US 20020013840A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2002/0013840 A1

Border et al. (43) Pub. Date: Jan. 31, 2002

(54) NETWORK MANAGEMENT OF A PERFORMANCE ENHANCING PROXY ARCHITECTURE

(76) Inventors: **John Border**, Poolesville, MD (US); **Deepak Arur**, McLean, VA (US)

Correspondence Address:
Hughes Electronics Corporation
Patent Docket Administration
Bldg. 1, Mail Stop A109
P.O. Box 956
El Segundo, CA 90245-0956 (US)

(21) Appl. No.: **09/905,151**

(22) Filed: **Jul. 13, 2001**

Related U.S. Application Data

(63) Non-provisional of provisional application No. 60/220,026, filed on Jul. 21, 2000. Non-provisional of provisional application No. 60/225,630, filed on Aug. 15, 2000.

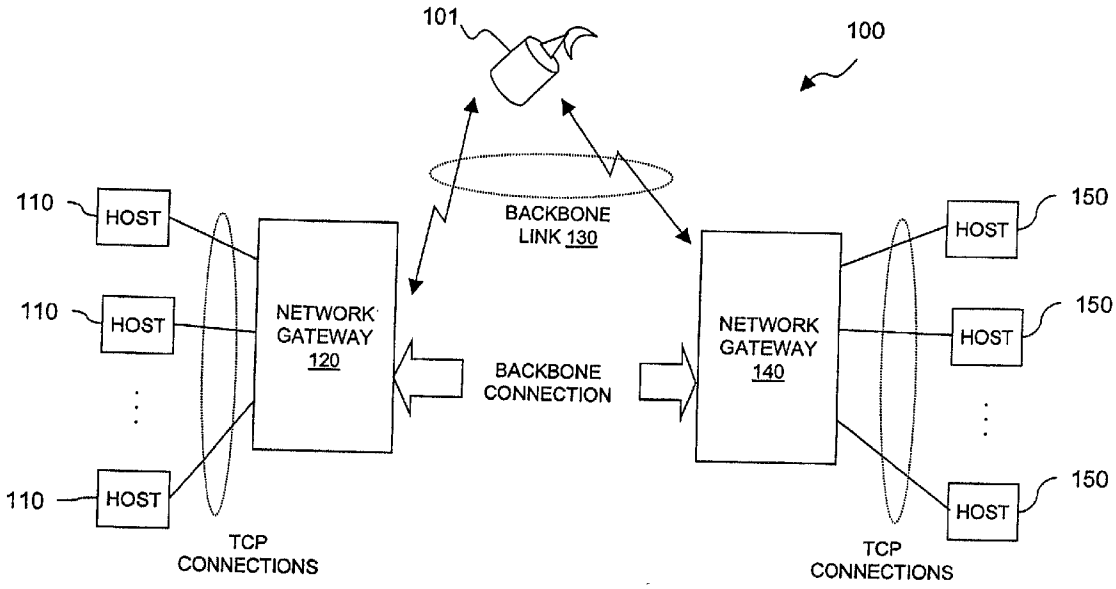
Publication Classification

(51) **Int. Cl.**[7] **G06F 15/173**; G06F 15/177

(52) **U.S. Cl.** **709/224**; 709/221

(57) **ABSTRACT**

A communication system having a proxy architecture is disclosed. The system includes a platform that provides performance enhancing functions. The platform includes a management protocol agent that gathers information associated with the platform. The system also includes a network management system that communicates with the platform. The network management system receives information from the management protocol agent. The information provides at least one of event data and alarm data, wherein the platform has a profile that specifies configuration parameters. The network management system is configured to selectively modify the profile in response to the received information and to forward the modified profile to the platform. The above arrangement has particular applicability to a bandwidth constrained communication system, such as a satellite network.



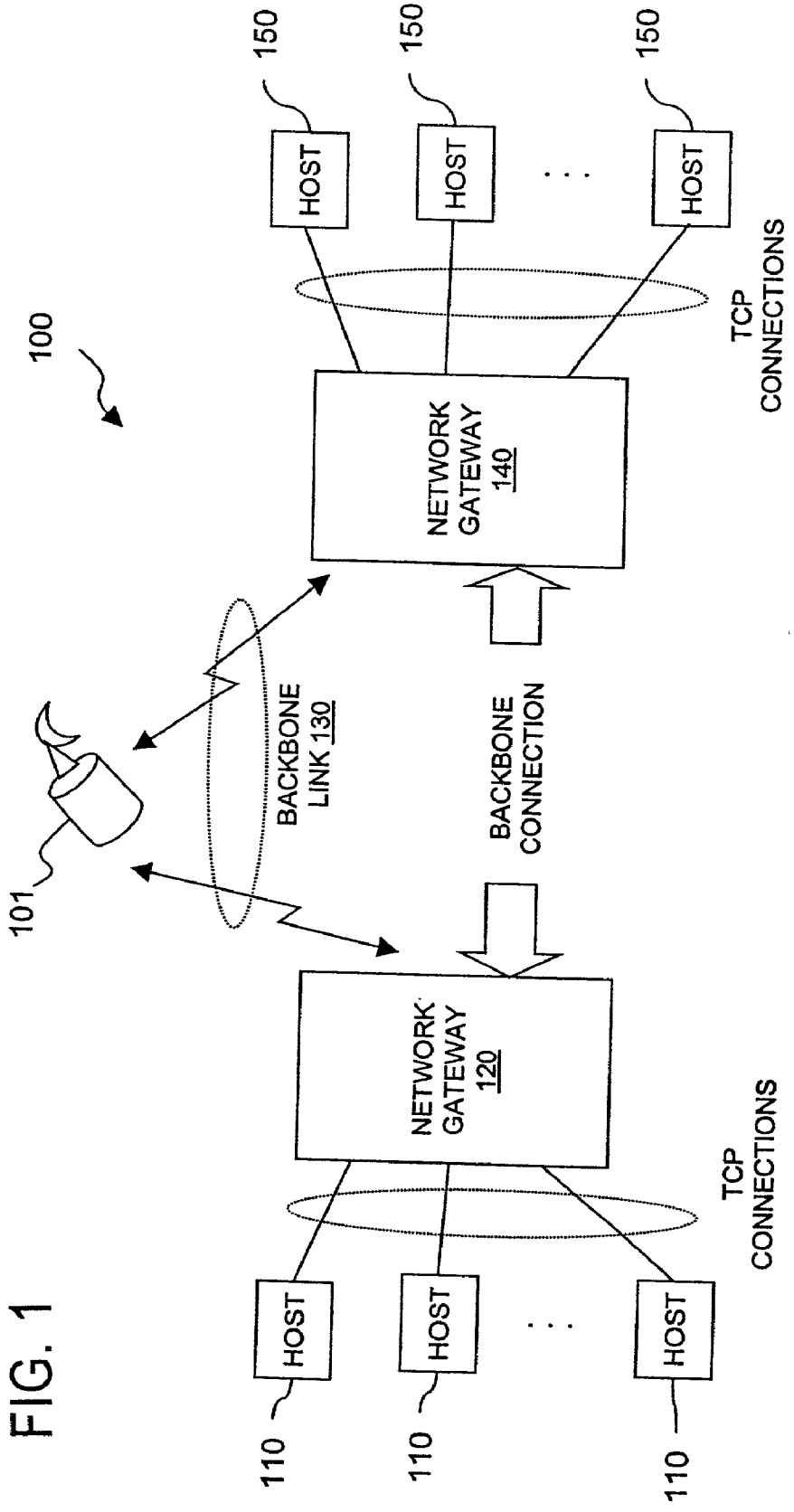

100
150
150
150
HOST
HOST
HOST
TCP CONNECTIONS
NETWORK GATEWAY 140
BACKBONE LINK 130
BACKBONE CONNECTION
101
NETWORK GATEWAY 120
TCP CONNECTIONS
HOST
HOST
HOST
110
110
110

