standards 10918-1 and 10918-2.

MPV - Designates the use MPEG-I and MPEG-II video encoding elementary streams as specified in ISO Standards ISO/IEC 11172 and 13818-2, respectively.

MP2T - MP2T designates the use of MPEG-II transport streams, for either audio or video.

nv - The encoding is implemented in the program 'nv', version 4, developed at Xerox PARC

4.1.3 Extensibility

New Sub-classifications will easily added as new entries in the RTSP Sub-Engine's "Sub-Protocol Info" table, if the Payload Types of the corresponding audio/video stream are known.

4.2 Novell Service Advertising Protocol (SAP)

4.2.1 Features

The Novell Service Advertising Protocol (SAP) is a protocol similar in nature to the "SUN RPC Portmapper" protocol. It is used to support the dynamic management and locating of "services" with regards to their locations (network addresses) and port assignments.

SAP uses a completely different protocol than the SUN RPC protocol Portmapper. Also, a fundamental difference from Sun RPC is that SAP periodically broadcasts services that are in it's advertising database.

Key capabilities of this sub-engine include:

1. Tracking SAP announcements periodically broadcast by Novell Netware servers.
2. Distinguishing such "announcement" traffic from traffic on application connections subsequently "mapped".
3. Detecting assignments of server application access assignments to various hosts and/or sockets and creating sub-classifications for these access points.
4. When traffic to these access points is seen, it will be classified
    - (a) By the appropriate application under "*nov-sap*", if the server application identifier in the announcement is a **known** sub-application.
    - (b) Minimally as "*nov-sap*", if the server application is unknown.
5. Allowing known sub-applications to be specified with respect to flow reporting with a single levels of identification
    - (a) Level 1 – SAP Mapped "Application Group"
    - (b) Level 2 – Sub-application within the Application Group

4.2.2 Sub-classifications

Sub-classifications under "nov-sap" will include the following in the "ipx.nov-pep" subtree.

nov-sap	→ *announce*	*(Novell SAP Announcements)*
	→ *ms-exchange*	*(Microsoft Exchange)*
	→ *sybase_sqlany*	*(Sybase SQL Anywhere)*
	→ *sybase_sqlenterprise*	*(Sybase SQL Enterprise)*
	→ *gupta-sqlbase*	*(Gupta SQLBase)*
	→ *ms-sna-server*	*(Microsoft SNA Server)*
	→ *ms-sql-server*	*(Microsoft SQL Server)*
	→ *citrix-app-server*	*(Citrix Application Server)*
	→ *citrix-app-server-nt*	*(Citrix Application Server for NT)*
	→ *hp-laserjet*	*(HP Laserjet Printer)*
	→ *advertising-print-svr*	*(Advertising Print Server)*
	→ *netware-sql-server*	*(Novell Netware SQL Server)*
	→ *remote-bridge*	*(Remote Bridge Router Service)*
	→ *bridge-server*	*(Bridge Server)*
	→ *print-queue*	*(Print Queue Server)*

4.2.3 Extensibility

New Sub-classifications will easily added as new entries in the Novell SAP Sub-Engine's "Sub-Protocol Info" table, if the SAP IDs of the corresponding application are known.

4.3 MS-Media

4.3.1 Features

MS-Media is a audio/video streaming, multimedia application (similar to RealAudio) from Microsoft. MS-Media may be configured to operate over UDP when transferring its payload. In this configuration, MS-Media has an unusual mechanism to allocate UDP resources for this purpose via an initial TCP connection.

The MS-Media sub-engine will be implemented specifically for MS-Media's access protocol.

Key capabilities of this sub-engine include:

1. Tracking connections to and exchanges in well-known MS-Media port traffic.
2. Detecting assignments of UDP access points to various hosts and/or ports and creating MS-Media sub-classifications for these access points.
3. When traffic to these access points are seen, it will be classified
    (a) as "ms-media"

4.3.2 Sub-classifications

Such MS-Media traffic will be classified as "*ms-media*" in the "*udp*" subtree. No further sub-classifications for MS-Media will be initially supported.

4.3.3 Extensibility

Sub-Classification is beyond the initial scope of the MS-Media sub-engine. Eventually, the sub-engine will be able to sub-classify the types of payloads being transferred.

4.4 Streamworks and VDOLive

4.4.1 Features

Streamworks and VDOLive are multi-media, streaming applications which transfer their payloads over UDP.

Like BackWeb, Streamworks and VDOLive employ unusual mechanisms of exchange that makes traffic one direction very easy to see (well-known), but difficult to classify in the other direction.

The BackWeb sub-engine will be expanded to further support Streamworks and VDOLive classification.

For a description of the key capabilities of the sub-engine, see Section 3.7:

4.4.2 Sub-classifications

Streamworks and VDOLive traffic is classified as "streamworks-xing-mpeg" and "vdolive" in the "udp" subtree; respectively. No further sub-classifications for these protocols will be supported.

4.4.3 Extensibility

Sub-Classification is beyond the initial scope Streamworks and VDOLive. Eventually, the sub-engine will be able to sub-classify the types of payloads being transferred.

Our Ref./Docket No: APPT-001-1-1 Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): Dietz, *et al.* Application No.: 10/684,776 Filed: October 14, 2003 Title: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK	Group Art Unit: 2157 Examiner: Moustafa M. Meky

TRANSMITTAL: RESPONSE TO OFFICE ACTION

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Commissioner:

Transmitted herewith is a response to an office action for the above referenced application. Included with the response are:

X A Declaration under 37 CFR 1.131 with Exhibits;

This application has:

_____ a small entity status. If a claim for such status has not earlier been made, consider this as a claim for small entity status.

_____ No additional fee is required.

03/10/2005 AWONDAF1 00000092 10684776
01 FC:1252 450.00 OP

Certificate of Mailing under 37 CFR 1.8

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on.

Date: Mar. 2, 2005 Signed: [signature]
Name: Amy Drury

_____ Applicant(s) believe(s) that no Extension of Time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition for an extension of time.

_X_ Applicant(s) hereby petition(s) for an Extension of Time under 37 CFR 1.136(a) of:

_____ one months ($120) _X_ two months ($450)

_____ three months ($1020) _____ four months ($1590)

If an additional extension of time is required, please consider this as a petition therefor.

_X_ A credit card payment form for the required fee(s) is attached.

_X_ The Commissioner is hereby authorized to charge payment of the following fees associated with this communication or credit any overpayment to Deposit Account No. 50-0292 (A DUPLICATE OF THIS TRANSMITTAL IS ATTACHED):

_X_ Any missing filing fees required under 37 CFR 1.16 for presentation of additional claims.

_X_ Any missing extension or petition fees required under 37 CFR 1.17.

Respectfully Submitted,

Mar. 2, 2005
Date

Dov Rosenfeld, Reg. No. 38687

Address for correspondence:
Dov Rosenfeld
5507 College Avenue, Suite 2,
Oakland, CA 94618
Tel. 510-547-3378; Fax: +1-510-291-2985

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

OIPE JC183 MAR 0 4 2005 PATENT & TRADEMARK OFFICE

Application Number	10/684,776
Filing Date	14 Oct 2003
First Named Inventor	Dietz, Russell S.
Group Art Unit	2157
Examiner Name	Moustafa M. Meky
Attorney Docket Number	APPT-001-1-1

ENCLOSURES ***(check all that apply)***

- [ ] Fee Transmittal Form
- [x] Fee Attached
- [x] Amendment / Response
- [ ] [ ] After Final
- [x] [x] Affidavits/declaration(s) under 1,131 with Exhibits
- [ ] Extension of Time Request
- [ ] Express Abandonment Request
- [ ] Information Disclosure Statement
- [ ] Certified Copy of Priority Document(s)
- [ ] Response to Missing Parts/ Incomplete Application
- [ ]
- [ ] [ ] Response to Missing Parts under 37 CFR 1.52 or 1.53
- [ ] Assignment Papers *(for an Application)*
- [ ] Drawing(s)
- [ ] Licensing-related Papers
- [ ] Petition Routing Slip (PTO/SB/69) and Accompanying Petition
- [ ] To Convert a Provisional Application
- [ ] Power of Attorney, Revocation Change of Correspondence Address
- [ ] Terminal Disclaimer
- [ ] Small Entity Statement
- [ ] Request of Refund
- [ ] After Allowance Communication to Group
- [ ] Appeal Communication to Board of Appeals and Interferences
- [ ] Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*
- [ ] Proprietary Information
- [ ] Status Letter
- [x] Additional Enclosure(s) *(please identify below)*:
- [x] Return Postcard
- [x] Exhibits to Declaration under 1.131
- [ ]

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT/ CORRESPONDENCE ADDRESS

Firm *or* Individual name	Dov Rosenfeld, Reg. No. 38687
Signature	*[signature]*
Date	March 2, 2005

ADDRESS FOR CORRESPONDENCE

Firm *or* Individual name	Dov Rosenfeld 5507 College Avenue, Suite 2, Oakland, CA 94618, Tel: 510-547-3378

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on this date: March 2, 2005

Type or printed name	Amy Drury		
Signature	*[signature]*	Date	March 2, 2005

Freeform Search

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: l5 same (type or kind)

Display: 10 **Documents in Display Format:** TI **Starting with Number** 1

Generate: ○ Hit List ◉ Hit Count ○ Side by Side ○ Image

Search Clear Interrupt

Search History

DATE: Friday, April 29, 2005 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=ADJ			
L6	l5 same (type or kind)	24	L6
L5	l4 same (protocol or format)	116	L5
L4	L3 same layer	182	L4
L3	l1 same l2	1071	L3
L2	(pars$4 or examin$5 or monitor$4) near4 (header or packet or datagram)	6862	L2
L1	(packet or datagram) near3 header	9621	L1

END OF SEARCH HISTORY

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

21921 7590 05/03/2005

DOV ROSENFELD
5507 COLLEGE AVE
SUITE 2
OAKLAND, CA 94618

EXAMINER	
MEKY, MOUSTAFA M	
ART UNIT	**PAPER NUMBER**
2157	

DATE MAILED: 05/03/2005

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/684,776	10/14/2003	Russell S. Dietz	APPT-001-1-1	3352

TITLE OF INVENTION: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1400	$300	$1700	08/03/2005

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE REFLECTS A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE APPLIED IN THIS APPLICATION. THE PTOL-85B (OR AN EQUIVALENT) MUST BE RETURNED WITHIN THIS PERIOD EVEN IF NO FEE IS DUE OR THE APPLICATION WILL BE REGARDED AS ABANDONED.

HOW TO REPLY TO THIS NOTICE:

I. Review the SMALL ENTITY status shown above.

If the SMALL ENTITY is shown as YES, verify your current SMALL ENTITY status:

A. If the status is the same, pay the TOTAL FEE(S) DUE shown above.

B. If the status above is to be removed, check box 5b on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and twice the amount of the ISSUE FEE shown above, or

If the SMALL ENTITY is shown as NO:

A. Pay TOTAL FEE(S) DUE shown above, or

B. If applicant claimed SMALL ENTITY status before, or is now claiming SMALL ENTITY status, check box 5a on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and 1/2 the ISSUE FEE shown above.

II. PART B - FEE(S) TRANSMITTAL should be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). Even if the fee(s) have already been paid, Part B - Fee(s) Transmittal should be completed and returned. If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Mail Stop ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

PTOL-85 (Rev. 12/04) Approved for use through 04/30/2007.

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (703) 746-4000

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

21921 7590 05/03/2005

DOV ROSENFELD
5507 COLLEGE AVE
SUITE 2
OAKLAND, CA 94618

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (703) 746-4000, on the date indicated below.

(Depositor's name)

(Signature)

(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/684,776	10/14/2003	Russell S. Dietz	APPT-001-1-1	3352

TITLE OF INVENTION: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1400	$300	$1700	08/03/2005

EXAMINER	ART UNIT	CLASS-SUBCLASS
MEKY, MOUSTAFA M	2157	709-224000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).
   - ☐ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
   - ☐ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list
   (1) the names of up to 3 registered patent attorneys or agents OR, alternatively,
   (2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.
   1 ____
   2 ____
   3 ____

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE (B) RESIDENCE: (CITY and STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent) : ☐ Individual ☐ Corporation or other private group entity ☐ Government

4a. The following fee(s) are enclosed:
- ☐ Issue Fee
- ☐ Publication Fee (No small entity discount permitted)
- ☐ Advance Order - # of Copies ____

4b. Payment of Fee(s):
- ☐ A check in the amount of the fee(s) is enclosed.
- ☐ Payment by credit card. Form PTO-2038 is attached.
- ☐ The Director is hereby authorized by charge the required fee(s), or credit any overpayment, to Deposit Account Number ____ (enclose an extra copy of this form).

5. **Change in Entity Status** (from status indicated above)
- ☐ a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.
- ☐ b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

The Director of the USPTO is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above.
NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature ____ Date ____

Typed or printed name ____ Registration No. ____

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/684,776	10/14/2003	Russell S. Dietz	APPT-001-1-1	3352

21921 7590 05/03/2005

DOV ROSENFELD
5507 COLLEGE AVE
SUITE 2
OAKLAND, CA 94618

EXAMINER	
MEKY, MOUSTAFA M	
ART UNIT	PAPER NUMBER
2157	

DATE MAILED: 05/03/2005

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment to date is 0 day(s). If the issue fee is paid on the date that is three months after the mailing date of this notice and the patent issues on the Tuesday before the date that is 28 weeks (six and a half months) after the mailing date of this notice, the Patent Term Adjustment will be 0 day(s).

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571) 272-7702. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at (703) 305-8283.

PTOL-85 (Rev. 12/04) Approved for use through 04/30/2007.

Notice of Allowability	**Application No.** 10/684,776	**Applicant(s)** DIETZ ET AL.	
	Examiner Moustafa M. Meky	**Art Unit** 2157	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *the response and the declaration under 37 CFR 1.131 filed 3/4/2005*.

2. ☒ The allowed claim(s) is/are *11-59*.

3. ☒ The drawings filed on *14 October 2003* are accepted by the Examiner.

4. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☐ All b) ☐ Some* c) ☐ None of the:
   1. ☐ Certified copies of the priority documents have been received.
   2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
   3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

5. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

6. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
   1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.
   (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

7. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☒ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3. ☐ Information Disclosure Statements (PTO-1449 or PTO/SB/08), Paper No./Mail Date _____
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application (PTO-152)
6. ☐ Interview Summary (PTO-413), Paper No./Mail Date _____.
7. ☐ Examiner's Amendment/Comment
8. ☒ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

MOUSTAFA M. MEKY
PRIMARY EXAMINER

REASONS FOR ALLOWANCE

The following is an examiner's statement of reasons for allowance: None of the prior art of record taken singularly or in combination teaches or suggest:

- looking up using at least some of selected packet portions and determining if the packet is of an existing flow, if the packet is of an existing flow, classifying the packet as belonging to the found existing flow, and if the packet is of a new flow, storing a new flow-entry for the new flow in a flow-entry database, including identifying information for future packets to be identified with the new flow-entry, wherein the database comprising none or more flow-entries for previously encountered conversational flows.(claims 11 & 29);
- looking up using at least some of selected packet portions and determining if the packet is of an existing flow, obtaining the last encountered state of the flow and performing any state operations specified for the state of the flow starting from the last encountered state of the flow, and if the packet is of a new flow, performing any analysis required for the initial state of the new flow and storing a new flow-entry for the new flow in flow-entry database, including identifying information for future packets to be identified with the new flow-entry (claim 54).

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Moustafa M. Meky whose telephone number is 571-272-4005. The examiner can normally be reached on flex.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Ario Etienne can be reached on 571-272-4001. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

MMM
4/29/2005

MOUSTAFA M. MEKY
PRIMARY EXAMINER

Notice of References Cited	Application/Control No. 10/684,776	Applicant(s)/Patent Under Reexamination DIETZ ET AL.	
	Examiner Moustafa M. Meky	Art Unit 2157	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-6,791,947	09-2004	Oskouy et al.	370/238
*	B	US-6,510,509	01-2003	Chopra et al.	712/13
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

now US Pat. 6,789,116

MMM 4-29-2005

al., filed June 30, 2000, ~~Attorney/Agent Reference Number APPT-001-5,~~ and incorporated herein by reference.

FIELD OF INVENTION

[0008] The present invention relates to computer networks, specifically to the real-time elucidation of packets communicated within a data network, including classification according to protocol and application program.

BACKGROUND TO THE PRESENT INVENTION

[0009] There has long been a need for network activity monitors. This need has become especially acute, however, given the recent popularity of the Internet and other internets—an "internet" being any plurality of interconnected networks which forms a larger, single network. With the growth of networks used as a collection of clients obtaining services from one or more servers on the network, it is increasingly important to be able to monitor the use of those services and to rate them accordingly. Such objective information, for example, as which services (*i.e.,* application programs) are being used, who is using them, how often they have been accessed, and for how long, is very useful in the maintenance and continued operation of these networks. It is especially important that selected users be able to access a network remotely in order to generate reports on network use in real time. Similarly, a need exists for a real-time network monitor that can provide alarms notifying selected users of problems that may occur with the network or site.

[0010] One prior art monitoring method uses log files. In this method, selected network activities may be analyzed retrospectively by reviewing log files, which are maintained by network servers and gateways. Log file monitors must access this data and analyze ("mine") its contents to determine statistics about the server or gateway. Several problems exist with this method, however. First, log file information does not provide a map of real-time usage; and secondly, log file mining does not supply complete information. This method relies on logs maintained by numerous network devices and servers, which requires that the information be subjected to refining and correlation. Also, sometimes information is simply not available to any gateway or server in order to make a log file entry.

METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This invention is a continuation of U.S. Patent Application Serial No. 09/608237 for METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK to inventors Dietz, et al., filed June 30, 2000, now US Pat. 6,651,099 ~~Attorney/Agent Reference Number APPT-001-1~~, the contents of which are incorporated herein by reference

[0002] This invention claims the benefit of U.S. Provisional Patent Application Serial No.: 60/141,903 for METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK to inventors Dietz, et al., filed June 30, 1999, the contents of which are incorporated herein by reference.

[0003] This application is related to the following U.S. patent applications, each filed concurrently with the present application, and each assigned to the assignee of the present invention:

MM 4-29-2005

[0004] U.S. Patent Application Serial No. 09/609179 for PROCESSING PROTOCOL SPECIFIC INFORMATION IN PACKETS SPECIFIED BY A PROTOCOL DESCRIPTION LANGUAGE, to inventors Koppenhaver, et al., filed June 30, 2000, now US Pat. 6,665,725 ~~Attorney/Agent Reference Number APPT-001-2~~, and incorporated herein by reference.

[0005] U.S. Patent Application Serial No. 09/608126 for RE-USING INFORMATION FROM DATA TRANSACTIONS FOR MAINTAINING STATISTICS IN NETWORK MONITORING, to inventors Dietz, et al., filed June 30, 2000, now US Pat. 6,839,751 ~~Attorney/Agent Reference Number APPT-001-3~~, and incorporated herein by reference.

[0006] U.S. Patent Application Serial No. 09/608266 for ASSOCIATIVE CACHE STRUCTURE FOR LOOKUPS AND UPDATES OF FLOW RECORDS IN A NETWORK MONITOR, to inventors Sarkissian, et al., filed June 30, 2000, now US Pat. 6,771,646 ~~Attorney/Agent Reference Number APPT-001-4~~, and incorporated herein by reference.