FIG. 1

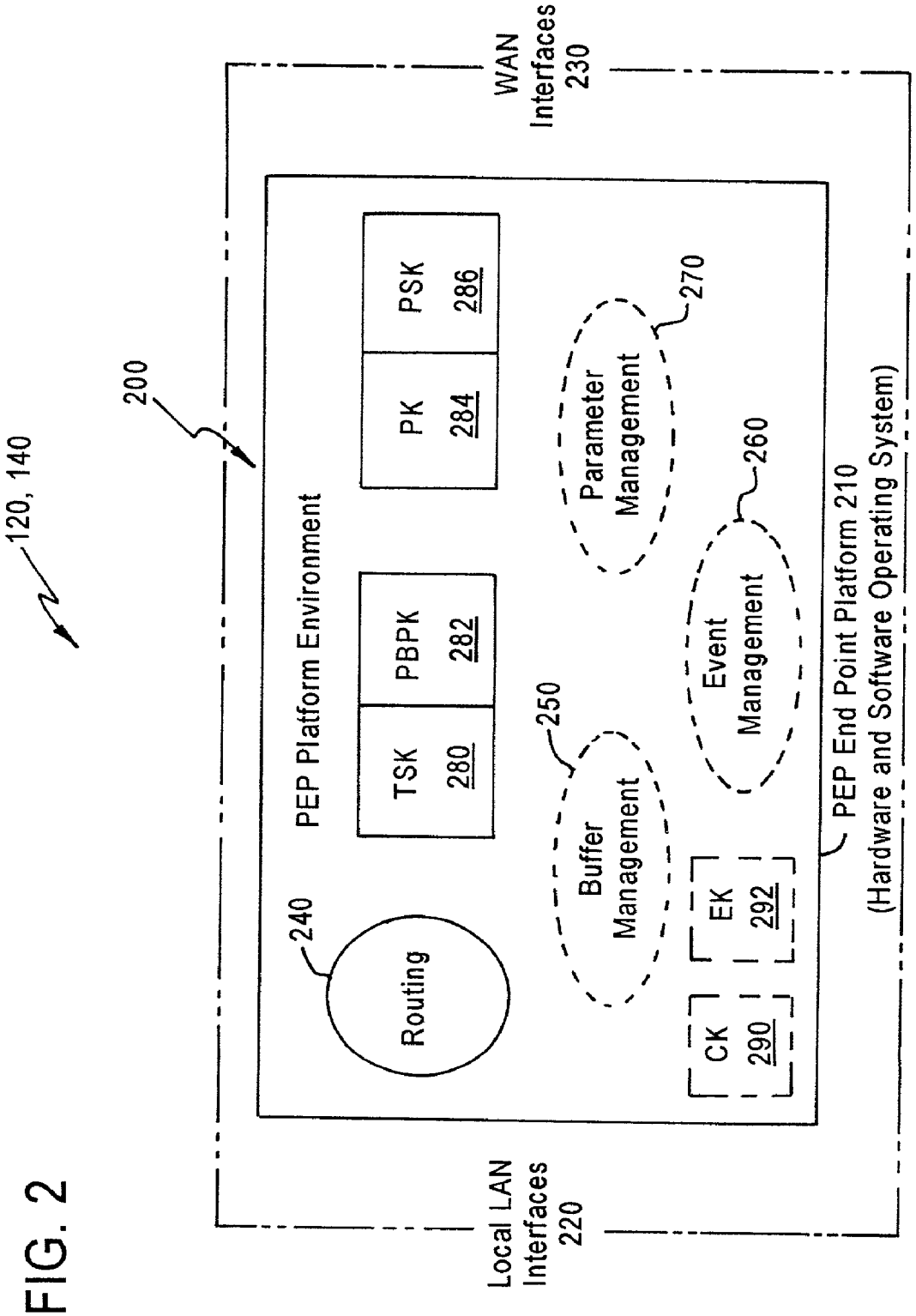

120, 140
200
WAN Interfaces 230
PEP Platform Environment
PSK 286
PK 284
TSK 280
PBPK 282
Parameter Management
270
260
Event Management
250
Buffer Management
240
Routing
EK 292
CK 290
PEP End Point Platform 210
(Hardware and Software Operating System)
Local LAN Interfaces 220

FIG. 2

FIG. 3

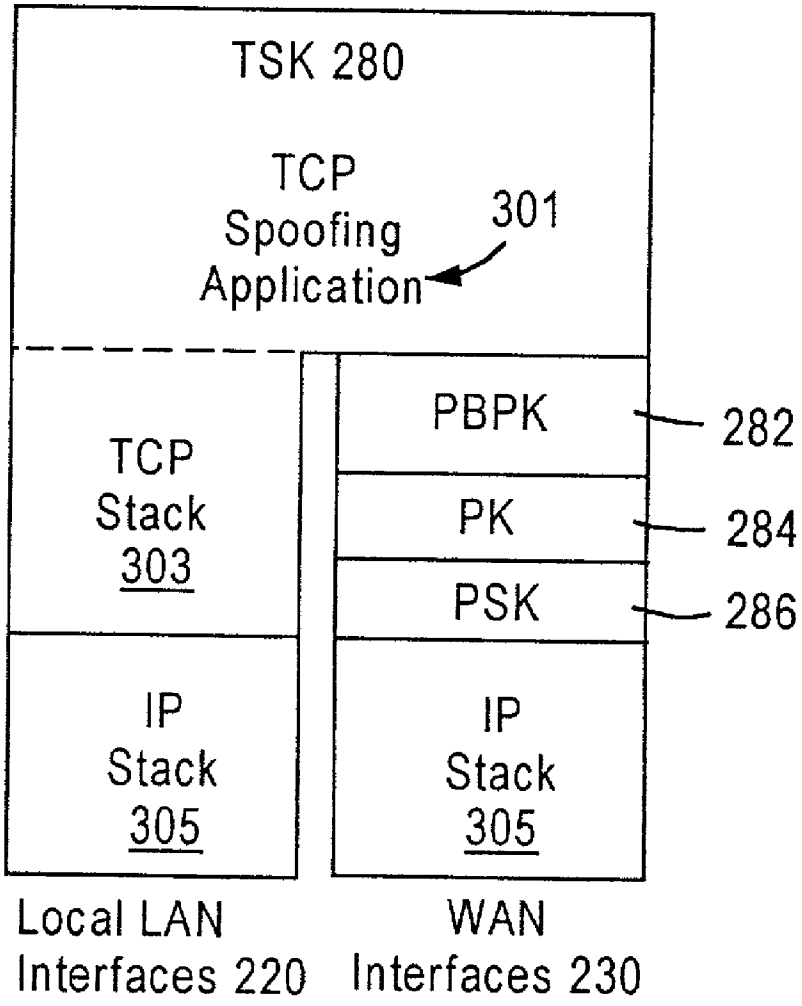

TSK 280
TCP Spoofing Application
301
TCP Stack 303
PBPK
282
PK
284
PSK
286
IP Stack 305
IP Stack 305
Local LAN Interfaces 220
WAN Interfaces 230

FIG. 4A

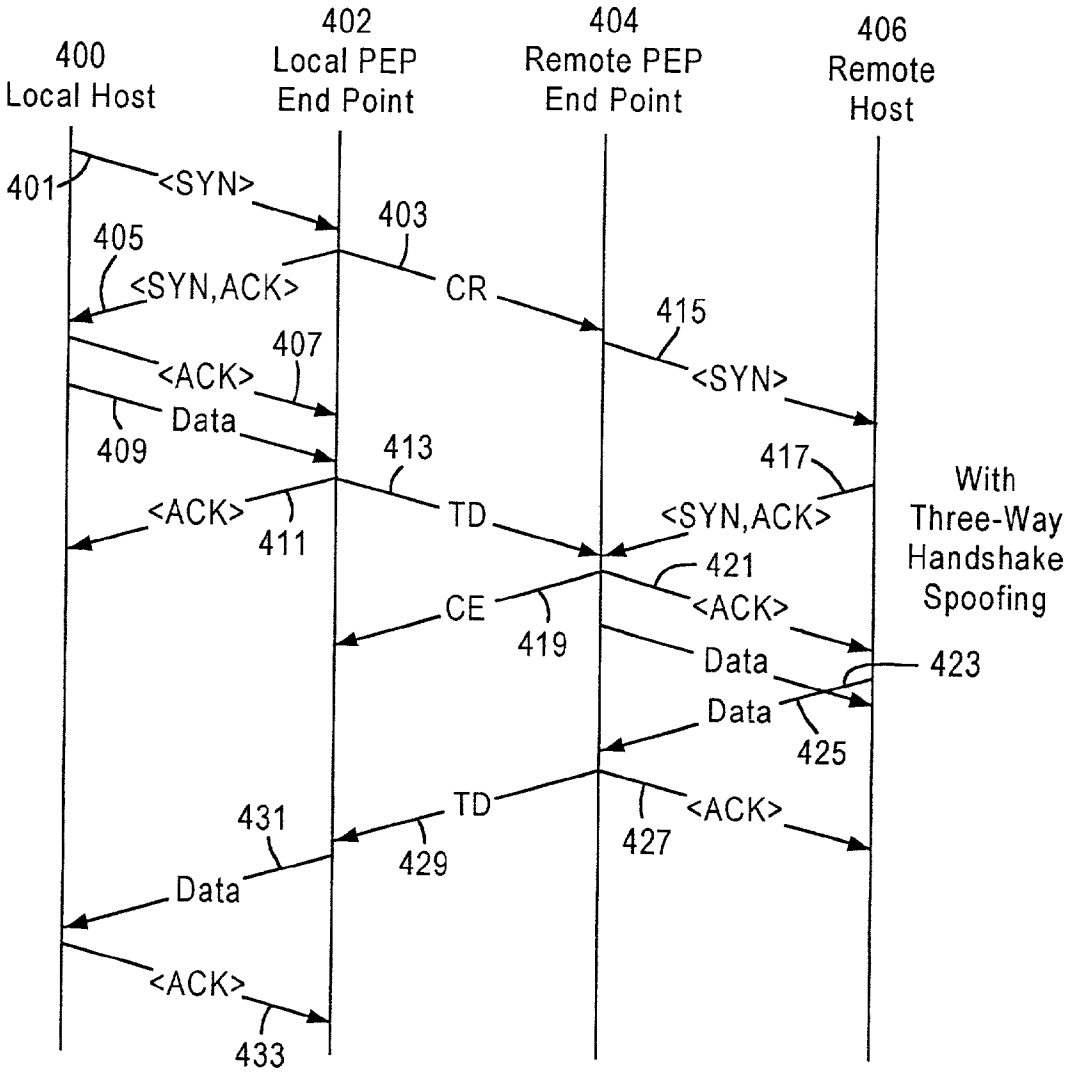

400
Local Host
402
Local PEP
End Point
404
Remote PEP
End Point
406
Remote
Host
401
<SYN>
403
405
<SYN,ACK>
CR
415
407
<ACK>
<SYN>
Data
409
413
417
With
Three-Way
Handshake
Spoofing
<ACK>
411
TD
<SYN,ACK>
421
CE
419
<ACK>
Data
423
Data
425
TD
431
429
427
<ACK>
Data
<ACK>
433