[0007] U.S. Patent Application Serial No. 09/608267 for STATE PROCESSOR FOR PATTERN MATCHING IN A NETWORK MONITOR DEVICE, to inventors Sarkissian, et

Issue Classification	Application/Control No.	Applicant(s)/Patent under Reexamination	
	10/684,776	DIETZ ET AL.	
	Examiner	Art Unit	
	Moustafa M. Meky	2157	

ISSUE CLASSIFICATION

ORIGINAL		CROSS REFERENCE(S)							
CLASS	SUBCLASS	CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)						
709	224	370	392						
INTERNATIONAL CLASSIFICATION									
/									
/									
/									
/									
/									

(Assistant Examiner) (Date)	Moustafa M. Meky MOUSTAFA M. MEKY PRIMARY EXAMINER 4/29/2005	Total Claims Allowed: 49
RWBrown 5-2-05 (Legal Instruments Examiner) (Date)	(Primary Examiner) (Date)	O.G. Print Claim(s): 1 — O.G. Print Fig.: 3

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original
	1	21	31		61		91		121		151		181
	2	22	32		62		92		122		152		182
	3	23	33		63		93		123		153		183
	4	24	34		64		94		124		154		184
	5	25	35		65		95		125		155		185
	6	26	36		66		96		126		156		186
	7	27	37		67		97		127		157		187
	8	28	38		68		98		128		158		188
	9	29	39		69		99		129		159		189
	10	30	40		70		100		130		160		190
1	11	31	41		71		101		131		161		191
2	12	32	42		72		102		132		162		192
3	13	33	43		73		103		133		163		193
4	14	34	44		74		104		134		164		194
5	15	35	45		75		105		135		165		195
6	16	36	46		76		106		136		166		196
7	17	37	47		77		107		137		167		197
8	18	38	48		78		108		138		168		198
9	19	39	49		79		109		139		169		199
10	20	40	50		80		110		140		170		200
11	21	41	51		81		111		141		171		201
12	22	42	52		82		112		142		172		202
13	23	43	53		83		113		143		173		203
14	24	44	54		84		114		144		174		204
15	25	45	55		85		115		145		175		205
16	26	46	56		86		116		146		176		206
17	27	47	57		87		117		147		177		207
18	28	48	58		88		118		148		178		208
19	29	49	59		89		119		149		179		209
20	30		60		90		120		150		180		210

Search Notes	Application/Control No. 10/684,776	Applicant(s)/Patent under Reexamination DIETZ ET AL.	
	Examiner Moustafa M. Meky	Art Unit 2157	

SEARCHED

Class	Subclass	Date	Examiner
UPDATE		4/29/2005	MMM

INTERFERENCE SEARCHED

Class	Subclass	Date	Examiner
709	**224**	**4/29/2005**	**MMM**
370	392	4/29/2005	MMM

SEARCH NOTES (INCLUDING SEARCH STRATEGY)

	DATE	EXMR
WEST	4/29/2005	MMM

Index of Claims

Application/Control No.	Applicant(s)/Patent under Reexamination
10/684,776	DIETZ ET AL.
Examiner	**Art Unit**
Moustafa M. Meky	2157

√	Rejected	–	(Through numeral) Cancelled	N	Non-Elected	A	Appeal
=	Allowed	+	Restricted	I	Interference	O	Objected

Claim Final	Claim Original	Date 4/29/2005	Date	Date	Date	Date	Date	Date	Date	Date
—	1	—								
—	2	—								
—	3	—								
—	4	—								
—	5	—								
—	6	—								
—	7	—								
—	8	—								
—	9	—								
—	10	—								
1	11	=								
2	12	=								
3	13	=								
4	14	=								
5	15	=								
6	16	=								
7	17	=								
8	18	=								
9	19	=								
10	20	=								
11	21	=								
12	22	=								
13	23	=								
14	24	=								
15	25	=								
16	26	=								
17	27	=								
18	28	=								
19	29	=								
20	30	=								
21	31	=								
22	32	=								
23	33	=								
24	34	=								
25	35	=								
26	36	=								
27	37	=								
28	38	=								
29	39	=								
30	40	=								
31	41	=								
32	42	=								
33	43	=								
34	44	=								
35	45	=								
36	46	=								
37	47	=								
38	48	=								
39	49	=								
40	50	=								

Claim Final	Claim Original	Date	Date	Date	Date	Date	Date	Date	Date	Date
41	51	=								
42	52	=								
43	53	=								
44	54	=								
45	55	=								
46	56	=								
47	57	=								
48	58	=								
49	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									

Claim Final	Claim Original	Date	Date	Date	Date	Date	Date	Date	Date	Date
	101									
	102									
	103									
	104									
	105									
	106									
	107									
	108									
	109									
	110									
	111									
	112									
	113									
	114									
	115									
	116									
	117									
	118									
	119									
	120									
	121									
	122									
	123									
	124									
	125									
	126									
	127									
	128									
	129									
	130									
	131									
	132									
	133									
	134									
	135									
	136									
	137									
	138									
	139									
	140									
	141									
	142									
	143									
	144									
	145									
	146									
	147									
	148									
	149									
	150									

INVENTEK
Dov Rosenfeld
5507 College Avenue, Suite 2
Oakland, CA 94618, USA
Phone: (510)547-3378; Fax: (510)291-2985
dov@inventek.com

Fax

RECEIVED
CENTRAL FAX CENTER
MAY 0 4 2005

Patent Application Ser. No.: 10/684,776	***Ref./Docket No***: APPT-001-1-1
Applicant(s): Dietz, *et al.*	***Examiner.***: Moustafa M. Meky
Filing Date: October 14, 2003	***Art Unit***: 2157

FAX COVER PAGE

TO: Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

United States Patent and Trademark Office
(Examiner Moustafa M. Meky, Art Unit 2157)

Fax No.: 703-872-9306

DATE: May 4, 2005

FROM: Dov Rosenfeld, Reg. No. 38687

RE: Information Disclosure Statement

Number of pages including cover: 11.

OFFICIAL COMMUNICATION

PLEASE URGENTLY DELIVER A COPY OF THIS IDS TO THE EXAMINER OF RECORD FOR THIS APPLICATION MOUSTAFA M. MEKY, ART UNIT 2157

Certificate of Facsimile Transmission under 37 CFR 1.8

I hereby certify that this response is being facsimile transmitted to the United States Patent and Trademark Office at telephone number 703-872-9306 addressed the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on.

Date: May 4, 2005 Signed: [signature]
Name: Amy Drury

Our Docket/Ref. No.: APPT-001-1-1 Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): Dietz et al. Serial No.: 10/684,776 Filed: October 14, 2003 Title: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK	Group Art Unit: 2157 Examiner: Moustafa M. Meky RECEIVED CENTRAL FAX CENTER MAY 0 4 2005

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL: INFORMATION DISCLOSURE STATEMENT

Dear Commissioner:

Transmitted herewith are:

X An Information Disclosure Statement for the above referenced patent application, together with PTO form 1449.

___ A payment for petition fees.

X The commissioner is hereby authorized to charge payment of any missing fee associated with this communication or credit any overpayment to Deposit Account 50-0292.

A DUPLICATE OF THIS TRANSMITTAL IS ATTACHED

Date: May 4, 2005

Respectfully submitted,

Dov Rosenfeld
Attorney/Agent for Applicant(s)
Reg. No. 38687

Correspondence Address:
Dov Rosenfeld
5507 College Avenue, Suite 2
Oakland, CA 94618
Telephone No.: 510-547-3378

Certificate of Facsimile Transmission under 37 CFR 1.8

I hereby certify that this response is being facsimile transmitted to the United States Patent and Trademark Office at telephone number 703-872-9306 addressed the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on.

Date: May 4, 2005 Signed:
Name: Amy Drury

Our Docket/Ref. No.: APPT-001-1-1 Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): Dietz et al. Serial No.: 10/684,776 Filed: October 14, 2003 Title: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK	Group Art Unit: 2157 Examiner: Moustafa M. Meky RECEIVED CENTRAL FAX CENTER MAY 0 4 2005

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL: INFORMATION DISCLOSURE STATEMENT

Dear Commissioner:

Transmitted herewith are:

_X_ An Information Disclosure Statement for the above referenced patent application, together with PTO form 1449 and a copy of each reference cited in form 1449.

___ A payment for petition fees.

_X_ The commissioner is hereby authorized to charge payment of any missing fee associated with this communication or credit any overpayment to Deposit Account 50-0292.

A DUPLICATE OF THIS TRANSMITTAL IS ATTACHED

Date: May 4, 2005

Respectfully submitted,

Dov Rosenfeld
Attorney/Agent for Applicant(s)
Reg. No. 38687

Correspondence Address:
Dov Rosenfeld
5507 College Avenue, Suite 2
Oakland, CA 94618
Telephone No.: 510-547-3378

Certificate of Facsimile Transmission under 37 CFR 1.8

I hereby certify that this response is being facsimile transmitted to the United States Patent and Trademark Office at telephone number 703-872-9306 addressed the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on.

Date: May 4, 2005 Signed: Name: Amy Drury

Our Docket/Ref. No.: APPT-001-1-1 Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): Dietz et al. Serial No.: 10/684,776 Filed: October 14, 2003 Title: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK	Group Art Unit: 2157 Examiner: Moustafa M. Meky RECEIVED CENTRAL FAX CENTER MAY 04 2005

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

INFORMATION DISCLOSURE STATEMENT

Dear Commissioner:

This Information Disclosure Statement is submitted:

____ under 37 CFR 1.97(b), or
(Within three months of filing national application; or date of entry of international application; or before mailing date of first office action on the merits; whichever occurs last)

X under 37 CFR 1.97(c) together with either a:
___ Certification under 37 CFR 1.97(e), or
X a $180.00 fee under 37 CFR 1.17(p)
(After the CFR 1.97(b) time period, but before final action or notice of allowance, whichever occurs first)

____ under 37 CFR 1.97(d) together with a:
___ Certification under 37 CFR 1.97(e), and
___ a petition under 37 CFR 1.97(d)(2)(ii), and
___ a $130.00 petition fee set forth in 37 CFR 1.17(i)(1).
(Filed after final action or notice of allowance, whichever occurs first, but before payment of the issue fee)

05/05/2005 EKOLII 00000019 10684776
180.00 OP

Certificate of Facsimile Transmission under 37 CFR 1.8

I hereby certify that this response is being facsimile transmitted to the United States Patent and Trademark Office at telephone number 703-872-9306 addressed the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on.

Date: May 4, 2005 Signed: [signature]
Name: Amy Drury

X Applicant(s) submit herewith Form PTO 1449-Information Disclosure Citation of patents, publications or other information of which applicant(s) are aware, which applicant(s) believe(s) may be material to the examination of this application and for which there may be a duty to disclose in accordance with 37 CFR 1.56.

It is expressly requested that the cited information be made of record in the application and appear among the "references cited" on any patent to issue therefrom.

The above-identified application is a continuation of prior U.S. Patent Application 09/608,237, filed June 30, 2000. This prior application is being relied upon for an earlier filing date under 35 U.S.C. § 120. Because the listed references were either cited by the PTO, or submitted to the PTO in this prior application, under 37 CFR § 1.98(d) Applicants submit that copies need not be provided.

As provided for by 37 CFR 1.97(g) and (h), no inference should be made that the information and references cited are prior art merely because they are in this statement and no representation is being made that a search has been conducted or that this statement encompasses all the possible relevant information.

Date: May 4, 2005

Respectfully submitted,

Dov Rosenfeld
Attorney/Agent for Applicant(s)
Reg. No. 38687

Correspondence Address:
Dov Rosenfeld
5507 College Avenue, Suite 2
Oakland, CA 94618
Telephone No.: 510-547-3378

INFORMATION DISCLOSURE STATEMENT (Use several sheets if necessary)	ATTY. DOCKET NO. APPT-001-1-1	SERIAL NO. 10/684,776
	APPLICANT Dietz et al.	
	FILING DATE 14 Oct 2003	GROUP 2157

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
	AA	4736320	Apr. 5, 1988	Bristol	364	300	Oct. 8, 1985
	AB	4891639	Jan. 2, 1990	Nakamura	340	825.500	Jun. 23, 1988
	AC	5101402	Mar. 31, 1992	Chui et al.	370	17	May 24, 1988
	AD	5247517	Sep. 21, 1993	Ross et al.	370	85.5	Sep. 2, 1992
	AE	5247693	Sep. 21, 1993	Bristol	395	800	Nov. 17, 1992
	AF	5315580	May 24, 1994	Phaal	370	13	Aug. 26, 1991
	AG	5339268	Aug. 16, 1994	Machida	365	49	Nov. 24, 1992
	AH	5351243	Sep. 27, 1994	Kalkunte et. al.	370	92	Dec. 27, 1991
	AI	5365514	Nov. 15, 1994	Hershey et al.	370	17	Mar. 1, 1993
	AJ	5375070	Dec. 20, 1994	Hershey at al.	364	550	Mar. 1, 1993
	AK	5394394	Feb. 28, 1995	Crowther et al.	370	60	Jun. 24, 1993

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLI-CATION DATE	COUNTRY	CLASS	SUB-CLASS	TRANS-LATION YES	NO
	AL							
	AM							

OTHER DISCLOSURES (Including Author, Title, Date, Pertinent Pages, Place of Publication, Etc.)

	AN	"Technical Note: the Narus System," Downloaded April 29, 1999 from www.narus.com, Narus Corporation, Redwood City California.
	AO	

EXAMINER	DATE CONSIDERED

*EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to Applicant.

INFORMATION DISCLOSURE STATEMENT	ATTY. DOCKET NO. APPT-001-1-1	SERIAL NO. 10/684,776
(Use several sheets if necessary)	APPLICANT Dietz et al.	
	FILING DATE 14 Oct 2003	GROUP 2157

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
	BA	5414650	May 9, 1995	Hekhuis	364	715.02	Mar. 24, 1993
	BB	5430709	Jul. 4, 1995	Galloway	370	13	Jun. 17, 1992
	BC	5432776	Jul. 11, 1995	Harper	370	17	Sep. 30, 1993
	BD	5493689	Feb. 20, 1996	Waclawsky et al.	395	821	Mar. 1, 1993
	BE	5500855	Mar. 19, 1996	Hershey et al.	370	17	Jan. 26, 1994
	BF	5568471	Oct. 22, 1996	Hershey et al.	370	17	Sep. 6, 1995
	BG	5574875	Nov. 12, 1996	Stansfield et al.	395	403	Mar. 12, 1993
	BH	5586266	Dec. 17, 1996	Hershey et al.	395	200.11	Oct. 15, 1993
	BI	5606668	Feb. 25, 1997	Shwed	395	200.11	Dec. 15, 1993
	BJ	5608662	Mar. 4, 1997	Large et al.	364	724.01	Jan. 12, 1995
	BK	5634009	May 27, 1997	Iddon et al.	395	200.11	Oct. 27, 1995

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLI-CATION DATE	COUNTRY	CLASS	SUB-CLASS	TRANSLATION YES	NO
	BL							
	BM							

OTHER DISCLOSURES (Including Author, Title, Date, Pertinent Pages, Place of Publication, Etc.)

	BN	
	BO	

EXAMINER	DATE CONSIDERED

*EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to Applicant.

FORM - 1449 SHEET 3 OF 5.

INFORMATION DISCLOSURE STATEMENT

(Use several sheets if necessary)

ATTY. DOCKET NO. APPT-001-1-1

SERIAL NO. 10/684,776

APPLICANT Dietz et al.

FILING DATE 14 Oct 2003

GROUP 2157

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
	CA	5651002	Jul. 22, 1997	Van Seters et all.	370	392	Jul. 12, 1995
	CB	5684954	Nov. 4, 1997	Kaiserswerth et al.	395	200.2	Mar. 20, 1993
	CC	5732213	Mar. 24, 1998	Gessel et al.	395	200.11	Mar. 22, 1996
	CD	5740355	Apr. 14, 1998	Watanabe et al.	395	183.21	Jun. 4, 1996
	CE	5761424	Jun. 2, 1998	Adams et al.	395	200.47	Dec. 29, 1995
	CF	5764638	Jun. 9, 1998	Ketchum	370	401	Sep. 14, 1995
	CG	5781735	Jul. 14, 1998	Southard	395	200.54	Sep. 4, 1997
	CH	5784298	Jul. 21, 1998	Hershey et al.	364	557	Jul. 11, 1996
	CI	5787253	Jul. 28, 1998	McCreery et al.	395	200.61	May 28, 1996
	CJ	5805808	Sep. 8, 1998	Hansani et al.	395	200.2	Apr. 9, 1997
	CK	5812529	Sep. 22, 1998	Czarnik et al.	370	245	Nov. 12, 1996

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLICATION DATE	COUNTRY	CLASS	SUB-CLASS	TRANSLATION YES	NO
	CL							
	CM							

OTHER DISCLOSURES (Including Author, Title, Date, Pertinent Pages, Place of Publication, Etc.)

	CN	
	CO	

EXAMINER

DATE CONSIDERED

*EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to Applicant.

aLFORM . 1449 SHEET 4 OF 5.

INFORMATION DISCLOSURE STATEMENT (Use several sheets if necessary)	ATTY. DOCKET NO. APPT-001-1-1	SERIAL NO. 10/684,776
	APPLICANT Dietz et al.	
	FILING DATE 14 Oct 2003	GROUP 2157

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
	DA	5819028	Oct. 6, 1998	Manghirmalani et al.	395	185.1	Apr. 16, 1997
	DB	5825774	Oct. 20, 1998	Ready et al.	370	401	Jul. 12, 1995
	DC	5835726	Nov. 10, 1998	Shwed et al.	395	200.59	Jun. 17, 1996
	DD	5838919	Nov. 17, 1998	Schwaller et al.	395	200.54	Sep. 10, 1996
	DE	5841895	Nov. 24, 1998	Huffman	382	155	Oct. 25, 1996
	DF	5850386	Dec. 15, 1998	Anderson et al.	370	241	Nov. 1, 1996
	DG	5,703,877	Dec. 30, 1997	Nuber et al.	370	395	Nov. 22, 1995
	DH	5862335	Jan. 19, 1999	Welch, Jr. et al.	395	200.54	Apr. 1, 1993
	DI	5878420	Mar. 2, 1999	de la Salle	707	10	Oct. 29, 1997
	DJ	5893155	Apr. 6, 1999	Cheriton	711	144	Dec. 3, 1996
	DK	5903754	May 11, 1999	Pearson	395	680	Nov. 14, 1997

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLI-CATION DATE	COUNTRY	CLASS	SUB-CLASS	TRANS-LATION YES	NO
	DL							
	DM							

OTHER DISCLOSURES (Including Author, Title, Date, Pertinent Pages, Place of Publication, Etc.)

	DN	
	DO	

EXAMINER	DATE CONSIDERED

*EXAMINER: initial if citation considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to Applicant.

INFORMATION DISCLOSURE STATEMENT *(Use several sheets if necessary)*	ATTY. DOCKET NO. APPT-001-1-1	SERIAL NO. 10/684,776
	APPLICANT Dietz et al.	
	FILING DATE 14 Oct 2003	GROUP 2157

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
	EA	5414704	May 9, 1995	Spinney	370	60	Apr. 5, 1994
	EB	6014380	Jan 11, 2000	Hendel et al.	370	392	Jun. 30, 1997
	EC	5511215	Apr. 23, 1996	Terasaka et al.	395	800	Oct. 26, 1993
	ED	5,249,292	Sep. 28, 1993	Chiappa	395	650	Mar.10,1992
	EE	5,511,213	Apr. 23, 1996	Correa	395	800	May 8, 1992
	EF						
	EG						
	EH						
	EI						
	EJ						
	EK						

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLI-CATION DATE	COUNTRY	CLASS	SUB-CLASS	TRANS-LATION YES	NO
	EL							
	EM							

OTHER DISCLOSURES (Including Author, Title, Date, Pertinent Pages, Place of Publication, Etc.)

	EN	
	EO	

EXAMINER	DATE CONSIDERED

*EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to Applicant.