FIG. 4B

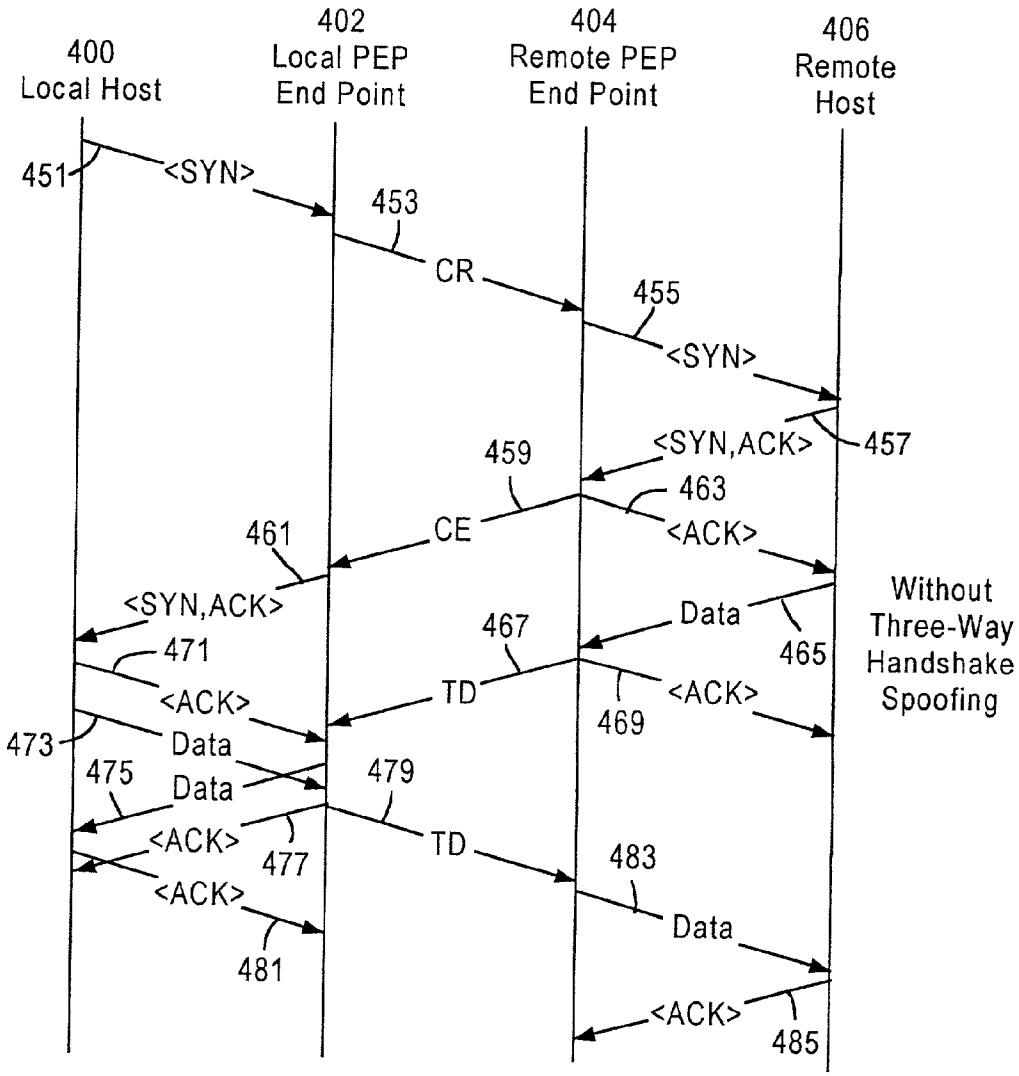
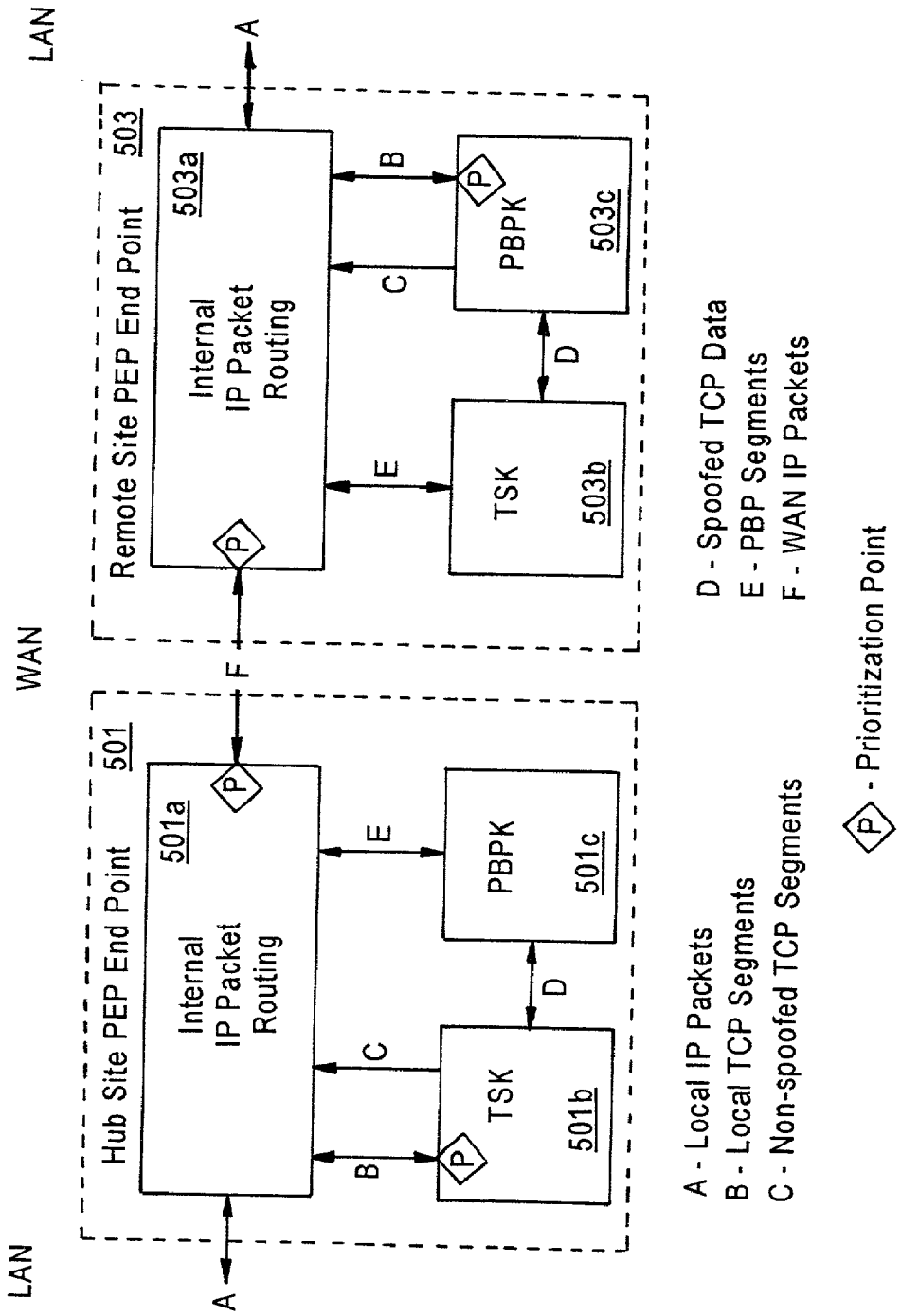

400
Local Host
402
Local PEP
End Point
404
Remote PEP
End Point
406
Remote
Host
451
<SYN>
453
CR
455
<SYN>
457
<SYN,ACK>
459
463
<ACK>
CE
461
<SYN,ACK>
Data
467
465
Without
Three-Way
Handshake
Spoofing
471
<ACK>
TD
<ACK>
469
473
Data
475
Data
479
<ACK>
477
TD
483
<ACK>
Data
481
<ACK>
485

FIG. 5


LAN
WAN
LAN
Hub Site PEP End Point
501
501a
Internal
IP Packet
Routing
501b
TSK
501c
PBPK
Remote Site PEP End Point
503
503a
Internal
IP Packet
Routing
503b
TSK
503c
PBPK
A
B
C
D
E
F
A - Local IP Packets
B - Local TCP Segments
C - Non-spoofed TCP Segments
D - Spoofed TCP Data
E - PBP Segments
F - WAN IP Packets
P - Prioritization Point

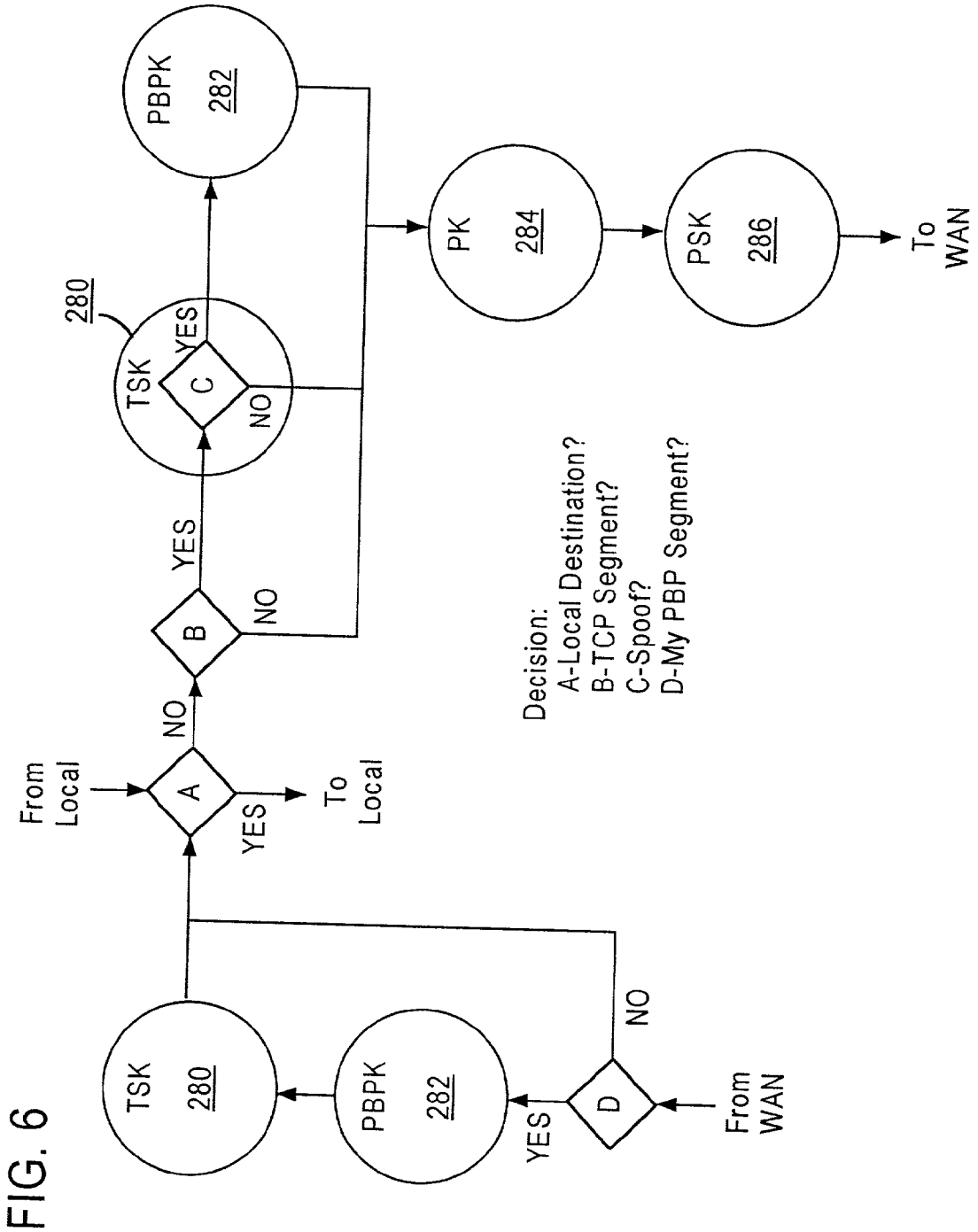

From Local
A
NO
YES
To Local
B
NO
YES
C
NO
YES
280
TSK
PBPK 282
PK 284
PSK 286
To WAN
From WAN
D
NO
YES
TSK 280
PBPK 282
Decision:
A-Local Destination?
B-TCP Segment?
C-Spoof?
D-My PBP Segment?

FIG. 6

FIG. 7

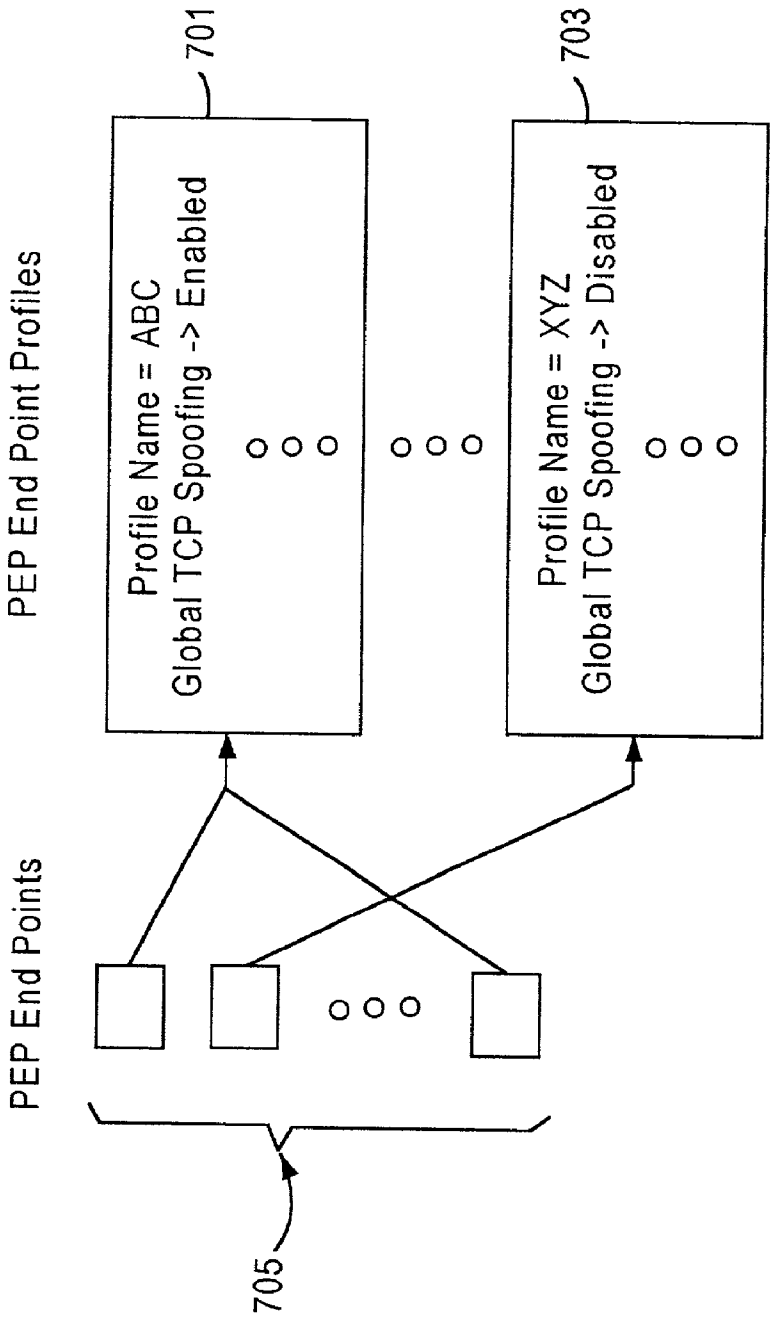

PEP End Point Profiles
701
Profile Name = ABC
Global TCP Spoofing -> Enabled
703
Profile Name = XYZ
Global TCP Spoofing -> Disabled
PEP End Points
705