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail — Mail Stop ISSUE FEE, Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450

or Fax (703) 746-4000

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

21921 7590 05/03/2005

DOV ROSENFELD
5507 COLLEGE AVE
SUITE 2
OAKLAND, CA 94618

05/17/2005 MGEBREM2 00000038 10684776

01 FC:1501 1400.00 OP
02 FC:1504 300.00 OP
03 FC:8001 [illegible]

OIPE JC14 MAY 1 7 2005 PATENT & TRADEMARK OFFICE

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (703) 746-4000, on the date indicated below.

Amy Drury (Depositor's name)
[signature] (Signature)
May 17, 2005 (Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/684,776	10/14/2003	Russell S. Dietz	APPT-001-1-1	3352

TITLE OF INVENTION: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1400	$300	$1700	08/03/2005

EXAMINER	ART UNIT	CLASS-SUBCLASS
MEKY, MOUSTAFA M	2157	709-224000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).
   - ☐ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
   - ☐ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. Use of a Customer Number is required.

2. For printing on the patent front page, list
   (1) the names of up to 3 registered patent attorneys or agents OR, alternatively,
   (2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.
   1. Dov Rosenfeld
   2. Inventek
   3. ______

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: Hi/fn, Inc.

(B) RESIDENCE: (CITY and STATE OR COUNTRY): Los Gatos, CA

Please check the appropriate assignee category or categories (will not be printed on the patent): ☐ Individual ☒ Corporation or other private group entity ☐ Government

4a. The following fee(s) are enclosed:
- ☒ Issue Fee
- ☒ Publication Fee (No small entity discount permitted)
- ☒ Advance Order - # of Copies 10

4b. Payment of Fee(s):
- ☐ A check in the amount of the fee(s) is enclosed.
- ☒ Payment by credit card. Form PTO-2038 is attached.
- ☒ The Director is hereby authorized by charge the required fee(s), or credit any overpayment, to Deposit Account Number 50-0292 (enclose an extra copy of this form).

5. Change in Entity Status (from status indicated above)
- ☐ a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.
- ☐ b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

The Director of the USPTO is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above.
NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature X [signature] Date May 17, 2005

Typed or printed name Dov Rosenfeld Registration No. 38,687

INVENTEK

Dov Rosenfeld
5507 College Avenue, Suite 2
Oakland, CA 94618, USA
Phone: (510) 547-3378; Fax: (510) 291-2985
dov@inventek.com

OIPE JC148 MAY 1 7 2005 PATENT & TRADEMARK OFFICE

Fax

OUR REF: APPT-001-1-1

TO:	Mail Stop Issue Fee Commissioner for Patents P.O. Box 1450 Alexandria, VA 22313-1450	***FAX No.:*** (703) 746-4000
DATE:	May 17, 2005	
FROM:	Dov Rosenfeld, Reg. No., 38,687	
RE:	Issue Fee for Application No.: 10/684,776	

Number of pages including cover: 6

	OFFICIAL COMMUNICATION **ISSUE FEE PAYMENT**

Included herewith are:

- A transmittal letter and copy
- Fee(s) Transmittal (form PTOL-85)
- Credit Card charge form for issue fee

Certificate of Facsimile Transmission under 37 CFR 1.8

I hereby certify that this response is being facsimile transmitted to the United States Patent and Trademark Office at telephone number (703) 746-4000 addressed to Mail Stop Issue Fee, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on.

Date: May 17, 2005 Signed: [signature]
Name: Amy Drury

Our Ref./Docket No: APPT-001-1-1 Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): Dietz, *et al.*	Group Art Unit: 2157
Application No.: 10/684,776	Examiner: Moustafa M. Meky
Filed: October 14, 2003	Notice of Allowance Mailed: May, 3, 2005
Title: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK	Confirmation No: 3352

SUBMISSION OF ISSUE FEE

Mail Stop ISSUE FEE
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Commissioner:

Transmitted herewith is a completed "Issue Fee Transmittal" Form. Included with the form are:

_X_ A credit card payment form for the issue fee, publication fee and any advance order of copies;
____ drawing corrections (with separate letter);
____ formal drawings (with separate letter);

_X_ The Commissioner is hereby authorized to charge payment of the any missing fee or credit any overpayment to Deposit Account No. 50-0292
(A DUPLICATE OF THIS TRANSMITTAL IS ATTACHED):

Respectfully Submitted,

May 17, 2005
Date

Dov Rosenfeld, Reg. No. 38687

Address for correspondence:
Dov Rosenfeld
5507 College Avenue, Suite 2,
Oakland, CA 94618
Tel. 510-547-3378; Fax: 510-291-2985

Certificate of Facsimile Transmission under 37 CFR 1.8

I hereby certify that this response is being facsimile transmitted to the United States Patent and Trademark Office at telephone number (703) 746-4000 addressed to Mail Stop Issue Fee, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on.

Date: May 17, 2005 Signed: [signature]
Name: Amy Drury

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/684,776	10/14/2003	Russell S. Dietz	APPT-001-1-1	3352

21921 7590 08/19/2005

DOV ROSENFELD
5507 COLLEGE AVE
SUITE 2
OAKLAND, CA 94618

EXAMINER	
MEKY, MOUSTAFA M	
ART UNIT	**PAPER NUMBER**
2157	

DATE MAILED: 08/19/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

Supp. Notice of Allowability	Application No. 10/684,776	Applicant(s) DIETZ ET AL.	
	Examiner Moustafa M. Meky	Art Unit 2157	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *the IDS filed 5/4/2005*.

2. ☒ The allowed claim(s) is/are *11-50*.

3. ☒ The drawings filed on *14 October 2003* are accepted by the Examiner.

4. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☐ All b) ☐ Some* c) ☐ None of the:
   1. ☐ Certified copies of the priority documents have been received.
   2. ☐ Certified copies of the priority documents have been received in Application No. _____.
   3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

5. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

6. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
   1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.
   (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

7. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☐ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3. ☒ Information Disclosure Statements (PTO-1449 or PTO/SB/08), Paper No./Mail Date 5/4/2005
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application (PTO-152)
6. ☐ Interview Summary (PTO-413), Paper No./Mail Date _____.
7. ☐ Examiner's Amendment/Comment
8. ☐ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

MOUSTAFA M. MEKY
PRIMARY EXAMINER

Et al FORM - 1449 SHEET 1 OF 5.

INFORMATION DISCLOSURE STATEMENT (Use several sheets if necessary)	ATTY. DOCKET NO. APPT-001-1-1	SERIAL NO. 10/684,776
	APPLICANT Dietz et al.	
	FILING DATE 14 Oct 2003	GROUP 2157

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
MMM	AA	4736320	Apr. 5, 1988	Bristol	364	300	Oct. 8, 1985
MMM	AB	4891639	Jan. 2, 1990	Nakamura	340	825.500	Jun. 23, 1988
MMM	AC	5101402	Mar. 31, 1992	Chui et al.	370	17	May 24, 1988
MMM	AD	5247517	Sep. 21, 1993	Ross et al.	370	85.5	Sep. 2, 1992
MMM	AE	5247693	Sep. 21, 1993	Bristol	395	800	Nov. 17, 1992
MMM	AF	5315580	May 24, 1994	Phaal	370	13	Aug. 26, 1991
MMM	AG	5339268	Aug. 16, 1994	Machida	365	49	Nov. 24, 1992
MMM	AH	5351243	Sep. 27, 1994	Kalkunte et. al.	370	92	Dec. 27, 1991
MMM	AI	5365514	Nov. 15, 1994	Hershey et al.	370	17	Mar. 1, 1993
MMM	AJ	5375070	Dec. 20, 1994	Hershey at al.	364	550	Mar. 1, 1993
MMM	AK	5394394	Feb. 28, 1995	Crowther et al.	370	60	Jun. 24, 1993

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLICATION DATE	COUNTRY	CLASS	SUB-CLASS	TRANSLATION YES	NO
	AL							
	AM							

OTHER DISCLOSURES (Including Author, Title, Date, Pertinent Pages, Place of Publication, Etc.)

MMM	AN	"Technical Note: the Narus System," Downloaded April 29, 1999 from www.narus.com, Narus Corporation, Redwood City California.
	AO	

EXAMINER	DATE CONSIDERED
M. Mehy	8-14-2005

*EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to Applicant.

INFORMATION DISCLOSURE STATEMENT

(Use several sheets if necessary)

ATTY. DOCKET NO.	SERIAL NO.
APPT-001-1-1	10/684,776
APPLICANT	
Dietz et al.	
FILING DATE	GROUP
14 Oct 2003	2157

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
MMM	BA	5414650	May 9, 1995	Hekhuis	364	715.02	Mar. 24, 1993
MMM	BB	5430709	Jul. 4, 1995	Galloway	370	13	Jun. 17, 1992
MMM	BC	5432776	Jul. 11, 1995	Harper	370	17	Sep. 30, 1993
MMM	BD	5493689	Feb. 20, 1996	Waclawsky et al.	395	821	Mar. 1, 1993
MMM	BE	5500855	Mar. 19, 1996	Hershey et al.	370	17	Jan. 26, 1994
MMM	BF	5568471	Oct. 22, 1996	Hershey et al.	370	17	Sep. 6, 1995
MMM	BG	5574875	Nov. 12, 1996	Stansfield et al.	395	403	Mar. 12, 1993
MMM	BH	5586266	Dec. 17, 1996	Hershey et al.	395	200.11	Oct. 15, 1993
MMM	BI	5606668	Feb. 25, 1997	Shwed	395	200.11	Dec. 15, 1993
MMM	BJ	5608662	Mar. 4, 1997	Large et al.	364	724.01	Jan. 12, 1995
MMM	BK	5634009	May 27, 1997	Iddon et al.	395	200.11	Oct. 27, 1995

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLICATION DATE	COUNTRY	CLASS	SUB-CLASS	TRANSLATION YES	NO
	BL							
	BM							

OTHER DISCLOSURES (Including Author, Title, Date, Pertinent Pages, Place of Publication, Etc.)

	BN	
	BO	

EXAMINER	DATE CONSIDERED
M. Mehy	8-14-2005

*EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to Applicant.

aLFORM - 1449 SHEET 3 OF 5.

INFORMATION DISCLOSURE STATEMENT (Use several sheets if necessary)	ATTY. DOCKET NO. APPT-001-1-1	SERIAL NO. 10/684,776
	APPLICANT Dietz et al.	
	FILING DATE 14 Oct 2003	GROUP 2157

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
MMM	CA	5651002	Jul. 22, 1997	Van Seters et all.	370	392	Jul. 12, 1995
MMM	CB	5684954	Nov. 4, 1997	Kaiserswerth et al.	395	200.2	Mar. 20, 1993
MMM	CC	5732213	Mar. 24, 1998	Gessel et al.	395	200.11	Mar. 22, 1996
MMM	CD	5740355	Apr. 14, 1998	Watanabe et al.	395	183.21	Jun. 4, 1996
MMM	CE	5761424	Jun. 2, 1998	Adams et al.	395	200.47	Dec. 29, 1995
MMM	CF	5764638	Jun. 9, 1998	Ketchum	370	401	Sep. 14, 1995
MMM	CG	5781735	Jul. 14, 1998	Southard	395	200.54	Sep. 4, 1997
MMM	CH	5784298	Jul. 21, 1998	Hershey et al.	364	557	Jul. 11, 1996
MMM	CI	5787253	Jul. 28, 1998	McCreery et al.	395	200.61	May 28, 1996
MMM	CJ	5805808	Sep. 8, 1998	Hansani et al.	395	200.2	Apr. 9, 1997
MMM	CK	5812529	Sep. 22, 1998	Czernik et al.	370	245	Nov. 12, 1996

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLICATION DATE	COUNTRY	CLASS	SUB-CLASS	TRANSLATION YES	NO
	CL							
	CM							

OTHER DISCLOSURES (Including Author, Title, Date, Pertinent Pages, Place of Publication, Etc.)

	CN	
	CO	

EXAMINER M. Mehy	DATE CONSIDERED 8-14-2005

*EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to Applicant.

INFORMATION DISCLOSURE STATEMENT (Use several sheets if necessary)	ATTY. DOCKET NO. APPT-001-1-1	SERIAL NO. 10/684,776
	APPLICANT Dietz et al.	
	FILING DATE 14 Oct 2003	GROUP 2157

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
MMM	DA	5819028	Oct. 6, 1998	Manghirmalani et al.	395	185.1	Apr. 16, 1997
MMM	DB	5825774	Oct. 20, 1998	Ready et al.	370	401	Jul. 12, 1995
MMM	DC	5835726	Nov. 10, 1998	Shwed et al.	395	200.59	Jun. 17, 1996
MMM	DD	5838919	Nov. 17, 1998	Schwaller et al.	395	200.54	Sep. 10, 1996
MMM	DE	5841895	Nov. 24, 1998	Huffman	382	155	Oct. 25, 1996
MMM	DF	5850386	Dec. 15, 1998	Anderson et al.	370	241	Nov. 1, 1996
MMM	DG	5,703,877	Dec. 30, 1997	Nuber et al.	370	395	Nov. 22, 1995
MMM	DH	5862335	Jan. 19, 1999	Welch, Jr. et al.	395	200.54	Apr. 1, 1993
MMM	DI	5878420	Mar. 2, 1999	de la Salle	707	10	Oct. 29, 1997
MMM	DJ	5893155	Apr. 6, 1999	Cheriton	711	144	Dec. 3, 1996
MMM	DK	5903754	May 11, 1999	Pearson	395	680	Nov. 14, 1997

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLI-CATION DATE	COUNTRY	CLASS	SUB-CLASS	TRANSLATION YES	NO
	DL							
	DM							

OTHER DISCLOSURES (Including Author, Title, Date, Pertinent Pages, Place of Publication, Etc.)

	DN	
	DO	

EXAMINER M. Mehy	DATE CONSIDERED 8-14-2005

*EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to Applicant.

PTO FORM - 1449 SHEET 5 OF 5.

INFORMATION DISCLOSURE STATEMENT (Use several sheets if necessary)	ATTY. DOCKET NO. APPT-001-1-1	SERIAL NO. 10/684,776
	APPLICANT Dietz et al.	
	FILING DATE 14 Oct 2003	GROUP 2157

U.S. PATENT DOCUMENTS

*EXAMINER INITIAL		DOCUMENT NUMBER	DATE	NAME	CLASS	SUB-CLASS	FILING DATE IF APPROPRIATE
MMM	EA	5414704	May 9, 1995	Spinney	370	60	Apr. 5, 1994
MMM	EB	6014380	Jan 11, 2000	Hendel et al.	370	392	Jun. 30, 1997
MMM	EC	5511215	Apr. 23, 1996	Terasaka et al.	395	800	Oct. 26, 1993
MMM	ED	5,249,292	Sep. 28, 1993	Chiappa	395	650	Mar.10,1992
MMM	EE	5,511,213	Apr. 23, 1996	Correa	395	800	May 8, 1992
	EF						
	EG						
	EH						
	EI						
	EJ						
	EK						

FOREIGN PATENT DOCUMENTS

		DOCUMENT NUMBER	PUBLI-CATION DATE	COUNTRY	CLASS	SUB-CLASS	TRANSLATION YES	NO
	EL							
	EM							

OTHER DISCLOSURES (Including Author, Title, Date, Pertinent Pages, Place of Publication, Etc.)

	EN	
	EO	

EXAMINER M. Mely	DATE CONSIDERED 8-14-2005

*EXAMINER: Initial if citation considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include a copy of this form with next communication to Applicant.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/684,776	10/14/2003	Russell S. Dietz	APPT-001-1-1	3352

21921 7590 08/26/2005

DOV ROSENFELD
5507 COLLEGE AVE
SUITE 2
OAKLAND, CA 94618

EXAMINER	
MEKY, MOUSTAFA M	
ART UNIT	**PAPER NUMBER**
2157	

DATE MAILED: 08/26/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

2nd Supp. Notice of Allowability	Application No. 10/684,776	Applicant(s) DIETZ ET AL.	
	Examiner Moustafa M. Meky	Art Unit 2157	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *the IDS filed 5/4/2005*.

2. ☒ The allowed claim(s) is/are *11-59*.

3. ☐ The drawings filed on _____ are accepted by the Examiner.

4. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☐ All b) ☐ Some* c) ☐ None of the:
   1. ☐ Certified copies of the priority documents have been received.
   2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
   3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

5. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

6. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
   1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.
   (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

7. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☐ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3. ☐ Information Disclosure Statements (PTO-1449 or PTO/SB/08), Paper No./Mail Date _____
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application (PTO-152)
6. ☐ Interview Summary (PTO-413), Paper No./Mail Date _____ .
7. ☐ Examiner's Amendment/Comment
8. ☐ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

Moustafa M. Meky

MOUSTAFA M. MEKY
PRIMARY EXAMINER

OIPE OCT 2 4 2005 PATENT & TRADEMARK OFFICE

Our Ref./Docket No: APPT-001-1-1

Patent

C3C

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Dietz, *et al.*

Assignee: Hi/fn, Inc.

Patent No: 6,954,789

Issue Date: October, 11, 2005

Application No.: 10/684,776

Filed: October 14, 2003

Title: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK

Certificate OCT 2 6 2005 of Correction

REQUEST FOR CERTIFICATE OF CORRECTIONS

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Commissioner:

The above patent contains significant error(s) as indicated on the attached Certificate of Correction form (submitted in duplicate).

X Such error(s) arose through the fault of the Patent and Trademark Office. It is requested that the certificate be issued at no cost to the applicant.

However, if it is determined that the error(s) arose through the fault of applicant(s), please note that such error is of clerical error or minor nature and occurred in good faith and therefore issuance of the certificate of Correction is respectfully requested. The Commissioner is authorized to charge Deposit Account No. 50-0292 any required fee. A duplicate of this request is attached.

____ Such error(s) arose through the fault of applicant(s). A credit card charge form for the fee is enclosed. Each such error is of clerical error or minor nature and occurred in good faith and therefore issuance of the certificate of Correction is respectfully requested.

Certificate of Mailing under 37 CFR 1.8

I hereby certify that this response is being deposited with the United States Postal Service as first class mail in an envelope addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on.

Date: Oct. 21, 2005

Signed: [signature]

Name: Amy Drury

OCT 2 7 2005

OCT 2 7 2005

Such error(s) specifically:

In column 9, line 60, kindly change "layer model" to --layered model--.

In column 33, line 60, kindly insert between "and a" and "denoted as" the phrase -- next-state that the state processor should proceed to for more complex recognition jobs,--.

In column 35, line 51 (the 1st line of claim 7), kindly change "clalm" to --claim--.

In column 37, line 14 (the 1st line of claim 23), kindly change "ciaim" to --claim--.

The undersigned requests being contacted at (510) 547-3378 if there are any questions or clarifications, or if there are any problems with issuance of the Certificate of Correction.

Respectfully Submitted,

Oct. 21, 2005
Date

Dov Rosenfeld, Reg. No. 38687
Agent of Record.

Address for correspondence:
Dov Rosenfeld
5507 College Avenue, Suite 2,
Oakland, CA 94618
Tel. (510) 547-3378; Fax: (510) 291-2985

OCT 27 2005

PTO/SB/44 (10-96)
Approved for use through 6/30/99. OMB 0651-0033
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

(Also Form PTO-1050)

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO : 6,954,789

DATED : October 11, 2005

INVENTOR(S) : Dietz, et al.

It is certified that an error appears in the above-identified patent and that said Letters Patent are hereby corrected as shown below:

In column 9, line 60, kindly change "layer model" to --layered model--.

In column 33, line 60, kindly insert between "and a" and "denoted as" the phrase -- next-state that the state processor should proceed to for more complex recognition jobs,--.