FIG. 8

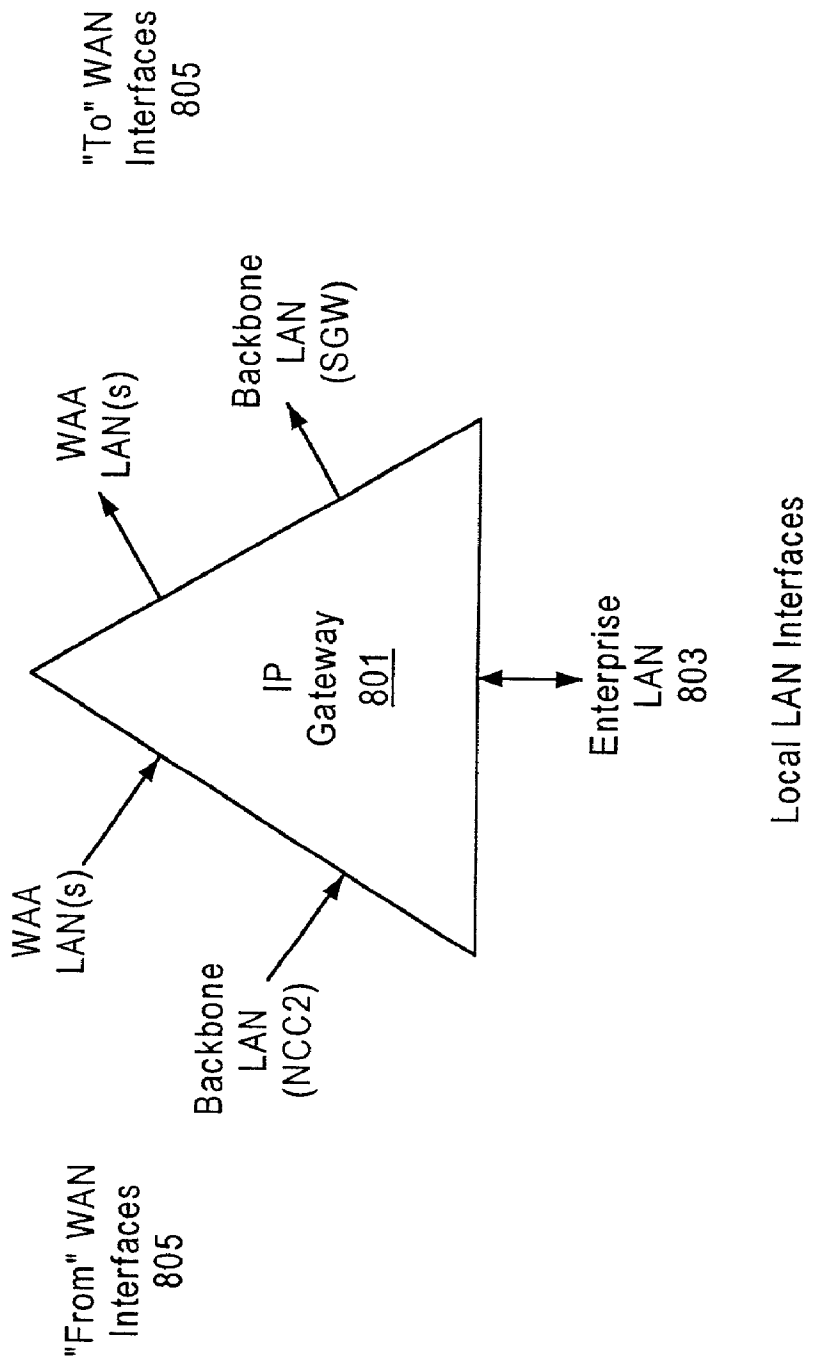

"To" WAN Interfaces 805
WAA LAN(s)
Backbone LAN (SGW)
IP Gateway 801
Enterprise LAN 803
Local LAN Interfaces
WAA LAN(s)
Backbone LAN (NCC2)
"From" WAN Interfaces 805

FIG. 9

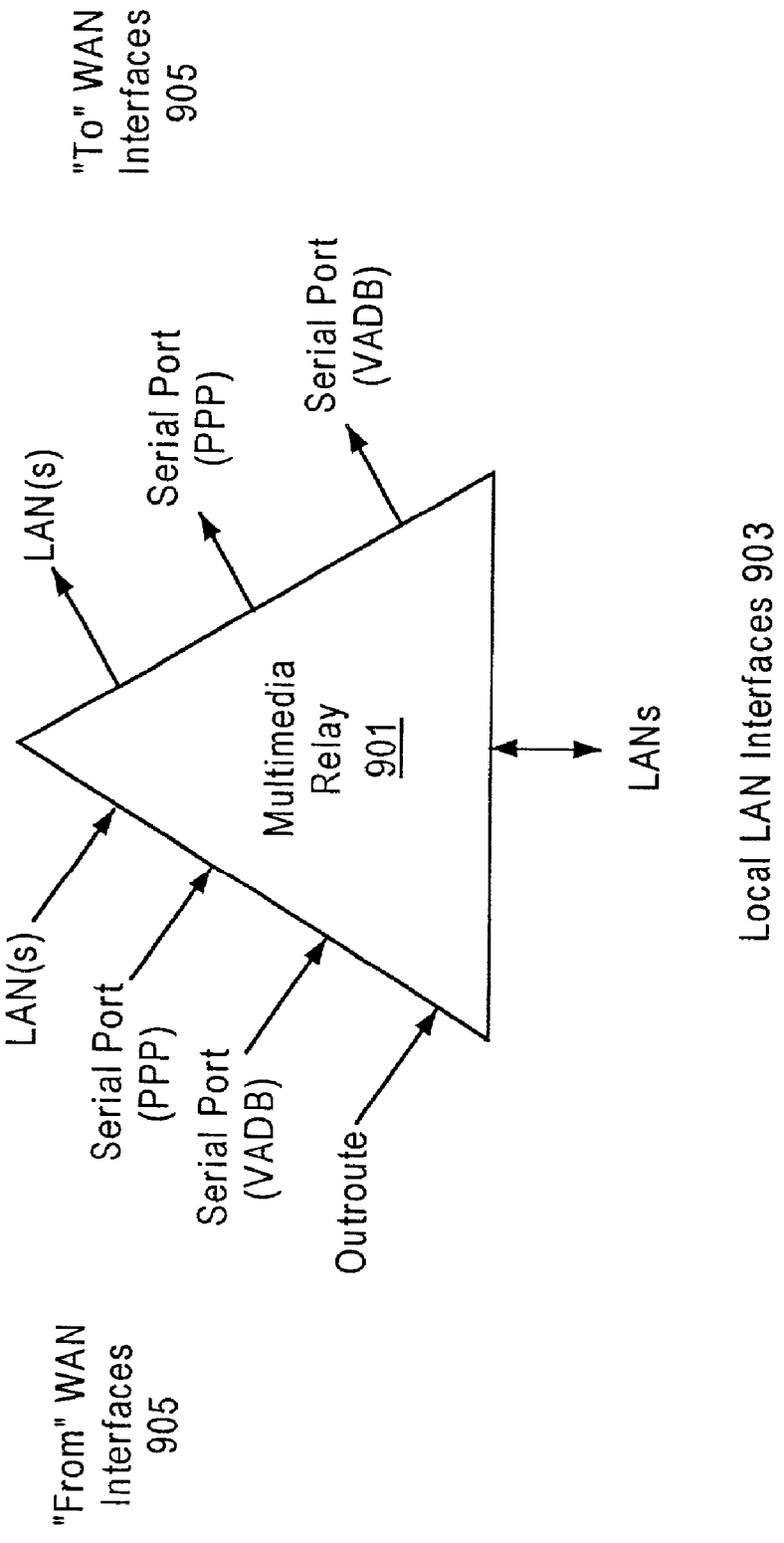

"To" WAN Interfaces 905
LAN(s)
Serial Port (PPP)
Serial Port (VADB)
Multimedia Relay 901
LANs
Local LAN Interfaces 903
LAN(s)
Serial Port (PPP)
Serial Port (VADB)
Outroute
"From" WAN Interfaces 905