In column 35, line 51 (the 1st line of claim 7), kindly change "clalm" to --claim--.

In column 37, line 14 (the 1st line of claim 23), kindly change "ciaim" to --claim--.

MAILING ADDRESS OF SENDER (Atty/Agent of Record):
Dov Rosenfeld, Reg. No. 38687
5507 College Avenue, Suite 2
Oakland, CA 94618

PATENT NO: 6,954,789
No. of additional copies

OCT 27 2005

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,954,789 B2
DATED : October 11, 2005
INVENTOR(S) : Dietz et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 9,
Line 60, change "layer model" to -- layered model --.

Column 33,
Line 60, insert between "and a" and "denoted as" the phrase -- next-state that the state processor should proceed to for more complex recognition jobs, --.

Column 35,
Lline 51, change "clalm" to -- claim --.

Column 37,
Line 14, change "ciaim" to -- claim --.

Signed and Sealed this

Seventh Day of March, 2006

JON W. DUDAS
Director of the United States Patent and Trademark Office

AO 120 (Rev. 08/10)

TO: Mail Stop 8 Director of the U.S. Patent and Trademark Office P.O. Box 1450 Alexandria, VA 22313-1450	**REPORT ON THE FILING OR DETERMINATION OF AN ACTION REGARDING A PATENT OR TRADEMARK**

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court Eastern District of Texas, Marshall Division on the following

☐ Trademarks or ☑ Patents. (☐ the patent action involves 35 U.S.C. § 292.):

DOCKET NO. 2:13-cv-206	DATE FILED 3/12/2013	U.S. DISTRICT COURT Eastern District of Texas, Marshall Division
PLAINTIFF Packet Intelligence, LLC		DEFENDANT Huawei Device USA Inc., et al.

PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1 6,651,099	11/18/2003	See Attachment A
2 6,954,789	10/11/2005	See Attachment A
3 6,665,725	12/16/2003	See Attachment A
4 6,839,751	1/4/2005	See Attachment A
5 6,771,646	8/3/2004	See Attachment A

In the above—entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY ☐ Amendment ☐ Answer ☐ Cross Bill ☐ Other Pleading	
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1 6,789,116	9/7/2004	See Attachment A
2 7,229,282	11/20/2007	See Attachment A
3		
4		
5		

In the above—entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT

CLERK	(BY) DEPUTY CLERK	DATE

Copy 1—Upon initiation of action, mail this copy to Director Copy 3—Upon termination of action, mail this copy to Director
Copy 2—Upon filing document adding patent(s), mail this copy to Director Copy 4—Case file copy

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

REVOCATION AND NEW POWER OF ATTORNEY

Under 37 CFR §3.73(b), Packet Intelligence LLC, a corporation, certifies that it is the assignee of 100% of the entire right, title, and interest in the patent applications identified by virtue of the assignments as identified and listed in TABLE 1 attached.

The undersigned, whose title is supplied below, is empowered to act on behalf of the assignee.

The undersigned, acting on behalf of the assignee, hereby revokes all powers of attorney previously granted in the application and appoints all registered practitioners associated with.

PTO Customer Number: 96039

with full power of substitution and revocation, to prosecute the patents and patent applications and to transact all business in the United States Patent and Trademark Office connected with the patents and patent applications.

This Revocation and New Power of Attorney is being filed to remove from the previous list of practitioners, any of those practitioners who have neither previously had nor will in the future have any direct or indirect participation in the prosecution of this application.

All correspondence regarding the patents and patent applications should be sent to:

PTO Customer Number: 96039

Direct all telephone calls to Lawrence A. Aaronson, Reg. No. 38,369, at telephone number 404.645.7700.

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 Section 1001 of Title 18 of the United States

Code and that such willful false statements may jeopardize the validity of the application or any patents issued thereon.

Respectfully submitted

Packet Intelligence LLC

Date: 10 Apr 2013

Name: Rondal Moore
Title: General Counsel

TABLE 1
Packet Intelligence LLC

Attorney Docket No.	Application No.	Date Filed	Patent No.	Assignor	Assignee	Assignment Recordation	
						Date	Reel / Frame
	09608237	06/30/2000	6651099	Inventors	APPTITUDE, INC.	11/06/2000	011226/0563
				HI/FN, INC.	EXAR CORPORATION	09/03/2009	023180/0733
				APPTITUDE, INC	HI/FN, INC.	08/16/2012	028800/0034
				EXAR CORPORATION	PACKET INTELLIGENCE LLC	02/01/2013	029737/0613
	10684776	10/14/2003	6954789	HI/FN, INC.	EXAR CORPORATION	09/03/2009	023180/0733
				APPTITUDE, INC	HI/FN, INC	08/16/2012	028800/0034
				EXAR CORPORATION	PACKET INTELLIGENCE LLC	02/01/2013	029737/0613
	09609179	06/30/2000	6665725	Inventors	APPTITUDE, INC.	10/24/2000	011258/0647
				HI/FN, INC.	EXAR CORPORATION	09/03/2009	023180/0733
				APPTITUDE, INC	HI/FN, INC.	08/16/2012	028800/0034
				EXAR CORPORATION	PACKET INTELLIGENCE LLC	02/01/2013	029737/0613
	09608126	06/30/2000	6839751	Inventors	APPTITUDE, INC.	11/06/2000	011226/0879
				HI/FN, INC.	EXAR CORPORATION	09/03/2009	023180/0733
				APPTITUDE, INC	HI/FN, INC.	08/16/2012	028800/0034
				EXAR CORPORATION	PACKET INTELLIGENCE LLC	02/01/2013	029737/0613

Attorney Docket No.	Application No.	Date Filed	Patent No.	Assignor	Assignee	Assignment Recordation	
						Date	Reel / Frame
	09608266	06/30/2000	6771646	Inventors	APPTITUDE, INC.	10/24/2000	011258/0672
				HI/FN, INC.	EXAR CORPORATION	09/03/2009	023180/0733
				APPTITUDE, INC	HI/FN, INC.	08/16/2012	028800/0034
				BARES, WILLIAM H.	EXAR CORPORATION	08/16/2012	028799/0658
				EXAR CORPORATION	PACKET INTELLIGENCE LLC	02/01/2013	029737/0613
	09608267	06/30/2000	6789116	Inventors	APPTITUDE, INC.	10/24/2000	011253/0063
				HI/FN, INC.	EXAR CORPORATION	09/03/2009	023180/0733
				APPTITUDE, INC	HI/FN, INC.	08/16/2012	028800/0034
				BARES, WILLIAM H	EXAR CORPORATION	08/16/2012	028799/0658
				EXAR CORPORATION	PACKET INTELLIGENCE LLC	02/01/2013	029737/0613
	10828776	04/20/2004	7299282	HI/FN, INC.	EXAR CORPORATION	09/03/2009	023180/0733
				APPTITUDE, INC	HI/FN, INC.	08/16/2012	028800/0034
				BARES, WILLIAM H	EXAR CORPORATION	08/16/2012	028799/0658
				EXAR CORPORATION	PACKET INTELLIGENCE LLC	02/01/2013	029737/0613
	60/141,903	06-30-1999		EXAR CORPORATION	PACKET INTELLIGENCE LLC	02/01/2013	029737/0699

Attorney Docket No.	Application No.	Date Filed	Patent No.	Assignor	Assignee	Assignment Recordation	
						Date	Reel / Frame
	08/914,043	08-05-1997	6320846	STRUCTURED INTERNETWORKS, INC	SILICON VALLEY BANK	06/03/1998	009231/0554
				LIL DZUNG-JI HSU, JACK	STRUCTURED INTERNETWORKS	03/22/1999	009847/0812
				STRUCTURED INTERNETWORKS, INC.	HI/FN, INC.	04/26/2001	011791/0415
				HI/FN, INC.	EXAR CORPORATION	09/03/2009	023180/0733
				EXAR CORPORATION	PACKET INTELLIGENCE LLC	02/01/2013	029737/0613
	09/956,487	09-19-2001	6816459	EXAR CORPORATION	PACKET INTELLIGENCE LLC	02/01/2013	029737/0613
	11/257,485	10-24-2005	7260558	EXAR CORPORATION	PACKET INTELLIGENCE LLC	02/01/2013	029737/0613

Electronic Acknowledgement Receipt

EFS ID:	15493777
Application Number:	10684776
International Application Number:	
Confirmation Number:	3352
Title of Invention:	METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK
First Named Inventor/Applicant Name:	Russell S. Dietz
Customer Number:	21921
Filer:	Lawrence Aaronson/Karen Montgomery
Filer Authorized By:	Lawrence Aaronson
Attorney Docket Number:	APPT-001-1-1
Receipt Date:	11-APR-2013
Filing Date:	14-OCT-2003
Time Stamp:	15:05:26
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Power of Attorney	10354_001GEN_POA_Packet_Intelligence_.pdf	2335876 1f3a0173fa51a833481db928627bb153c4a228a8	no	5

Warnings:

Information:

Total Files Size (in bytes):	2335876

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
10/684,776	10/14/2003	Russell S. Dietz	

CONFIRMATION NO. 3352

POA ACCEPTANCE LETTER

OC000000060412762

96039
Meunier Carlin & Curfman LLC
817 W. Peachtree Street, Suite 500
Atlanta, GA 30308

Date Mailed: 04/15/2013

NOTICE OF ACCEPTANCE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 04/11/2013.

The Power of Attorney in this application is accepted. Correspondence in this application will be mailed to the above address as provided by 37 CFR 1.33.

/rmturner myles/

Office of Data Management, Application Assistance Unit (571) 272-4000, or (571) 272-4200, or 1-888-786-0101

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
10/684,776	10/14/2003	Russell S. Dietz	APPT-001-1-1

CONFIRMATION NO. 3352

POWER OF ATTORNEY NOTICE

OC000000060412727

21921
DOV ROSENFELD
5507 COLLEGE AVE
SUITE 2
OAKLAND, CA 94618

Date Mailed: 04/15/2013

NOTICE REGARDING CHANGE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 04/11/2013.

- The Power of Attorney to you in this application has been revoked by the assignee who has intervened as provided by 37 CFR 3.71. Future correspondence will be mailed to the new address of record(37 CFR 1.33).

/rmturner myles/

Office of Data Management, Application Assistance Unit (571) 272-4000, or (571) 272-4200, or 1-888-786-0101

"FEE ADDRESS" INDICATION FORM

Address to:
Mail Stop M Correspondence
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

- OR -

Fax to:
571-273-6500

INSTRUCTIONS: The issue fee must have been paid for application(s) listed on this form. In addition, only an address represented by a Customer Number can be established as the fee address for maintenance fee purposes (hereafter, fee address). A fee address should be established when correspondence related to maintenance fees should be mailed to a different address than the correspondence address for the application. **When to check the first box below**: If you have a Customer Number to represent the fee address. **When to check the second box below**: If you have no Customer Number representing the desired fee address, in which case a completed Request for Customer Number (PTO/SB/125) must be attached to this form. For more information on Customer Numbers, see the Manual of Patent Examining Procedure (MPEP) § 403.

For the following listed application(s), please recognize as the "Fee Address" under the provisions of 37 CFR 1.363 the address associated with:

☑ Customer Number: 96039

OR

☐ The attached Request for Customer Number (PTO/SB/125) form.

PATENT NUMBER (if known)	APPLICATION NUMBER
6,954,789	10/684,776

Completed by (check one):

☐ Applicant/Inventor

/Lawrence A. Aaronson/
Signature

☑ Attorney or Agent of record 38,369 (Reg. No.)

Lawrence A. Aaronson
Typed or printed name

☐ Assignee of record of the entire interest. See 37 CFR 3.71. Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96)

404.645.7700
Requester's telephone number

☐ Assignee recorded at Reel ________ Frame________

July 17, 2013
Date

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more that one signature is required, see below*.

☑ *** Total of** 1 **forms are submitted.**

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Electronic Acknowledgement Receipt

EFS ID:	16340031
Application Number:	10684776
International Application Number:	
Confirmation Number:	3352
Title of Invention:	METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK
First Named Inventor/Applicant Name:	Russell S. Dietz
Customer Number:	96039
Filer:	Lawrence Aaronson/Sharon Etelman
Filer Authorized By:	Lawrence Aaronson
Attorney Docket Number:	
Receipt Date:	17-JUL-2013
Filing Date:	14-OCT-2003
Time Stamp:	13:45:59
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Change of Address	10354_002US2_2013_07_17_Fee_Address_Indication_Form.pdf	312888 9d678ea5f686ce6b0b05df279cb888c53e2c2eaa	no	2

Warnings:

Information:

Total Files Size (in bytes):	312888

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/44 (09-07)
Approved for use through 08/31/2013. OMB 0651-0033
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

(Also Form PTO-1050)

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 6,954,789

APPLICATION NO.: 10/684,776

ISSUE DATE : October 11, 2005

INVENTOR(S) : Russell S. Dietz, Joseph R. Maixner, Andrew A. Koppenhaver, William H. Bares, Haig A. Sarkissian, James F. Torgerson

It is certified that an error appears or errors appear in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

IN THE CLAIMS:

Column 35, line 47, claim 6, change “packers” to --packets--.

Column 36, line 18, claim 15, change “entiy” to --entry--.

Column 37, line 32, claim 26, change “method” to --monitor--.

Column 37, line 40, claim 27, change “method” to --monitor--.

Column 37, lines 55 and 56, claim 29, change “for the initial state of the new flow in the case that the packet is from an existing flow” to --for the initial state of the new flow in the case that the packet is not from an existing flow--.

MAILING ADDRESS OF SENDER (Please do not use customer number below):

Meunier Carlin & Curfman, LLC
817 W. Peachtree St., NW, Suite 500
Atlanta, GA 30308

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Electronic Patent Application Fee Transmittal

Application Number:	10684776
Filing Date:	14-Oct-2003
Title of Invention:	METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK
First Named Inventor/Applicant Name:	Russell S. Dietz
Filer:	Lawrence Aaronson/Karen Carroll
Attorney Docket Number:	

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Certificate of Correction	1811	1	100	100
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
	Total in USD ($)			**100**

Electronic Acknowledgement Receipt

EFS ID:	16761787
Application Number:	10684776
International Application Number:	
Confirmation Number:	3352
Title of Invention:	METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK
First Named Inventor/Applicant Name:	Russell S. Dietz
Customer Number:	96039
Filer:	Lawrence Aaronson/Karen Carroll
Filer Authorized By:	Lawrence Aaronson
Attorney Docket Number:	
Receipt Date:	04-SEP-2013
Filing Date:	14-OCT-2003
Time Stamp:	15:46:01
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Electronic Funds Transfer
Payment was successfully received in RAM	$100
RAM confirmation Number	2768
Deposit Account	
Authorized User	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)

1	Transmittal Letter	6954789_Request_for_Certificate_of_Correction_Tns.pdf	77792 9d7a35b7721197ea6c255e81ca221171df7e216e	no	1
Warnings:					
Information:					
2	Request for Certificate of Correction	6954789_PTO_SB_44_Certificate_of_Correction.pdf	165213 1776d68f1b82a3f239f86f564c76ea3e615ab00a	no	2
Warnings:					
Information:					
3	Fee Worksheet (SB06)	fee-info.pdf	29891 159a298d679d2d5caf57d5dee0ca40da006c85fd	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	272896		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

DOCKET NO.: 10354-001GEN **PATENT**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re Application of:

Russell S. Dietz, Joseph R. Maixner,

Andrew A. Koppenhaver, William H. Bares, **Confirmation No.: 3352**

Haig A. Sarkissian, James F. Torgerson

Application No.: 10/684,776 **Group Art Unit: 2157**

Patent No.: 6,954,789 **Issue Date: October 11, 2005**

Filing Date: October 14, 2003 **Examiner: Moustafa M. Meky**

For: METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK

Commissioner for Patents
Office of Patent Publications
ATTN: Certificate of Correction Branch
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

REQUEST FOR CERTIFICATE OF CORRECTION
PURSUANT TO 37 CFR § 1.322 & 37 CFR § 1.323

It is respectfully requested that a Certificate of Correction be issued for the above-identified patent. The patent has **five (5)** errors that are the fault of the applicant. Applicant's errors occurred in good faith and are of a clerical or typographical nature, or minor character, and are not believed to constitute new matter or require examination.

Enclosed herewith please find a completed Certificate of Correction form.

The fee in the amount of **$100.00** is attached.

Respectfully submitted,

Date: September 4, 2013

/Lawrence A. Aaronson/
Lawrence Aaronson
Reg. No. 38,369

Meunier Carlin & Curfman, LLC
817 W. Peachtree St., NW
Suite 500
Atlanta, GA 30308
phone: (404) 645-7713
fax: (404) 645-7707

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,954,789 B2
APPLICATION NO. : 10/684776
DATED : October 11, 2005
INVENTOR(S) : Russell S. Dietz et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

IN THE CLAIMS:

Column 35, line 47, claim 6, change “packers” to --packets--.

Column 36, line 18, claim 15, change “entiy” to --entry--.

Column 37, line 32, claim 26, change “method” to --monitor--.

Column 37, line 40, claim 27, change “method” to --monitor--.

Column 37, lines 55 and 56, claim 29, change “for the initial state of the new flow in the case that the packet is from an existing flow” to --for the initial state of the new flow in the case that the packet is not from an existing flow--.

Signed and Sealed this
First Day of October, 2013

Teresa Stanek Rea
Deputy Director of the United States Patent and Trademark Office

AO 120 (Rev. 08/10)

TO: **Mail Stop 8** **Director of the U.S. Patent and Trademark Office** **P.O. Box 1450** **Alexandria, VA 22313-1450**	**REPORT ON THE FILING OR DETERMINATION OF AN ACTION REGARDING A PATENT OR TRADEMARK**

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court Eastern District of Texas on the following

☐ Trademarks or ☑ Patents. (☐ the patent action involves 35 U.S.C. § 292.):

DOCKET NO.	DATE FILED	U.S. DISTRICT COURT
2:14-cv-00252	3/24/2014	Eastern District of Texas

PLAINTIFF	DEFENDANT
PACKET INTELLIGENCE LLC	CISCO SYSTEMS, INC.

PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1 6,651,099	11/18/2003	Packet Intelligence LLC
2 6,665,725	12/16/2003	Packet Intelligence LLC
3 6,771,646	8/3/2004	Packet Intelligence LLC
4 6,839,751	1/4/2005	Packet Intelligence LLC
5 6,954,789	10/11/2005	Packet Intelligence LLC

In the above—entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY ☐ Amendment ☐ Answer ☐ Cross Bill ☐ Other Pleading

PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1 *		*Patent listed above.
2		
3		
4		
5		

In the above—entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT

CLERK	(BY) DEPUTY CLERK	DATE

Copy 1—Upon initiation of action, mail this copy to Director Copy 3—Upon termination of action, mail this copy to Director
Copy 2—Upon filing document adding patent(s), mail this copy to Director Copy 4—Case file copy

AO 120 (Rev. 08/10)

TO: Mail Stop 8 Director of the U.S. Patent and Trademark Office P.O. Box 1450 Alexandria, VA 22313-1450	**REPORT ON THE FILING OR DETERMINATION OF AN ACTION REGARDING A PATENT OR TRADEMARK**

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court Eastern District of Texas on the following

☐ Trademarks or ☑ Patents. (☐ the patent action involves 35 U.S.C. § 292.):

DOCKET NO.	DATE FILED	U.S. DISTRICT COURT
2:16-cv-00147	2/17/2016	Eastern District of Texas
PLAINTIFF		DEFENDANT
Packet Intelligence LLC		Sandvine Corporation Sandvine Incorporated ULC

PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1 6,651,099	11/18/2003	Packet Intelligence LLC
2 6,665,725	12/16/2003	Packet Intelligence LLC
3 6,771,646	8/3/2004	Packet Intelligence LLC
4 6,839,751	1/4/2005	Packet Intelligence LLC
5 6,954,789	10/11/2005	Packet Intelligence LLC

In the above—entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY ☐ Amendment ☐ Answer ☐ Cross Bill ☐ Other Pleading	
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1		
2		
3		
4		
5		

In the above—entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT

CLERK	(BY) DEPUTY CLERK	DATE

Copy 1—Upon initiation of action, mail this copy to Director Copy 3—Upon termination of action, mail this copy to Director
Copy 2—Upon filing document adding patent(s), mail this copy to Director Copy 4—Case file copy

AO 120 (Rev. 08/10)

TO: **Mail Stop 8** **Director of the U.S. Patent and Trademark Office** **P.O. Box 1450** **Alexandria, VA 22313-1450**	**REPORT ON THE FILING OR DETERMINATION OF AN ACTION REGARDING A PATENT OR TRADEMARK**

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court Eastern District of Texas on the following

☐ Trademarks or ☑ Patents. (☐ the patent action involves 35 U.S.C. § 292.):

DOCKET NO. 2:16-cv-230	DATE FILED 3/15/2016	U.S. DISTRICT COURT Eastern District of Texas
PLAINTIFF Packet Intelligence LLC		DEFENDANT NetScout Systems, Inc. Tektronix Communications Tektronix Texas, LLC
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1 6,651,099	11/18/2003	Packet Intelligence LLC
2 6,665,725	12/16/2003	Packet Intelligence LLC
3 6,771,646	8/3/2004	Packet Intelligence LLC
4 6,839,751	1/4/2005	Packet Intelligence LLC
5 6,954,789	10/11/2005	Packet Intelligence LLC

In the above—entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY ☐ Amendment ☐ Answer ☐ Cross Bill ☐ Other Pleading	
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1		
2		
3		
4		
5		

In the above—entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT

CLERK	(BY) DEPUTY CLERK	DATE

Copy 1—Upon initiation of action, mail this copy to Director Copy 3—Upon termination of action, mail this copy to Director
Copy 2—Upon filing document adding patent(s), mail this copy to Director Copy 4—Case file copy

AO 120 (Rev. 08/10)

TO: **Mail Stop 8** **Director of the U.S. Patent and Trademark Office** **P.O. Box 1450** **Alexandria, VA 22313-1450**	**REPORT ON THE FILING OR DETERMINATION OF AN ACTION REGARDING A PATENT OR TRADEMARK**

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court Eastern District of Texas on the following

☐ Trademarks or ☑ Patents. (☐ the patent action involves 35 U.S.C. § 292.):

DOCKET NO.	DATE FILED	U.S. DISTRICT COURT
2:16-cv-230	3/15/2016	Eastern District of Texas

PLAINTIFF	DEFENDANT
Packet Intelligence LLC	NetScout Systems, Inc. Tektronix Communications Tektronix Texas, LLC

PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1 6,651,099	11/18/2003	Packet Intelligence LLC
2 6,665,725	12/16/2003	Packet Intelligence LLC
3 6,771,646	8/3/2004	Packet Intelligence LLC
4 6,839,751	1/4/2005	Packet Intelligence LLC
5 6,954,789	10/11/2005	Packet Intelligence LLC

In the above—entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY ☐ Amendment ☐ Answer ☐ Cross Bill ☐ Other Pleading	
PATENT OR TRADEMARK NO.	**DATE OF PATENT OR TRADEMARK**	**HOLDER OF PATENT OR TRADEMARK**
1		
2		
3		
4		
5		

In the above—entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT

CLERK	(BY) DEPUTY CLERK	DATE

Copy 1—Upon initiation of action, mail this copy to Director Copy 3—Upon termination of action, mail this copy to Director
Copy 2—Upon filing document adding patent(s), mail this copy to Director Copy 4—Case file copy

OIPE JUL 13 2017 PATENT & TRADEMARK OFFICE

7-14-17 Box M

July 12, 2017

Mail Stop Petition
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Re: Deficiency Payment under 37 CFR 1.28(c) for U.S. Patent No. 6,954,789; U.S. Patent Application No. 10/684,776; Conf. No.: 3352

Dear Sir or Madam:

I am outside counsel for Packet Intelligence LLC, the owner of the above-identified patent.

It has come to our attention that Packet Intelligence LLC made a single deficient maintenance fee payment for the above-identified patent's 11.5 year maintenance fee when it paid the small entity fee, i.e. $3,700.00, rather than the large entity fee, i.e., $7400.00. This error in the fee payment was made in good faith.

This deficient $3700.00 payment for the 11.5 year maintenance fee was submitted March 30, 2017.

Please find enclosed an additional payment of $3700.00 from the patent owner, which constitutes both the deficiency amount owed for the 11.5 year maintenance fee and the total deficiency amount owed for the above-identified patent.

This correspondence also serves as notice that Packet Intelligence LLC is no longer entitled to small entity status. Please contact the undersigned at the above address with any questions regarding this payment.

Respectfully submitted,

Lawrence A. Aaronson
Registration No. 38,369

2017 JUL 17 AM 9:07

U.S.P.T.O. RECEIPTS ACCOUNTING DIVISION

07/20/2017 DALLEN 00000006 6954789
01 FC:1599 3700.00 OP

OPAP
JUL 1 3 2017
PATENT & TRADEMARK OFFICE

United States Patent and Trademark Office

Office of the Commissioner for Patents

Maintenance Fee Statement

CURRENT MAINTENANCE FEE ADDRESS	CUSTOMER #	ENTITY STATUS	STATEMENT GENERATED
MEUNIER CARLIN & CURFMAN LLC 999 PEACHTREE STREET NE SUITE 1300 ATLANTA, US 30309	96039	SMALL	07/03/2017 17:34:59

Invention

METHOD AND APPARATUS FOR MONITORING TRAFFIC IN A NETWORK

PATENT #	APPLICATION #	FILING DATE	ISSUE DATE
6954789	10684776	10/14/2003	10/11/2005

Payment Details

PAYMENT DATE	DATE POSTED	TRANSACTION ID	ATTORNEY DOCKET #	TOTAL PAYMENT
03/30/2017	03/30/2017	033017INTMTFEE00007671504623		$3,700.00

Fee Code.	Description	Sale ID	Fee Amount
2553	MAINTENANCE FEE DUE AT 11.5 YEARS	033017INTMTFEE00007671	$3,700.00

According to the records of the United States Patent and Trademark Office (USPTO), the maintenance fee and any necessary surcharge have been timely paid for the patent listed above. The payment shown above is subject to actual collection. If the payment is refused or charged back by a financial institution, the payment will be void and the maintenance fee and any necessary surcharge unpaid.

P.O. Box 1450. Alexandria. VA 22313-1450 • www.uspto.gov

Document code: WFEE

United States Patent and Trademark Office
Sales Receipt for Accounting Date: 12/06/2017

GARIAS SALE #00000002 Mailroom Dt: 07/13/2017 10684776
01 FC : 1559 3,700.00 OP

Document code: WFEE

United States Patent and Trademark Office
Sales Receipt for Accounting Date: 12/06/2017

GARIAS ADJ #00000002 Mailroom Dt: 07/13/2017
Seq No: 6 Sales Acctg Dt: 07/20/2017 10684776
01 FC : 1599 -3700.00 OP

Trials@uspto.gov
571-272-7822

Paper 8
Entered: July 26, 2017

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SANDVINE CORPORATION and SANDVINE INCORPORATED ULC,
Petitioner,

v.

PACKET INTELLIGENCE, LLC,
Patent Owner.

Case IPR2017-00629
Patent 6,954,789 B2

Before ELENI MANTIS MERCADER, JUSTIN T. ARBES, and
WILLIAM M. FINK, *Administrative Patent Judges*.

FINK, *Administrative Patent Judge*.

DECISION
Denying Institution of *Inter Partes* Review
37 C.F.R. § 42.108

Sandvine Corporation and Sandvine Incorporated ULC (collectively, "Petitioner") filed a Petition (Paper 2, "Pet.") requesting *inter partes* review of claims 19–43 of U.S. Patent No. 6,954,789 B2 (Ex. 1004, "the '789 patent") pursuant to 35 U.S.C. § 311(a). Patent Owner Packet Intelligence, LLC filed a Preliminary Response (Paper 6, "Prelim. Resp.") pursuant to 35 U.S.C. § 313. Pursuant to 35 U.S.C. § 314(a), the Director may not authorize an *inter partes* review unless the information in the petition and preliminary response "shows that there is a reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition." For the reasons that follow, we have decided not to institute an *inter partes* review.

I. BACKGROUND

A. The '789 Patent[1]

The '789 patent discloses "[a] monitor for and a method of examining packets passing through a connection point on a computer network." Ex. 1002, Abstract. The '789 patent explains that there was a need in the art for "a real-time network monitor that can provide alarms notifying selected users of problems that may occur with the network or site." *Id.* at col. 2, ll. 3–5. The disclosed monitor receives packets passing in either direction through its connection point on the network and "elucidate[s] what application programs are associated with each packet" by extracting information from the packet, using selected parts of the extracted

[1] Petitioner challenges different claims of the '789 patent in Case IPR2017-00630. Petitioner also challenges patents related to the '789 patent in Cases IPR2017-00450, IPR2017-00451, IPR2017-00769, IPR2017-00862, and IPR2017-00863.

information to identify this packet as part of a flow, "build[ing] a unique flow signature (also called a 'key') for this flow," and "matching this flow in a database of known flows 324." *Id.* at col. 9, ll. 6–9, col 13, ll. 21–28, col. 13, ll. 60–65.

Figure 3 of the '789 patent is reproduced below.

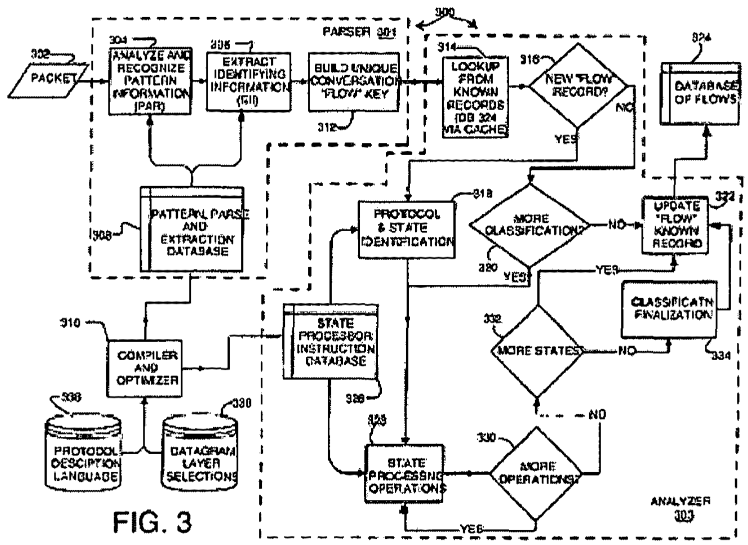


FIG. 3

Figure 3 depicts various components of network packet monitor 300, including parser subsystem 301, analyzer subsystem 303, and database of known flows 324. *Id.* at col. 11, l. 50–col. 13, l. 65. Parser subsystem 300 "parses the packet and determines the protocol types and associated headers for each protocol layer that exists in the packet 302," "extracts characteristic portions (signature information) from the packet 302," and builds the "unique flow signature (also called a 'key') for this flow." *Id.* at col. 12, l. 19–col. 13, l. 28, col. 33, l. 30–col. 34, l. 33 (describing an example of how the disclosed monitor builds signatures and flow states in the context of a Sun Remote Procedure Call (RPC), where, after all of the required processing, "KEY-2 may . . . be used to recognize packets that are in any way associated with the application 'a[2]'"), Fig. 2.

Analyzer system 303 then determines whether the packet has a matching flow-entry in database of flows 324, and processes the packet accordingly, including, for example, determining whether the packet belongs to an existing conversational flow or a new (i.e., not previously encountered) flow and, in the case of the latter, performing state processing to determine whether the conversational flow has been "fully characterized" and should be finalized. *Id.* at col. 13, l. 60–col. 16, l. 52. The '789 patent discloses that

> [f]uture packets that are part of the same conversational flow have their state analysis continued from a previously achieved state. When enough packets related to an application of interest have been processed, a final recognition state is ultimately reached, i.e., a set of states has been traversed by state analysis to completely characterize the conversational flow. The signature for that final state enables each new incoming packet of the same conversational flow to be individually recognized in real time.
>
> In this manner, one of the great advantages of the present invention is realized. Once a particular set of state transitions has been traversed for the first time and ends in a final state, a short-cut recognition pattern—a signature—[c]an be generated that will key on every new incoming packet that relates to the conversational flow. Checking a signature involves a simple operation, allowing high packet rates to be successfully monitored on the network.

Id. at col. 16, ll. 17–34.

B. Illustrative Claim

Claim 19 of the '789 patent[2] recites:

> 1. A packet monitor for examining packets passing through a connection point on a computer network, each packets conforming to one or more protocols, the monitor comprising:
>
> (a) a packet acquisition device coupled to the connection point and configured to receive packets passing through the connection point;
>
> (b) an input buffer memory coupled to and configured to accept a packet from the packet acquisition device;
>
> (c) a parser subsystem coupled to the input buffer memory and including a slicer, the parsing subsystem configured to extract selected portions of the accepted packet and to output a parser record containing the selected portions;
>
> (d) a memory for storing a database comprising none or more flow-entries for previously encountered conversational flows, each flow-entry identified by identifying information stored in the flow-entry;
>
> (e) a lookup engine coupled to the output of the parser subsystem and to the flow-entry memory and configured to lookup whether the particular packet whose parser record is output by the parser subsystem has a matching flow-entry, the looking up using at least some of the selected packet portions and determining if the packet is of an existing flow; and
>
> (f) a flow insertion engine coupled to the flow-entry memory and to the lookup engine and configured to create a flow-entry in the flow-entry database, the flow-entry including identifying information for future packets to be identified with the new flow-entry, the lookup engine configured such that if the packet is of an existing flow, the monitor classifies the packet as belonging to the found existing flow; and if the packet is of a new flow, the flow insertion engine stores a new flow-

[2] Claims 6, 7, 15, 23, 26, 27, and 29 of the '789 patent were corrected in Certificates of Correction dated March 7, 2006, and October 1, 2013.

entry for the new flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry,

wherein the operation of the parser subsystem depends on one or more of the protocols to which the packet conforms.

C. The Prior Art

Petitioner relies on the following prior art:

U.S. Patent No. 5,530,834, issued June 25, 1996 (Ex. 1011, "Colloff");

U.S. Patent No. 5,793,954 issued August 11, 1998 (Ex. 1012, "Baker");

U.S. Patent No. 6,115,393, filed July 21, 1995, issued Sept. 5, 2000 (Ex. 1007, "Engel");[3] and

U.S. Patent No. 6,182,146 B1, filed June 27, 1997, issued Jan. 30, 2001 (Ex. 1010, "Graham-Cumming").

D. The Asserted Grounds

Petitioner challenges claims 19–43 of the '789 patent on the following grounds:

Reference(s)	Basis	Claims Challenged
Engel	35 U.S.C. § 102(e)[4]	19–22, 25, 26, 29–31, 36–40, and 42

[3] Engel references Appendices I–VI filed with the originally filed application. *See, e.g.*, Ex. 1007, col. 1, ll. 10–15, col. 5, l. 52–col. 6, l. 3; Ex. 1008 (Appendices I–V); Ex. 1009 (Appendix VI).

[4] The Leahy-Smith America Invents Act, Pub. L. No. 112-29, 125 Stat. 284 (2011) ("AIA"), amended 35 U.S.C. §§ 102 and 103. Because the challenged claims of the '789 patent have an effective filing date before the effective date of the applicable AIA amendments, we refer to the pre-AIA versions of 35 U.S.C. §§ 102 and 103.

Reference(s)	Basis	Claims Challenged
Engel and Baker	35 U.S.C. § 103(a)	23 and 24
Engel and Graham-Cumming	35 U.S.C. § 103(a)	27, 28, 32, 41, and 43
Engel and Colloff	35 U.S.C. § 103(a)	33–35

E. Claim Interpretation

In an *inter partes* review, claim terms in an unexpired patent are given their "broadest reasonable construction in light of the specification of the patent in which they appear." 37 C.F.R. § 42.100(b). Under the broadest reasonable construction standard, claim terms are given their ordinary and customary meaning, as would be understood by one of ordinary skill in the art in the context of the entire disclosure. *In re Translogic Tech., Inc.*, 504 F.3d 1249, 1257 (Fed. Cir. 2007). Only terms in controversy need to be construed, and only to the extent necessary to resolve the controversy. *Vivid Techs., Inc. v. Am. Sci. & Eng'g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999).

Petitioner provides proposed constructions of the claim terms "conversational flow," "state of the flow," "state operations," and "parser record." Pet. 2–7. Patent Owner provides proposed constructions of the first three terms. Prelim. Resp. 23–30. Other than the term "conversational flow," discussed below, we determine that no claim term requires express construction to resolve the issues before us.

Claim 19 recites (emphasis added):

> (d) a memory for storing a database comprising none or more flow-entries for previously encountered *conversational flows*, each flow-entry identified by identifying information stored in the flow-entry;

Petitioner argues that "conversational flow" should be interpreted to mean "'the sequence of packets that are exchanged in any direction as a result of an activity,' where a 'conversational flow' is distinguished from a 'connection flow' in that some of the conversational flows 'involve more than one connection' or 'involve more than one exchange of packets between a client and server.'" Pet. 3. Petitioner acknowledges that the claims "require more than identifying and classifying 'only connection flows,'" citing statements in the '789 patent:

> Some prior art packet monitors classify packets into connection flows. The term "connection flow" is commonly used to describe all the packets involved with a single connection. A conversational flow, on the other hand, is the sequence of packets that are exchanged in any direction as a result of an activity—for instance, the running of an application on a server as requested by a client. It is desirable to be able to identify and classify conversational flows rather than only connection flows. The reason for this is that some conversational flows involve more than one connection, and some even involve more than one exchange of packets between a client and server.

Id. at 3–4 (quoting Ex. 1004, col. 2, ll. 42–53) (emphases omitted).

Relying on the same disclosure from the patent, Patent Owner argues that the term "conversational flow" is expressly defined as

> the sequence of packets that are exchanged in any direction as a result of an activity—for instance, the running of an application on a server as requested by a client—and where some conversational flows involve more than one connection, and some even involve more than one exchange of packets between a client and server.

Prelim. Resp. 24. Patent Owner further notes that Petitioner agreed to the above interpretation in the related district court case, and the district court adopted it. *Id.* at 24–25 (citing Ex. 2005, 6).

The parties' proposed interpretations are nearly identical. We agree with Patent Owner that the term "conversational flow" is expressly defined in the excerpt of the patent quoted above. Accordingly, for purposes of this Decision, we interpret "conversational flow" to mean the sequence of packets that are exchanged in any direction as a result of an activity (for instance, the running of an application on a server as requested by a client), where some conversational flows involve more than one connection, and some even involve more than one exchange of packets between a client and server.

II. DISCUSSION

A. Anticipation Ground Based on Engel

Petitioner contends that claims 19–22, 25, 26, 29–31, 36–40, and 42 are anticipated by Engel under 35 U.S.C. § 102(e), citing the testimony of Bill Lin, Ph.D., as support. Pet. 10–56 (citing Ex. 1006). We are not persuaded that Petitioner has established a reasonable likelihood of prevailing on its asserted ground for the reasons explained below.

1. Engel

Engel is directed to monitoring communications "in a network of nodes, each communication being effected by a transmission of one or more packets among two or more communicating nodes, and each communication complying with a predefined communication protocol." Ex. 1007, Abstract. "The contents of packets are detected passively and in real time, [and] communication information associated with multiple protocols is derived from the packet contents." *Id.* Such communication information "is associated with multiple layers of at least one of the protocols." *Id.* at col. 2, ll. 27–30. A network monitor "collects packets on the network and performs some degree of analysis to search for actual or potential problems and to maintain statistical information for use in later analysis." *Id.* at col. 6, ll. 52–65, Fig. 1. A Real Time Parser (RTP) performs layer-by-layer protocol parsing with appropriate routines. *Id.* at col. 11, ll. 25–30, col. 19, ll. 53–67, col. 21, ll. 15–30. "As each layer is parsed, the RTP invokes the appropriate functions in the statistics module (STATS) to update those statistical objects which must be changed." *Id.* at col. 11, ll. 35–37.

Figure 4 of Engel is reproduced below.

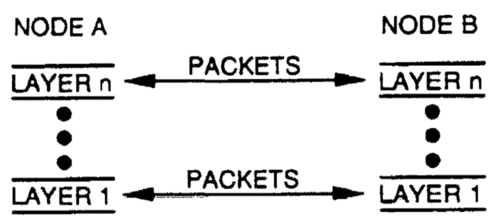


FIG 4

Figure 4 "illustrates the different layers of a communication between two nodes." *Id.* at col. 4, ll. 45–46. In Figure 4, "Nodes A and B are exchanging packets and are engaged in multiple dialogs." *Id.* at col. 9, ll. 31–33. "[A] dialog is a communication [at any layer] between a sender and a receiver, which is composed of one or more packets being transmitted between the two" and often "involve[s] exchanges in both directions." *Id.* at col. 9, ll. 19–24. As shown in Figure 4, "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." *Id.* at col. 9, ll. 37–41. Engel discloses keeping a dialog record that contains, *inter alia*, "the addresses of both ends of the dialog concatenated together to form a single address." *Id.* at col. 18, l. 63–col. 19, l. 4.

2. Analysis

Petitioner argues that Engel discloses all of the limitations of claim 19. Pet. 12–38. We focus on one limitation in particular, which is dispositive of Petitioner's asserted ground. Claim 19 recites:

> (d) a memory for storing a database comprising none or more flow-entries for previously encountered *conversational flows*, each flow-entry identified by identifying information stored in the flow-entry;[5]

For this limitation, Petitioner argues that Engel's STATS database is a database that stores "statistics relating to stations and dialogs encountered by the network monitor." *Id.* at 15. Petitioner contends that, through its packet processing and use of the STATS database, Engel monitors "conversational flows" in two different ways: via "Application Level Dialogs" and "Application-Specific Server Statistics." *Id.* at 11–12, 15–28.

At the outset, we note that Petitioner's arguments are premised on its position that related U.S. Patent No. 6,839,751 B1 (Ex. 1002, "the '751 patent") "provides conversational flow examples where flows are determined and metrics reported 'for a given application *and either* a

[5] We note that although claim 19 recites "a database comprising *none or more flow-entries* for previously encountered conversational flows" (emphasis added), the claim requires "conversational flows" at least due to the language of limitation (f). Limitation (f) covers the situation where there is an "existing" conversational flow and the situation of storing a new flow-entry in the database for a "new" conversational flow.

specific Client-Server Pair *or* a specific Server and all of its clients.'"[6] *Id.* at 15 (quoting Ex. 1002, col. 34, ll. 58–60). As Patent Owner points out, however, the cited portion of the '751 patent relates to tracking various statistics and does not mention conversational flows. *See* Prelim. Resp. 47–48; *see* Ex. 1002, col. 31, l. 52–col. 49, l. 65 (describing "how the monitor of the invention can be used to monitor the Quality of Service (QOS) by providing QOS Metrics"). The specific language quoted by Petitioner pertains to one such metric, "CSTraffic," which "contains information about the volume of traffic measured for a given application and either a specific Client-Server Pair or a specific Server and all of its clients." Ex. 1002, col. 34, ll. 55–60. However, we do not see how this language supports Petitioner's position. Monitoring statistics is not the same as monitoring a conversational flow, which necessarily identifies a "sequence of packets," under our interpretation. *See supra* Section I.E. In other words, tracking statistics for a given application between a particular client and server, or between a particular server and all of its clients, is not sufficient in and of itself; a conversational flow identifies a sequence of packets exchanged between two nodes as a result of specific activity (e.g., a "sequence of packets that are exchanged in any direction as a result of an activity").

We now turn to Petitioner's two arguments as to how Engel allegedly discloses conversational flows. First, Petitioner contends that Engel teaches

---

[6] The '789 patent states that the '751 patent is incorporated by reference. Ex. 1004, col. 1, ll. 28–33. Also, both patents claim the benefit of U.S. Provisional Patent Application No. 60/141,903 and have some of the same disclosure in their written descriptions.

"Application Level Dialogs." Pet. 15–22. Petitioner argues that Engel "considers the nature of the communication (e.g., the application program being invoked) in addition to the Client-Server Pair involved in the communication in identifying an application level dialog, rendering such dialog a conversational flow within the meaning of the '789 Patent." *Id.* at 15–16 (citing Ex. 1007, col. 9, l. 27–col. 10, l. 3). Petitioner relies on Engel's definition of "dialog" as "the exchange of messages and the associated meaning and state that is inherent in any particular exchange at any layer." *Id.* at 16 (quoting Ex. 1007, col. 9, ll. 24–27). According to Petitioner, Engel discloses "creating and maintaining dialogs" and collecting associated statistics at different layers. *Id.* at 16–17. In particular, Petitioner asserts that the dialog at the application level "is an exchange of one or more packets in any direction between two nodes as a result of an activity (e.g., [a Network File System (NFS)] mount request)." *Id.* at 17 (citing Ex. 1007, col. 9, l. 58–col. 10, l. 3). Petitioner concludes that tracking an application-level dialog constitutes tracking a conversational flow. *Id.* at 17–21 (citing Ex. 1007, Fig. 7C, showing dialog records that point to dialog statistics data structures in the STATS database, and corresponding portions of the source code in Ex. 1009).

Petitioner further relies on Dr. Lin's testimony to assert that, to the extent that conversational flows require the tracking of multiple connections, "Engel's application level lookup_dialog routines search for existing dialogs based on application and the client-server IP address pair of the communication." *Id.* at 20–21 (citing Ex. 1006 ¶ 86). Petitioner also contends, based on Dr. Lin's testimony, that "Engel's deallocation routines deallocate dialog records when the period of inactivity for a dialog address pair (measured from the last time an application-specific packet (e.g., NFS) between the IP address pair of the dialog was seen) has exceeded a defined application-specific threshold." *Id.* at 21 (citing Ex. 1007, col. 17, ll. 2–16; Ex. 1006 ¶¶ 106–107).

Patent Owner responds that Engel's dialogs are different from conversational flows because they are merely "collections of statistics for each [Open Systems Interconnection (OSI)] layer across packets (regardless of application origin)" and do not relate packets that are the result of an application activity. Prelim. Resp. 46–47 (citing Ex. 1007, col. 4, ll. 23–32; Ex. 2001 ¶¶ 78–79). For example, noting Petitioner's statement that "[e]ach application level dialog tracks all application activity between the client-server pair occurring on any connections until the dialog data structures are deallocated for reuse," Patent Owner contends that relating all application activity between a client and server is insufficient because a conversational flow requires relating flows for specific application activities. *Id.* at 50–51 (quoting Pet. 21; citing Ex. 2001 ¶¶ 80–81). We agree with Patent Owner.

Engel explains that a "dialog" is

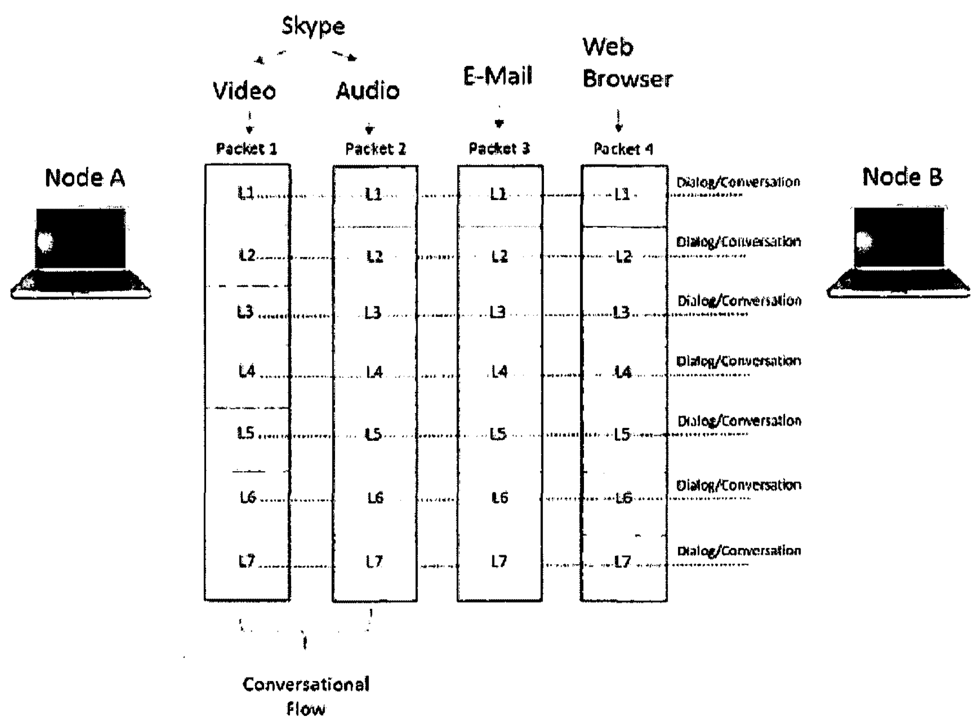
> a concept which provides a useful way of conceptualizing, organizing and displaying information about the performance of a network—for any protocol and for *any layer* of the multi-level protocol stack.
>
> As noted above, the basic unit of information in communication is a packet. A packet conveys meaning 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.
>
> 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.

Ex. 1007, col. 9, ll. 9–43 (emphases added).

As shown in Figure 4 (reproduced above in Section II.A.1), Nodes A and B engage in a dialog at each layer (e.g., the application layer). Thus, Engel monitors communications exchanged between the nodes at a particular layer, and tracks statistics for all communications at that layer (to assist in diagnosing network problems and improving network performance). *See, e.g.*, *id.* at col. 4, ll. 24–32 (Engel "organizes and presents network performance statistics in terms of dialogs which are occurring at any desired level of the communication"). Petitioner, however, does not point to any disclosure in Engel indicating that its disclosed system relates information from one packet to another as pertaining to a specific application activity. In other words, in Engel there is exchange of packets in the application layer constituting a dialog (*see, e.g.*, *id.* at col. 9, ll. 37–43) and collection of respective statistics for that dialog (*see, e.g.*, *id.* at col. 4, ll. 24–32). However, even if application activity causes the dialog to be created, Petitioner does not direct us to teachings or disclosure of identifying a sequence of packets as being part of a specific application activity. Moreover, Patent Owner notes that in Engel "[k]eeping statistics for protocol layers may be temporarily suspended when parsing and statistics gathering is not rapid enough to match the rate of packets to be parsed," which further suggests Engel does not relate sequences of packets to applications because some packets may be lost. Prelim. Resp. 37 (quoting Ex. 1007, col. 3, ll. 12–14; Ex. 2001 ¶ 74).

Patent Owner provides the following illustration on page 42 of its Preliminary Response:



The figure above depicts four packets exchanged between Nodes A and B, with each row representing a layer of the OSI model. Engel treats each layer separately. For example, packets 1 and 2 may both result from the same application (e.g., video and audio traffic using Skype), but Engel would not link them as being part of a single conversational flow. *See id.* at 42–43, 50–51 ("Engel is not concerned with which application a specific packet originated from or which packets relate to one another based on the application that generated them."). Patent Owner's explanation is consistent with the disclosure of Engel and is supported by the testimony of its declarant, Kevin C. Almeroth, Ph.D. *See, e.g.*, Ex. 1007, col. 9, ll. 9–43 ("a dialog is a communication between a sender and a receiver, which is composed of one or more packets being transmitted between the two," where, for example, "Layer n of Node A is having a dialog with Layer n of node B"); col. 9, l. 67–col. 10, l. 3 ("If in fact there exists more than one communication thread between Nodes A and B, then these would represent separate, different dialogs."); Ex. 2001 ¶¶ 65–85.

Petitioner's position appears to be that conversational flows exist in Engel simply because communications exist at the application layer and because those communications result from some application activity. *See* Pet. 15–22. As explained in connection with Patent Owner's illustration above, however, we do not see—and Petitioner does not point to—anything in Engel indicating that it links communications by application (as opposed to by layer and client-server pair) as our interpretation of "conversational flow" above requires. *See supra* Section I.E; Ex. 1003, col. 3, ll. 56–59 ("What distinguishes this invention from prior art network monitors is that it has the ability to recognize disjointed flows as belonging to the same conversational flow."). Petitioner argues that "Engel's application level lookup_dialog routines search for existing dialogs *based on application and the client-server IP address pair* of the communication," citing Dr. Lin's testimony as support, but Dr. Lin only explains in the cited paragraph how Engel searches for existing dialogs based on the client-server IP address pair, not based on application.[7] *See* Pet. 21 (emphasis added); Ex. 1006 ¶ 86. Indeed, Petitioner acknowledges that Engel's application level dialogs track "all application activity between the client-server pair," without pointing to any specific disclosure in Engel of tracking activity by application. *See* Pet. 21. Nor are we persuaded by Petitioner's argument regarding deallocation. *See id.* at 21–22. Engel deallocates dialog records after a period of inactivity "for a dialog address pair," which does not support Petitioner's contention that Engel links communications by application (as opposed to simply by layer and client-server pair). *See id.* (citing Ex. 1007, col. 17, ll. 2–16); Prelim. Resp. 51.

Second, Petitioner argues that Engel teaches "Application-Specific Server Statistics." Pet. 22–28. According to Petitioner, "Engel recognizes disjointed flows as belonging to the same conversational flow by tracking

[7] We also note that Dr. Lin explains in his declaration how Engel's disclosed system works, but does not opine that the reference discloses "conversational flows" or explain how any of the limitations of the claims are taught by Engel. *See* Ex. 1006 ¶¶ 51–108 (Dr. Lin stating that he "was asked to review Engel and the source code found in Engel's Appendix VI and to provide an overview of the operation of the Engel packet monitor as disclosed in the Engel patent and the Appendix VI source code"); Prelim. Resp. 46. Thus, we determine that Dr. Almeroth's testimony that Engel does *not* disclose "conversational flows" does not create a genuine issue of material fact (which would be viewed in the light most favorable to Petitioner pursuant to 37 C.F.R. § 42.108(c)).

statistics for a given application and a specific Server and all of its clients." *Id.* at 22–23. Petitioner argues that

> Engel discloses tracking application-specific server statistics (i.e., metrics for a given application and a specific Server and all of its clients). . . . Each application-specific server record includes a dialog link queue with a linked list of all application dialogs in which the server has participated (i.e., dialogs for a given application and a specific Server and all of its clients), and therefore recognizes and links disjointed exchanges associated with a particular application, even if the clients were not the same.

Id. at 23–25 (citing Ex. 1007, Fig. 7B (item 160), col. 18, ll. 41–62, col. 17, l. 32–col. 18, l. 40; Ex. 1009, 179, 245–248; Ex. 1006 ¶¶ 62–69).

As explained above, however, Petitioner has only shown that Engel links packets (and tracks corresponding statistics) by layer and client-server pair, not by application. Thus, we are not persuaded that Engel links communications "associated with a particular application" as Petitioner contends. *See id.* We agree with Patent Owner and Dr. Almeroth that

> [b]ecause Engel considers dialogs unique and does not relate subsequent packets with previously seen packets, Engel is user-agnostic—no effort is made to relate packets to specific application activities. Rather, all packet dialogs at a given OSI level are treated uniformly and are indistinct from one another; any context of who sent the packets or for what purpose is lost under Engel. In essence, each dialog is placed into a single bucket and Engel does not recognize or establish relationships among the dialogs.

Prelim. Resp. 52 (citations omitted); *see* Ex. 2001 ¶ 83.

Engel discloses that in a connection-oriented protocol, once a connection is established, Nodes A and B may have simultaneous *unique* dialogs. Ex. 1007, col. 9, ll. 51–57, Fig. 4. Engel further discloses that in a connectionless protocol, once the communication has begun, it is possible to determine the action requested. *Id.* at col. 9, ll. 58–67. However, if there are multiple communication threads between Nodes A and B, they would be *separate and different* dialogs. *Id.* at col. 9, l. 67–col. 10, l. 3. We see no disclosure in Engel of linking these disjointed dialogs into a "conversational flow." Petitioner's cited portions of Engel shed no light on such connections. *See* Pet. 25 (citing Ex. 1007, col. 18, ll. 41–62, col. 17, l. 32–col. 18, l. 40). Keeping statistics relating to separate dialogs does not extend to linking unique dialogs, let alone linking them based on a specific application.

For the reasons set forth above, we are not persuaded by Petitioner's arguments as to how Engel allegedly monitors "conversational flows" using the STATS database and, therefore, are not persuaded that Engel discloses

> (d) a memory for storing a database comprising none or more flow-entries for previously encountered *conversational flows*, each flow-entry identified by identifying information stored in the flow-entry;

Thus, Petitioner has not shown a reasonable likelihood of prevailing on its assertion that independent claim 19, as well as claims 20–22, 25, 26, 29–31, 36–40, and 42, which depend from claim 19, are anticipated by Engel.

B. Obviousness Grounds Based on Engel and Baker, Engel and Graham-Cumming, and Engel and Colloff

Petitioner contends that claims 23 and 24 are unpatentable over Engel and Baker, claims 27, 28, 32, 41, and 43 are unpatentable over Engel and Graham-Cumming, and claims 33–35 are unpatentable over Engel and Colloff, under 35 U.S.C. § 103(a). Pet. 57–73. Petitioner does not contend that any of Baker, Graham-Cumming, or Colloff teaches the "conversational flow" limitations of claim 19, from which these claims depend. *See id.*

Thus, we are not persuaded that Petitioner has established a reasonable likelihood of prevailing on its asserted grounds for the reasons explained above with respect to parent claim 19. *See supra* Section II.A.2.

C. Conclusion

We conclude that Petitioner has not demonstrated a reasonable likelihood of prevailing with respect to at least one claim of the '789 patent challenged in the Petition. Therefore, we do not institute an *inter partes* review on any of the asserted grounds as to any of the challenged claims.

III. ORDER

In consideration of the foregoing, it is hereby:

ORDERED that the Petition is denied as to all challenged claims of the '789 patent.

IPR2017-00629
Patent 6,954,789 B2

PETITIONER:

Eric A. Buresh
Mark C. Lang
Kathleen D. Fitterling
Abran J. Kean
ERISE IP, P.A.
eric.buresh@eriseip.com
mark.lang@eriseip.com
kathleen.fitterling@eriseip.com
abran.kean@eriseip.com

PATENT OWNER:

Steven W. Hartsell
Alexander E. Gasser
SKIERMONT DERBY LLP
shartsell@skiermontderby.com
agasser@skiermontderby.com

Trials@uspto.gov
571-272-7822

Paper 9
Entered: July 26, 2017

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SANDVINE CORPORATION and SANDVINE INCORPORATED ULC,
Petitioner,

v.

PACKET INTELLIGENCE, LLC,
Patent Owner.