FIG. 10

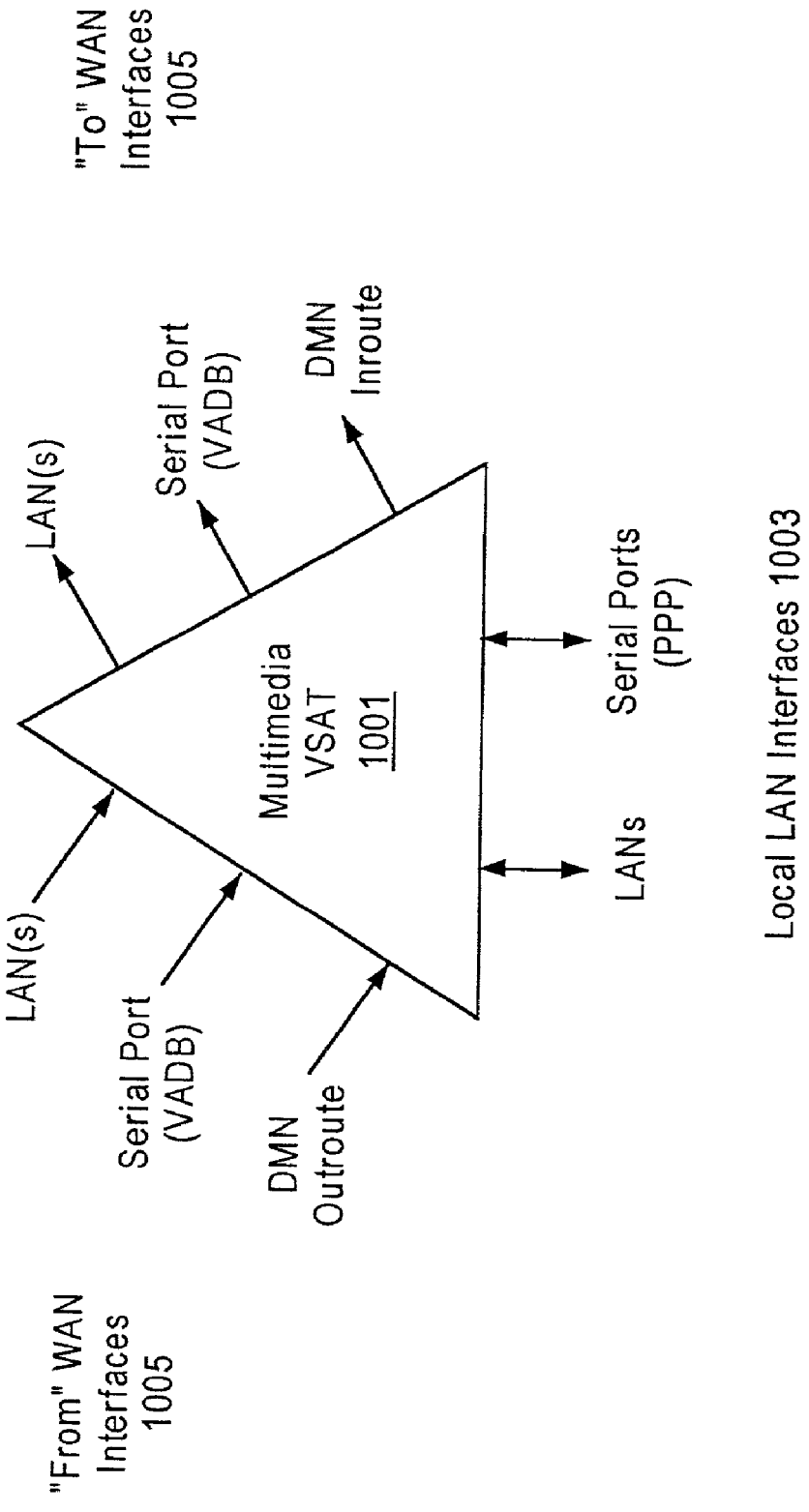

"To" WAN Interfaces 1005
LAN(s)
Serial Port (VADB)
DMN Inroute
Multimedia VSAT 1001
Serial Ports (PPP)
LANs
Local LAN Interfaces 1003
LAN(s)
Serial Port (VADB)
DMN Outroute
"From" WAN Interfaces 1005

FIG. 11

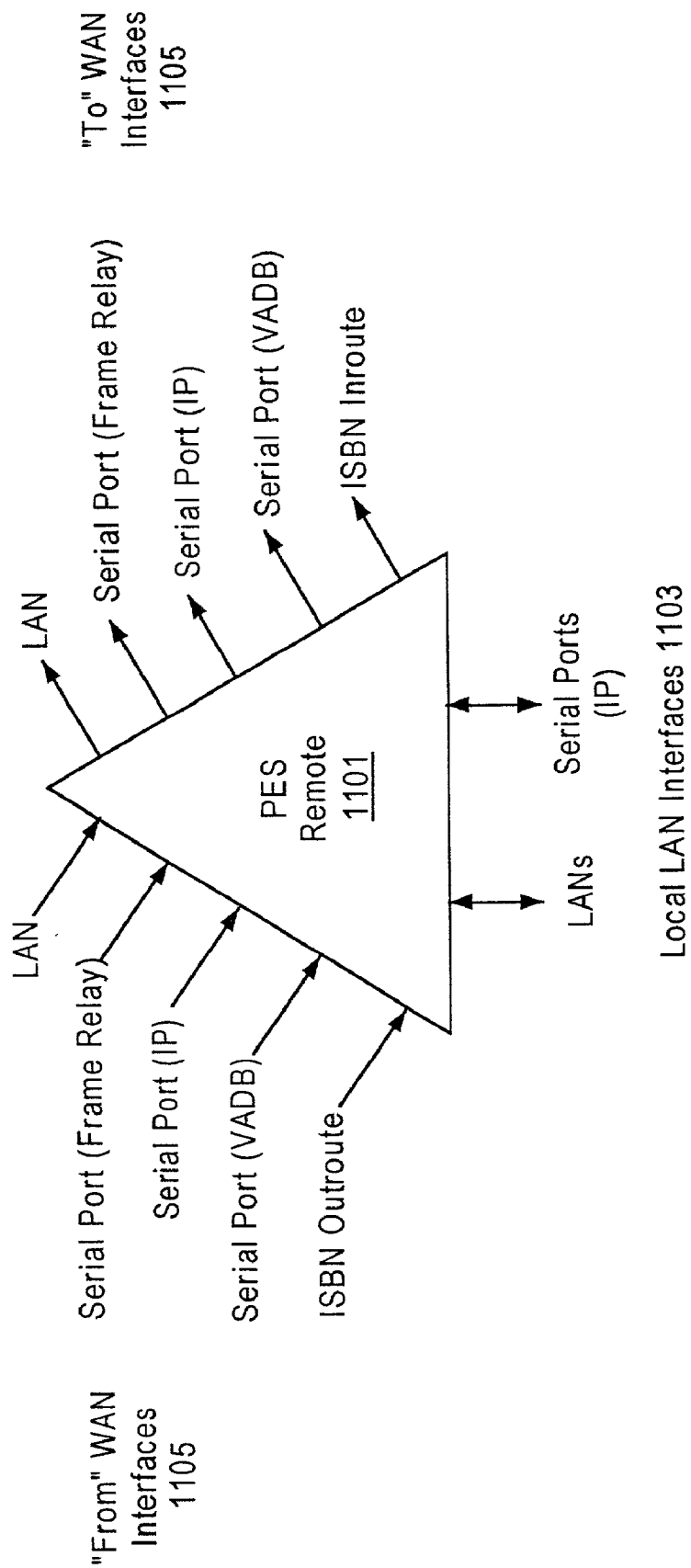

"From" WAN Interfaces 1105
"To" WAN Interfaces 1105
LAN
Serial Port (Frame Relay)
Serial Port (IP)
Serial Port (VADB)
ISBN Outroute
LAN
Serial Port (Frame Relay)
Serial Port (IP)
Serial Port (VADB)
ISBN Inroute
PES Remote 1101
LANs
Serial Ports (IP)
Local LAN Interfaces 1103