Case IPR2017-00630
Patent 6,954,789 B2

Before ELENI MANTIS MERCADER, JUSTIN T. ARBES, and WILLIAM M. FINK, *Administrative Patent Judges*.

FINK, *Administrative Patent Judge*.

DECISION
Denying Institution of *Inter Partes* Review
37 C.F.R. § 42.108

Sandvine Corporation and Sandvine Incorporated ULC (collectively, "Petitioner") filed a Petition (Paper 2, "Pet.") requesting *inter partes* review of claims 1–8, 11–18, and 44–49 of U.S. Patent No. 6,954,789 B2 (Ex. 1004, "the '789 patent") pursuant to 35 U.S.C. § 311(a). Patent Owner Packet Intelligence, LLC filed a Preliminary Response (Paper 6, "Prelim. Resp.") pursuant to 35 U.S.C. § 313. Pursuant to 35 U.S.C. § 314(a), the Director may not authorize an *inter partes* review unless the information in the petition and preliminary response "shows that there is a reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition." For the reasons that follow, we have decided not to institute an *inter partes* review.

I. BACKGROUND

A. The '789 Patent[1]

The '789 patent discloses "[a] monitor for and a method of examining packets passing through a connection point on a computer network." Ex. 1002, Abstract. The '789 patent explains that there was a need in the art for "a real-time network monitor that can provide alarms notifying selected users of problems that may occur with the network or site." *Id.* at col. 2, ll. 3–5. The disclosed monitor receives packets passing in either direction through its connection point on the network and "elucidate[s] what application programs are associated with each packet" by extracting information from the packet, using selected parts of the extracted

[1] Petitioner challenges different claims of the '789 patent in Case IPR2017-00629. Petitioner also challenges patents related to the '789 patent in Cases IPR2017-00450, IPR2017-00451, IPR2017-00769, IPR2017-00862, and IPR2017-00863.

information to identify this packet as part of a flow, "build[ing] a unique flow signature (also called a 'key') for this flow," and "matching this flow in a database of known flows 324." *Id.* at col. 9, ll. 6–9, col 13, ll. 21–28, col. 13, ll. 60–65.

Figure 3 of the '789 patent is reproduced below.

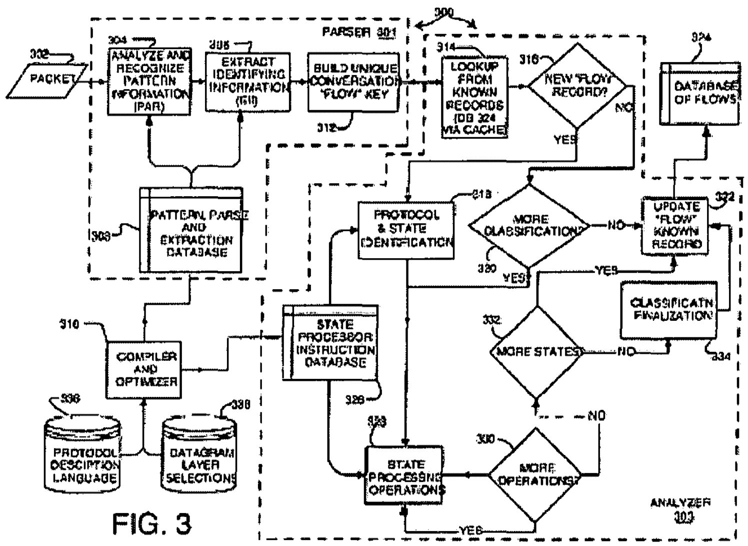


FIG. 3

Figure 3 depicts various components of network packet monitor 300, including parser subsystem 301, analyzer subsystem 303, and database of known flows 324. *Id.* at col. 11, l. 50–col. 13, l. 65. Parser subsystem 300 "parses the packet and determines the protocol types and associated headers for each protocol layer that exists in the packet 302," "extracts characteristic portions (signature information) from the packet 302," and builds the "unique flow signature (also called a 'key') for this flow." *Id.* at col. 12, l. 19–col. 13, l. 28, col. 33, l. 30–col. 34, l. 33 (describing an example of how the disclosed monitor builds signatures and flow states in the context of a Sun Remote Procedure Call (RPC), where, after all of the required processing, "KEY-2 may . . . be used to recognize packets that are in any way associated with the application 'a^{2}'"), Fig. 2.

Analyzer system 303 then determines whether the packet has a matching flow-entry in database of flows 324, and processes the packet accordingly, including, for example, determining whether the packet belongs to an existing conversational flow or a new (i.e., not previously encountered) flow and, in the case of the latter, performing state processing to determine whether the conversational flow has been "fully characterized" and should be finalized. *Id.* at col. 13, l. 60–col. 16, l. 52. The '789 patent discloses that

> [f]uture packets that are part of the same conversational flow have their state analysis continued from a previously achieved state. When enough packets related to an application of interest have been processed, a final recognition state is ultimately reached, i.e., a set of states has been traversed by state analysis to completely characterize the conversational flow. The signature for that final state enables each new incoming packet of the same conversational flow to be individually recognized in real time.
>
> In this manner, one of the great advantages of the present invention is realized. Once a particular set of state transitions has been traversed for the first time and ends in a final state, a short-cut recognition pattern—a signature—[c]an be generated that will key on every new incoming packet that relates to the conversational flow. Checking a signature involves a simple operation, allowing high packet rates to be successfully monitored on the network.

Id. at col. 16, ll. 17–34.

B. Illustrative Claim

Claims 1 and 44 of the '789 patent[2] recite:

1. A method of examining packets passing through a connection point on a computer network, each packets conforming to one or more protocols, the method comprising:

(a) receiving a packet from a packet acquisition device;

(b) performing one or more parsing/extraction operations on the packet to create a parser record comprising a function of selected portions of the packet;

(c) looking up a flow-entry database comprising none or more flow-entries for previously encountered conversational flows, the looking up using at least some of the selected packet portions and determining if the packet is of an existing flow;

(d) if the packet is of an existing flow, classifying the packet as belonging to the found existing flow; and

(e) if the packet is of a new flow, storing a new flow-entry for the new flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry,

wherein the parsing/extraction operations depend on one or more of the protocols to which the packet conforms.

44. A method of examining packets passing through a connection point on a computer network, the method comprising:

(a) receiving a packet from a packet acquisition device;

(b) performing one or more parsing/extraction operations on the packet according to a database of parsing/extraction operations to create a parser record comprising a function of selected portions of the packet, the database of parsing/extraction operations including information on how to determine a set of one or more protocol dependent extraction

[2] Claims 6, 7, 15, 23, 26, 27, and 29 of the '789 patent were corrected in Certificates of Correction dated March 7, 2006, and October 1, 2013.

> operations from data in the packet that indicate a protocol is used in the packet;
>
> (c) looking up a flow-entry database comprising none or more flow-entries for previously encountered conversational flows, the looking up using at least some of the selected packet portions, and determining if the packet is of an existing flow;
>
> (d) if the packet is of an existing flow, obtaining the last encountered state of the flow and performing any state operations specified for the state of the flow starting from the last encountered state of the flow; and
>
> (e) if the packet is of a new flow, performing any analysis required for the initial state of the new flow and storing a new flow-entry for the new flow in the flow-entry database, including identifying information for future packets to be identified with the new flow-entry.

C. The Prior Art

Petitioner relies on the following prior art:

U.S. Patent No. 5,793,954 issued August 11, 1998 (Ex. 1012, "Baker");

U.S. Patent No. 6,115,393, filed July 21, 1995, issued Sept. 5, 2000 (Ex. 1007, "Engel");[3] and

U.S. Patent No. 6,182,146 B1, filed June 27, 1997, issued Jan. 30, 2001 (Ex. 1010, "Graham-Cumming").

D. The Asserted Grounds

Petitioner challenges claims 1–8, 11–18, and 44–49 of the '789 patent on the following grounds:

[3] Engel references Appendices I–VI filed with the originally filed application. *See, e.g.*, Ex. 1007, col. 1, ll. 10–15, col. 5, l. 52–col. 6, l. 3; Ex. 1008 (Appendices I–V); Ex. 1009 (Appendix VI).

Reference(s)	Basis	Claims Challenged
Engel	35 U.S.C. § 102(e)[4]	1–8, 11, 13, and 15
Engel and Baker	35 U.S.C. § 103(a)	12, 44, 45, 47, and 48
Engel and Graham-Cumming	35 U.S.C. § 103(a)	14 and 16–18
Engel, Baker, and Graham-Cumming	35 U.S.C. § 103(a)	46 and 49

E. Claim Interpretation

In an *inter partes* review, claim terms in an unexpired patent are given their "broadest reasonable construction in light of the specification of the patent in which they appear." 37 C.F.R. § 42.100(b). Under the broadest reasonable construction standard, claim terms are given their ordinary and customary meaning, as would be understood by one of ordinary skill in the art in the context of the entire disclosure. *In re Translogic Tech., Inc.*, 504 F.3d 1249, 1257 (Fed. Cir. 2007). Only terms in controversy need to be construed, and only to the extent necessary to resolve the controversy. *Vivid Techs., Inc. v. Am. Sci. & Eng'g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999).

Petitioner provides proposed constructions of the claim terms "conversational flow," "state of the flow," "state operations," and "parser record." Pet. 2–8. Patent Owner provides proposed constructions of the first three terms. Prelim. Resp. 23–30. Other than the term "conversational

[4] The Leahy-Smith America Invents Act, Pub. L. No. 112-29, 125 Stat. 284 (2011) ("AIA"), amended 35 U.S.C. §§ 102 and 103. Because the challenged claims of the '789 patent have an effective filing date before the effective date of the applicable AIA amendments, we refer to the pre-AIA versions of 35 U.S.C. §§ 102 and 103.

flow," discussed below, we determine that no claim term requires express construction to resolve the issues before us.

Claims 1 and 44 both recite (emphasis added):

> (c) looking up a flow-entry database comprising none or more flow-entries for previously encountered *conversational flows*, the looking up using at least some of the selected packet portions and determining if the packet is of an existing flow;

Petitioner argues that "conversational flow" should be interpreted to mean "'the sequence of packets that are exchanged in any direction as a result of an activity,' where a 'conversational flow' is distinguished from a 'connection flow' in that some of the conversational flows 'involve more than one connection' or 'involve more than one exchange of packets between a client and server.'" Pet. 3. Petitioner acknowledges that the claims "require more than identifying and classifying 'only connection flows,'" citing statements in the '789 patent:

> Some prior art packet monitors classify packets into connection flows. The term "connection flow" is commonly used to describe all the packets involved with a single connection. A conversational flow, on the other hand, is the sequence of packets that are exchanged in any direction as a result of an activity—for instance, the running of an application on a server as requested by a client. It is desirable to be able to identify and classify conversational flows rather than only connection flows. The reason for this is that some conversational flows involve more than one connection, and some even involve more than one exchange of packets between a client and server.

Id. at 3–4 (quoting Ex. 1004, col. 2, ll. 42–53) (emphases omitted).

Relying on the same disclosure from the patent, Patent Owner argues that the term "conversational flow" is expressly defined as

> the sequence of packets that are exchanged in any direction as a result of an activity—for instance, the running of an application

> on a server as requested by a client—and where some conversational flows involve more than one connection, and some even involve more than one exchange of packets between a client and server.

Prelim. Resp. 24. Patent Owner further notes that Petitioner agreed to the above interpretation in the related district court case, and the district court adopted it. *Id.* at 24–25 (citing Ex. 2005, 6).

The parties' proposed interpretations are nearly identical. We agree with Patent Owner that the term "conversational flow" is expressly defined in the excerpt of the patent quoted above. Accordingly, for purposes of this Decision, we interpret "conversational flow" to mean the sequence of packets that are exchanged in any direction as a result of an activity (for instance, the running of an application on a server as requested by a client), where some conversational flows involve more than one connection, and some even involve more than one exchange of packets between a client and server.

II. DISCUSSION

A. Anticipation Ground Based on Engel

Petitioner contends that claims 1–8, 11, 13, and 15 are anticipated by Engel under 35 U.S.C. § 102(e), citing the testimony of Bill Lin, Ph.D., as support. Pet. 11–48 (citing Ex. 1006). We are not persuaded that Petitioner has established a reasonable likelihood of prevailing on its asserted ground for the reasons explained below.

1. Engel

Engel is directed to monitoring communications "in a network of nodes, each communication being effected by a transmission of one or more packets among two or more communicating nodes, and each communication complying with a predefined communication protocol." Ex. 1007, Abstract. "The contents of packets are detected passively and in real time, [and] communication information associated with multiple protocols is derived from the packet contents." *Id.* Such communication information "is associated with multiple layers of at least one of the protocols." *Id.* at col. 2, ll. 27–30. A network monitor "collects packets on the network and performs some degree of analysis to search for actual or potential problems and to maintain statistical information for use in later analysis." *Id.* at col. 6, ll. 52–65, Fig. 1. A Real Time Parser (RTP) performs layer-by-layer protocol parsing with appropriate routines. *Id.* at col. 11, ll. 25–30, col. 19, ll. 53–67, col. 21, ll. 15–30. "As each layer is parsed, the RTP invokes the appropriate functions in the statistics module (STATS) to update those statistical objects which must be changed." *Id.* at col. 11, ll. 35–37.

Figure 4 of Engel is reproduced below.

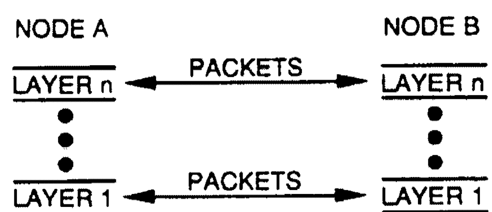


FIG 4

Figure 4 "illustrates the different layers of a communication between two nodes." *Id.* at col. 4, ll. 45–46. In Figure 4, "Nodes A and B are exchanging packets and are engaged in multiple dialogs." *Id.* at col. 9, ll. 31–33. "[A] dialog is a communication [at any layer] between a sender and a receiver, which is composed of one or more packets being transmitted between the two" and often "involve[s] exchanges in both directions." *Id.* at col. 9, ll. 19–24. As shown in Figure 4, "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." *Id.* at col. 9, ll. 37–41. Engel discloses keeping a dialog record that contains, *inter alia*, "the addresses of both ends of the dialog concatenated together to form a single address." *Id.* at col. 18, l. 63–col. 19, l. 4.

2. Analysis

Petitioner argues that Engel discloses all of the limitations of claim 1. Pet. 11–37. We focus on one limitation in particular, which is dispositive of Petitioner's asserted ground. Claim 1 recites:

> (c) looking up a flow-entry database comprising none or more flow-entries for previously encountered *conversational flows*, the looking up using at least some of the selected packet portions and determining if the packet is of an existing flow;[5]

---

[5] We note that although claims 1 and 44 recite "a flow-entry database comprising *none or more flow-entries* for previously encountered conversational flows" (emphasis added), the claims require "conversational flows" at least due to the language of limitations (d) and (e). Limitation (d) covers the situation where there is an "existing" conversational flow and limitation (e) covers the situation of storing a new flow-entry in the flow-entry database for a "new" conversational flow.

For this limitation, Petitioner argues that Engel's STATS database is a flow-entry database that stores "statistics relating to stations and dialogs encountered by the network monitor." *Id.* at 14. Petitioner contends that, through its packet processing and use of the STATS database, Engel monitors "conversational flows" in two different ways: via "Application Level Dialogs" and "Application-Specific Server Statistics." *Id.* at 11–12, 14–29.

At the outset, we note that Petitioner's arguments are premised on its position that related U.S. Patent No. 6,839,751 B1 (Ex. 1002, "the '751 patent") "provides conversational flow examples where flows are determined and metrics reported 'for a given application *and either* a specific Client-Server Pair *or* a specific Server and all of its clients.'"[6] *Id.* at 14–15 (quoting Ex. 1002, col. 34, ll. 58–60). As Patent Owner points out, however, the cited portion of the '751 patent relates to tracking various statistics and does not mention conversational flows. *See* Prelim. Resp. 47–48; *see* Ex. 1002, col. 31, l. 52–col. 49, l. 65 (describing "how the monitor of the invention can be used to monitor the Quality of Service (QOS) by providing QOS Metrics"). The specific language quoted by Petitioner pertains to one such metric, "CSTraffic," which "contains information about the volume of traffic measured for a given application and either a specific Client-Server Pair or a specific Server and all of its clients." Ex. 1002, col. 34, ll. 55–60. However, we do not see how this language

[6] The '789 patent states that the '751 patent is incorporated by reference. Ex. 1004, col. 1, ll. 28–33. Also, both patents claim the benefit of U.S. Provisional Patent Application No. 60/141,903 and have some of the same disclosure in their written descriptions.

supports Petitioner's position. Monitoring statistics is not the same as monitoring a conversational flow, which necessarily identifies a "sequence of packets," under our interpretation. *See supra* Section I.E. In other words, tracking statistics for a given application between a particular client and server, or between a particular server and all of its clients, is not sufficient in and of itself; a conversational flow identifies a sequence of packets exchanged between two nodes as a result of specific activity (e.g., a "sequence of packets that are exchanged in any direction as a result of an activity").

We now turn to Petitioner's two arguments as to how Engel allegedly discloses conversational flows. First, Petitioner contends that Engel teaches "Application Level Dialogs." Pet. 15–22. Petitioner argues that Engel "considers the nature of the communication (e.g., the application program being invoked) in addition to the Client-Server Pair involved in the communication in identifying an application level dialog, rendering such dialog a conversational flow within the meaning of the '789 Patent." *Id.* at 15–16 (quoting Ex. 1007, col. 9, l. 27–col. 10, l. 3). Petitioner relies on Engel's definition of "dialog" as "the exchange of messages and the associated meaning and state that is inherent in any particular exchange at any layer." *Id.* at 16 (citing Ex. 1007, col. 9, ll. 24–27). According to Petitioner, Engel discloses "creating and maintaining dialogs" and collecting associated statistics at different layers. *Id.* In particular, Petitioner asserts that the dialog at the application level "is an exchange of one or more packets in any direction between two nodes as a result of an activity (e.g., [a Network File System (NFS)] mount request)." *Id.* (citing Ex. 1007, col. 9, l. 58–col. 10, l. 3). Petitioner concludes that tracking an application-level dialog constitutes tracking a conversational flow. *Id.* at 16–21 (citing Ex. 1007, Fig. 7C, showing dialog records that point to dialog statistics data structures in the STATS database, and corresponding portions of the source code in Ex. 1009).

Petitioner further relies on Dr. Lin's testimony to assert that, to the extent that conversational flows require the tracking of multiple connections, "Engel's application level lookup_dialog routines search for existing dialogs based on application and the client-server IP address pair of the communication." *Id.* at 21 (citing Ex. 1006 ¶ 86). Petitioner also contends, based on Dr. Lin's testimony, that "Engel's deallocation routines deallocate dialog records when the period of inactivity for a dialog address pair (measured from the last time an application-specific packet (e.g., NFS) between the IP address pair of the dialog was seen) has exceeded a defined application-specific threshold." *Id.* (citing Ex. 1007, col. 17, ll. 2–16; Ex. 1006 ¶¶ 106–107).

Patent Owner responds that Engel's dialogs are different from conversational flows because they are merely "collections of statistics for each [Open Systems Interconnection (OSI)] layer across packets (regardless of application origin)" and do not relate packets that are the result of an application activity. Prelim. Resp. 46–47 (citing Ex. 1007, col. 4, ll. 23–32; Ex. 2001 ¶¶ 78–79). For example, noting Petitioner's statement that "[e]ach application level dialog tracks all application activity between the client-server pair occurring on any connections until the dialog data structures are deallocated for reuse," Patent Owner contends that relating all application activity between a client and server is insufficient because a conversational flow requires relating flows for specific application activities.

Id. at 50–51 (quoting Pet. 21; citing Ex. 2001 ¶¶ 80–81). We agree with Patent Owner.

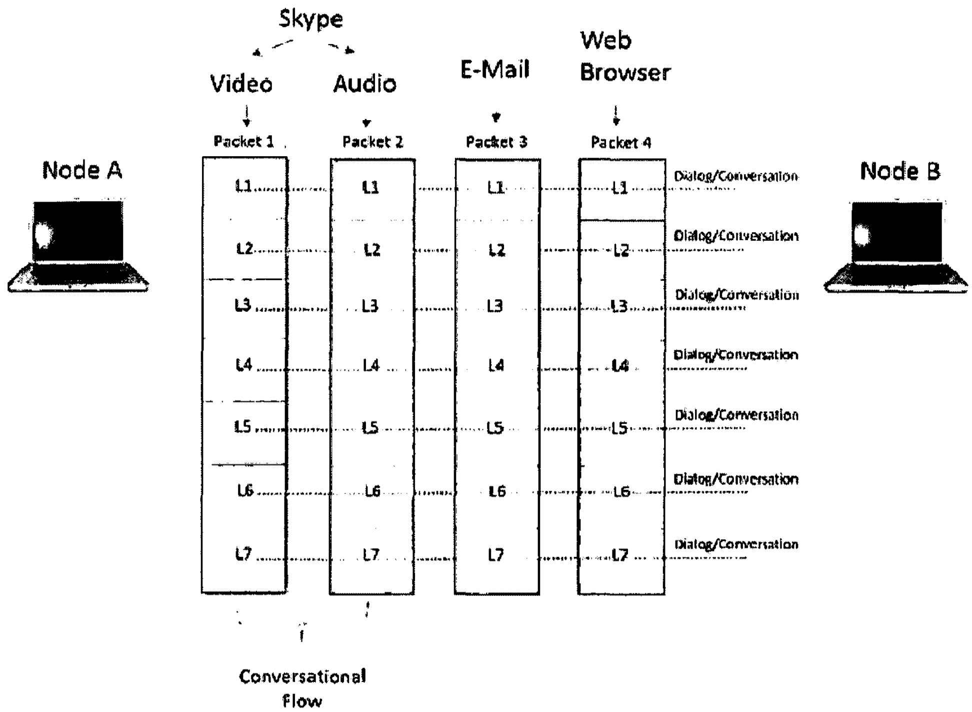
Engel explains that a "dialog" is

> a 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 communication is a packet. A packet conveys meaning 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.
>
> 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.

Ex. 1007, col. 9, ll. 9–43 (emphases added).

As shown in Figure 4 (reproduced above in Section II.A.1), Nodes A and B engage in a dialog at each layer (e.g., the application layer). Thus, Engel monitors communications exchanged between the nodes at a particular layer, and tracks statistics for all communications at that layer (to assist in diagnosing network problems and improving network performance). *See, e.g.*, *id.* at col. 4, ll. 24–32 (Engel "organizes and presents network performance statistics in terms of dialogs which are occurring at any desired level of the communication"). Petitioner, however, does not point to any disclosure in Engel indicating that its disclosed system relates information from one packet to another as pertaining to a specific application activity. In other words, in Engel there is exchange of packets in the application layer constituting a dialog (*see, e.g.*, *id.* at col. 9, ll. 37–43) and collection of respective statistics for that dialog (*see, e.g.*, *id.* at col. 4, ll. 24–32). However, even if application activity causes the dialog to be created, Petitioner does not direct us to teachings or disclosure of identifying a sequence of packets as being part of a specific application activity. Moreover, Patent Owner notes that in Engel "[k]eeping statistics for protocol layers may be temporarily suspended when parsing and statistics gathering is not rapid enough to match the rate of packets to be parsed," which further suggests Engel does not relate sequences of packets to applications because some packets may be lost. Prelim. Resp. 37 (quoting Ex. 1007, col. 3, ll. 12–14; Ex. 2001 ¶ 74).

Patent Owner provides the following illustration on page 42 of its Preliminary Response:



The figure above depicts four packets exchanged between Nodes A and B, with each row representing a layer of the OSI model. Engel treats each layer separately. For example, packets 1 and 2 may both result from the same application (e.g., video and audio traffic using Skype), but Engel would not link them as being part of a single conversational flow. *See id.* at 42–43, 51 ("Engel is not concerned with which application a specific packet originated from or which packets relate to one another based on the application that generated them."). Patent Owner's explanation is consistent with the disclosure of Engel and is supported by the testimony of its declarant, Kevin C. Almeroth, Ph.D. *See, e.g.*, Ex. 1007, col. 9, ll. 9–43 ("a dialog is a communication between a sender and a receiver, which is composed of one or more packets being transmitted between the two," where, for example, "Layer n of Node A is having a dialog with Layer n of node B"); col. 9,

l. 67–col. 10, l. 3 ("If in fact there exists more than one communication thread between Nodes A and B, then these would represent separate, different dialogs."); Ex. 2001 ¶¶ 65–85.

Petitioner's position appears to be that conversational flows exist in Engel simply because communications exist at the application layer and because those communications result from some application activity. *See* Pet. 15–22. As explained in connection with Patent Owner's illustration above, however, we do not see—and Petitioner does not point to—anything in Engel indicating that it links communications by application (as opposed to by layer and client-server pair) as our interpretation of "conversational flow" above requires. *See supra* Section I.E; Ex. 1003, col. 3, ll. 56–59 ("What distinguishes this invention from prior art network monitors is that it has the ability to recognize disjointed flows as belonging to the same conversational flow."). Petitioner argues that "Engel's application level lookup_dialog routines search for existing dialogs *based on application and the client-server IP address pair* of the communication," citing Dr. Lin's testimony as support, but Dr. Lin only explains in the cited paragraph how Engel searches for existing dialogs based on the client-server IP address pair, not based on application.[7] *See* Pet. 21 (emphasis added); Ex. 1006

[7] We also note that Dr. Lin explains in his declaration how Engel's disclosed system works, but does not opine that the reference discloses "conversational flows" or explain how any of the limitations of the claims are taught by Engel. *See* Ex. 1006 ¶¶ 51–108 (Dr. Lin stating that he "was asked to review Engel and the source code found in Engel's Appendix VI and to provide an overview of the operation of the Engel packet monitor as disclosed in the Engel patent and the Appendix VI source code"); Prelim. Resp. 46. Thus, we determine that Dr. Almeroth's testimony that Engel does *not* disclose "conversational flows" does not create a genuine issue of

¶ 86. Indeed, Petitioner acknowledges that Engel's application level dialogs track "all application activity between the client-server pair," without pointing to any specific disclosure in Engel of tracking activity by application. *See* Pet. 21. Nor are we persuaded by Petitioner's argument regarding deallocation. *See id.* Engel deallocates dialog records after a period of inactivity "for a dialog address pair," which does not support Petitioner's contention that Engel links communications by application (as opposed to simply by layer and client-server pair). *See id.* (citing Ex. 1007, col. 17, ll. 2–16); Prelim. Resp. 51.

Second, Petitioner argues that Engel teaches "Application-Specific Server Statistics." Pet. 22–30. According to Petitioner, "Engel recognizes disjointed flows as belonging to the same conversational flow by tracking statistics for a given application and a specific Server and all of its clients." *Id.* at 23. Petitioner argues that

> Engel discloses tracking application-specific server statistics (i.e., metrics for a given application and a specific Server and all of its clients). . . . Each application-specific server record includes a dialog link queue with a linked list of all application dialogs in which the server has participated (i.e., dialogs for a given application and a specific Server and all of its clients), and therefore recognizes and links disjointed exchanges associated with a particular application, even if the clients were not the same.

Id. at 24–26 (citing Ex. 1007, Fig. 7B (item 160), col. 18, ll. 41–62, col. 17, l. 32–col. 18, l. 40; Ex. 1009, 179, 245–248; Ex. 1006 ¶¶ 62–69).

As explained above, however, Petitioner has only shown that Engel links packets (and tracks corresponding statistics) by layer and client-server

---

material fact (which would be viewed in the light most favorable to Petitioner pursuant to 37 C.F.R. § 42.108(c)).

pair, not by application. Thus, we are not persuaded that Engel links communications "associated with a particular application" as Petitioner contends. *See id.* We agree with Patent Owner and Dr. Almeroth that

> [b]ecause Engel considers dialogs unique and does not relate subsequent packets with previously seen packets, Engel is user-agnostic—no effort is made to relate packets to specific application activities. Rather, all packet dialogs at a given OSI level are treated uniformly and are indistinct from one another; any context of who sent the packets or for what purpose is lost under Engel. In essence, each dialog is placed into a single bucket and Engel does not recognize or establish relationships among the dialogs.

Prelim. Resp. 52 (citations omitted); *see* Ex. 2001 ¶ 83.

Engel discloses that in a connection-oriented protocol, once a connection is established, Nodes A and B may have simultaneous *unique* dialogs. Ex. 1007, col. 9, ll. 51–57, Fig. 4. Engel further discloses that in a connectionless protocol, once the communication has begun, it is possible to determine the action requested. *Id.* at col. 9, ll. 58–67. However, if there are multiple communication threads between Nodes A and B, they would be *separate and different* dialogs. *Id.* at col. 9, l. 67–col. 10, l. 3. We see no disclosure in Engel of linking these disjointed dialogs into a "conversational flow." Petitioner's cited portions of Engel shed no light on such connections. *See* Pet. 24–26 (citing Ex. 1007, col. 18, ll. 41–62, col. 17, l. 32–col. 18, l. 40). Keeping statistics relating to separate dialogs does not extend to linking unique dialogs, let alone linking them based on a specific application.

For the reasons set forth above, we are not persuaded by Petitioner's arguments as to how Engel allegedly monitors "conversational flows" using the STATS database and, therefore, are not persuaded that Engel discloses

> (c) looking up a flow-entry database comprising none or more flow-entries for previously encountered *conversational flows*, the looking up using at least some of the selected packet portions and determining if the packet is of an existing flow;

Thus, Petitioner has not shown a reasonable likelihood of prevailing on its assertion that independent claim 1, as well as claims 2–8, 11, 13, and 15, which depend from claim 1, are anticipated by Engel.

B. Obviousness Ground Based on Engel and Baker

Petitioner contends that independent claim 44 and dependent claims 12, 45, 47, and 48 are unpatentable over Engel and Baker, under 35 U.S.C. § 103(a). Pet. 57–73. Claim 12 depends from claim 1. Claim 44 recites an identical "conversational flow" limitation to that discussed above with respect to claim 1:

> (c) looking up a flow-entry database comprising none or more flow-entries for previously encountered *conversational flows*, the looking up using at least some of the selected packet portions and determining if the packet is of an existing flow.

Petitioner relies on the same analysis for "Element 44(c)" as discussed above in Section II.A.2. Pet. 52 ("*See* Element 1(c)."). Accordingly, for the reasons set forth above, we are not persuaded that Petitioner has established a reasonable likelihood of prevailing on its asserted ground for the reasons explained above with respect to claim 44, as well as claims 12, 45, 47, and 48, which depend from claims 1 or 44. *See supra* Section II.A.2.

C. Obviousness Grounds Based on Engel and Graham-Cumming, and Engel, Baker, and Graham-Cumming

Petitioner contends that claims 14 and 16–18 are unpatentable over Engel and Graham-Cumming, and claims 46 and 49 are unpatentable over

Engel, Baker, and Graham-Cumming, under 35 U.S.C. § 103(a). Pet. 53–62. Petitioner does not contend that either of Baker or Graham-Cumming teaches the "conversational flow" limitations of claims 1 and 44. *See id.* Thus, we are not persuaded that Petitioner has established a reasonable likelihood of prevailing on its asserted grounds for the reasons explained above with respect to parent claims 1 and 44. *See supra* Section II.A.2.

D. Conclusion

We conclude that Petitioner has not demonstrated a reasonable likelihood of prevailing with respect to at least one claim of the '789 patent challenged in the Petition. Therefore, we do not institute an *inter partes* review on any of the asserted grounds as to any of the challenged claims.

III. ORDER

In consideration of the foregoing, it is hereby:

ORDERED that the Petition is denied as to all challenged claims of the '789 patent.

PETITIONER:

Eric A. Buresh
Mark C. Lang
Kathleen D. Fitterling
Abran J. Kean
ERISE IP, P.A.
eric.buresh@eriseip.com
mark.lang@eriseip.com
kathleen.fitterling@eriseip.com
abran.kean@eriseip.com

PATENT OWNER:

Steven W. Hartsell
Alexander E. Gasser
SKIERMONT DERBY LLP
shartsell@skiermontderby.com
agasser@skiermontderby.com

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/684,776	10/14/2003	Russell S. Dietz		3352

96039 7590 12/11/2017

Meunier Carlin & Curfman LLC
999 Peachtree Street NE
Suite 1300
Atlanta, GA 30309

EXAMINER	
MEKY, MOUSTAFA M	

ART UNIT	PAPER NUMBER
2157	

NOTIFICATION DATE	DELIVERY MODE
12/11/2017	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

docketing@mcciplaw.com

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

In re Patent No. 6,954,789 :
Issue Date: October 11, 2005 :
Application No. 10/684,776 : NOTICE
Filed: October 14, 2003 :
Title of Invention: METHOD AND :
APPARATUS FOR MONITORING TRAFFIC
IN A NETWORK

This is a notice regarding your request for acceptance of a fee deficiency submission under 37 CFR 1.28(c) filed July 13, 2017.

On September 1, 1998, the Court of Appeals for the Federal Circuit held that 37 CFR 1.28(c) is the sole provision governing the time for correction of the erroneous payment of the issue fee as a small entity. **See DH Technology v. Synergystex International, Inc. 154 F.3d 1333, 47 USPQ2d 1865 (Fed. Cir. Sept. 1, 1998).**

The Office no longer investigates or rejects original or reissue applications under 37 CFR 1.56. **1098 Off. Gaz. Pat. Office 502 (January 3, 1989)**. Therefore, nothing in this Notice is intended to imply that an investigation was done.

Your fee deficiency submission under 37 CFR 1.28 is hereby **ACCEPTED**.

This patent is no longer entitled to small entity status. Accordingly, all future fees paid in this patent must be paid at the undiscounted entity rate.

Applicant is reminded that pursuant to 37 CFR 1.28(c)(2)(i) "where a fee paid in error as a small entity was subject to a fee decrease between the time the fee was paid in error and the time the deficiency is paid in full, the deficiency owed is equal to the amount (previously) paid in error." If the itemization submitted with the present request does not comply with 37 CFR 1.28(c)(2)(i) because a fee has decreased, applicant should, within ONE MONTH of the date of this Notice, submit a renewed request for acceptance of a fee deficiency under 37 CFR 1.28(c) along with balance of the correct fee deficiency in compliance with 37 CFR 1.28(c)(2)(i) and a replacement itemized listing of each fee erroneously paid as a small entity in compliance with 37 CFR 1.28(c)(2)(ii). **NO EXTENSION OF TIME UNDER 37 CFR 1.136 IS PERMITTED.**

Inquiries related to this communication should be directed to Joy Dobbs at (571) 272-3001.

/Liana Walsh/
Liana Walsh
Petitions Lead Paralegal Specialist
Office of Petitions

Office of Petitions: Decision Count Sheet

Mailing Month: 12

Application No. 10684776

For US serial numbers: enter number only, no slashes or commas. Ex: 10123456
For PCT: enter "51+single digit of year of filing+last 5 numbers", Ex. for PCT/US05/12345, enter 51512345

Deciding Official: Joy Dobbs

Count (1) - Palm Credit 10684776

Decision: GRANT

FINANCE WORK NEEDED
☑ Select Check Box for YES

* G R A N T *

Decision Type: 321 - 37 CFR 1.28 TO MAKE ENTITY STATUS LARGE FR

* 3 2 1 *

Notes:

Count (2)

Decision: n/a

FINANCE WORK NEEDED
☐ Select Check Box for YES

Decision Type: NONE

Notes:

Count (3)

Decision: n/a

FINANCE WORK NEEDED
☐ Select Check Box for YES

Decision Type: NONE

Notes:

Initials of Approving Official (if required)

If more than 3 decisions, attach 2nd count sheet & mark this box ☐

Printed on: 12/5/2017

Office of Petitions Internal Document - Ver. 5.0

Office of Petitions: Routing Sheet

4700

Application No. 10684776

This application is being forwarded to your office for further processing. A decision has been rendered on a petition filed in this application, as indicated below. For details of this decision, please see the document PET.OP.DEC filed on the same date as this document.

- [X] **GRANTED**
- [ ] **DISMISSED**
- [ ] **DENIED**

AO 120 (Rev. 08/10)

TO: **Mail Stop 8 Director of the U.S. Patent and Trademark Office P.O. Box 1450 Alexandria, VA 22313-1450**	**REPORT ON THE FILING OR DETERMINATION OF AN ACTION REGARDING A PATENT OR TRADEMARK**

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court for the District of Delaware on the following

☐ Trademarks or ☑ Patents. (☐ the patent action involves 35 U.S.C. § 292.):

DOCKET NO.	DATE FILED 11/10/2017	U.S. DISTRICT COURT for the District of Delaware
PLAINTIFF NetScout Systems, Inc.		DEFENDANT Packet Intelligence LLC, Packet Intelligence Holdings LLC, and Longhorn Asset Group LLC

PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1 6,651,099	11/18/2003	Defendants
2 6,665,725	12/16/2003	Defendants
3 6,771,646	8/3/2004	Defendants
4 6,839,751	1/4/2005	Defendants
5 6,954,789	10/11/2005	Defendants

In the above—entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY ☐ Amendment ☐ Answer ☐ Cross Bill ☐ Other Pleading	
PATENT OR TRADEMARK NO.	**DATE OF PATENT OR TRADEMARK**	**HOLDER OF PATENT OR TRADEMARK**
1		
2		
3		
4		
5		

In the above—entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT

CLERK	(BY) DEPUTY CLERK	DATE

Copy 1—Upon initiation of action, mail this copy to Director Copy 3—Upon termination of action, mail this copy to Director
Copy 2—Upon filing document adding patent(s), mail this copy to Director Copy 4—Case file copy

AO 120 (Rev. 08/10)

TO: **Mail Stop 8 Director of the U.S. Patent and Trademark Office P.O. Box 1450 Alexandria, VA 22313-1450**	**REPORT ON THE FILING OR DETERMINATION OF AN ACTION REGARDING A PATENT OR TRADEMARK**

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court Eastern District of Texas on the following

☐ Trademarks or ☑ Patents. (☐ the patent action involves 35 U.S.C. § 292.):

DOCKET NO.	DATE FILED	U.S. DISTRICT COURT
2:16-cv-00147	2/17/2016	Eastern District of Texas

PLAINTIFF	DEFENDANT
Packet Intelligence LLC	Sandvine Corporation Sandvine Incorporated ULC

PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1 6,651,099	11/18/2003	Packet Intelligence LLC
2 6,665,725	12/16/2003	Packet Intelligence LLC
3 6,771,646	8/3/2004	Packet Intelligence LLC
4 6,839,751	1/4/2005	Packet Intelligence LLC
5 6,954,789	10/11/2005	Packet Intelligence LLC

In the above—entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY ☐ Amendment ☐ Answer ☐ Cross Bill ☐ Other Pleading	
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1		
2		
3		
4		
5		

In the above—entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT

CLERK	(BY) DEPUTY CLERK	DATE

Copy 1—Upon initiation of action, mail this copy to Director Copy 3—Upon termination of action, mail this copy to Director
Copy 2—Upon filing document adding patent(s), mail this copy to Director Copy 4—Case file copy