FIG. 12

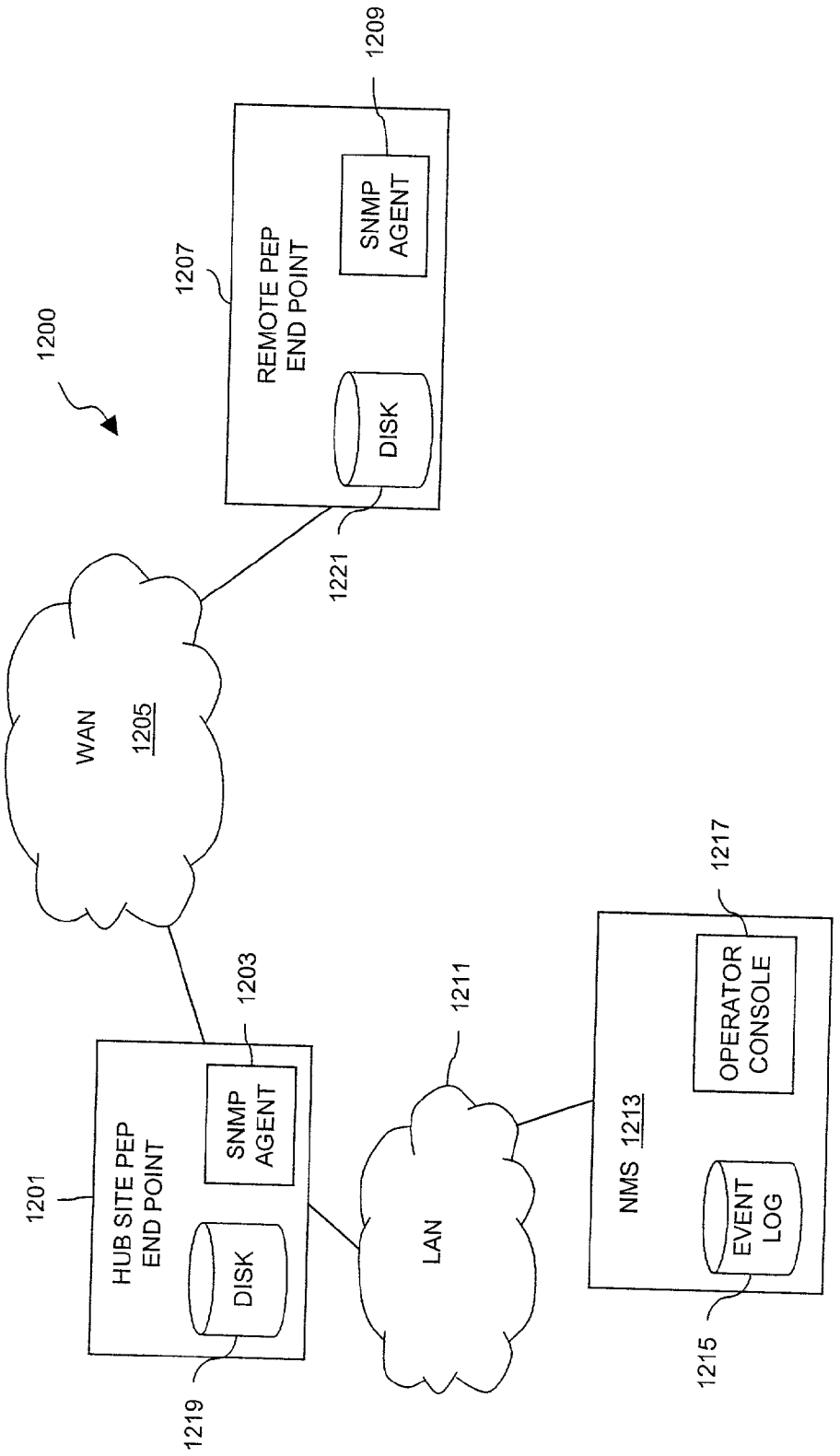
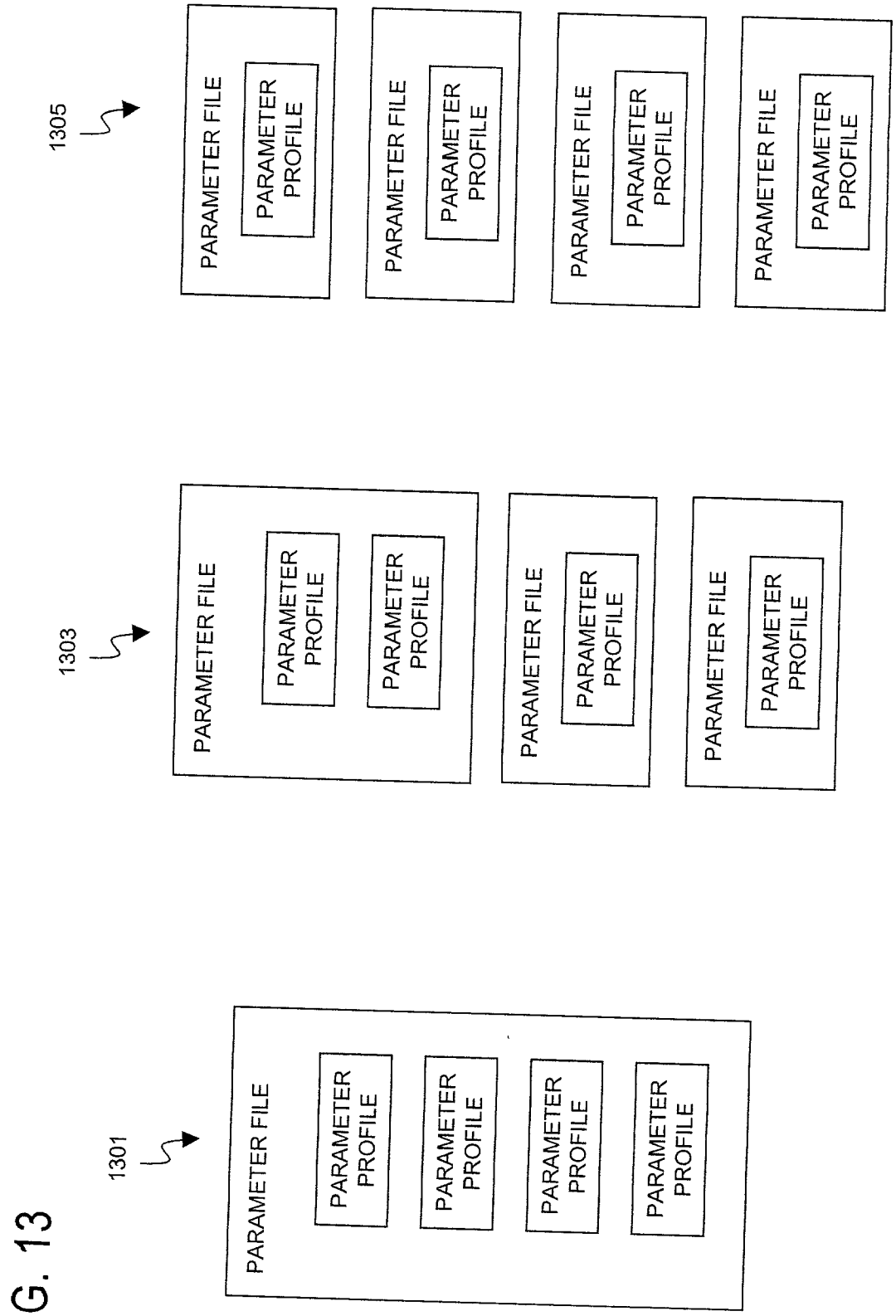

1200
1201
HUB SITE PEP END POINT
SNMP AGENT
1203
DISK
1219
WAN 1205
1207
REMOTE PEP END POINT
SNMP AGENT
1209
DISK
1221
LAN
1211
NMS 1213
OPERATOR CONSOLE
1217
EVENT LOG
1215

FIG. 13


1301
PARAMETER FILE
PARAMETER PROFILE
PARAMETER PROFILE
PARAMETER PROFILE
PARAMETER PROFILE
1303
PARAMETER FILE
PARAMETER PROFILE
PARAMETER PROFILE
PARAMETER FILE
PARAMETER PROFILE
PARAMETER FILE
PARAMETER PROFILE
1305
PARAMETER FILE
PARAMETER PROFILE
PARAMETER FILE
PARAMETER PROFILE
PARAMETER FILE
PARAMETER PROFILE
PARAMETER FILE
PARAMETER PROFILE

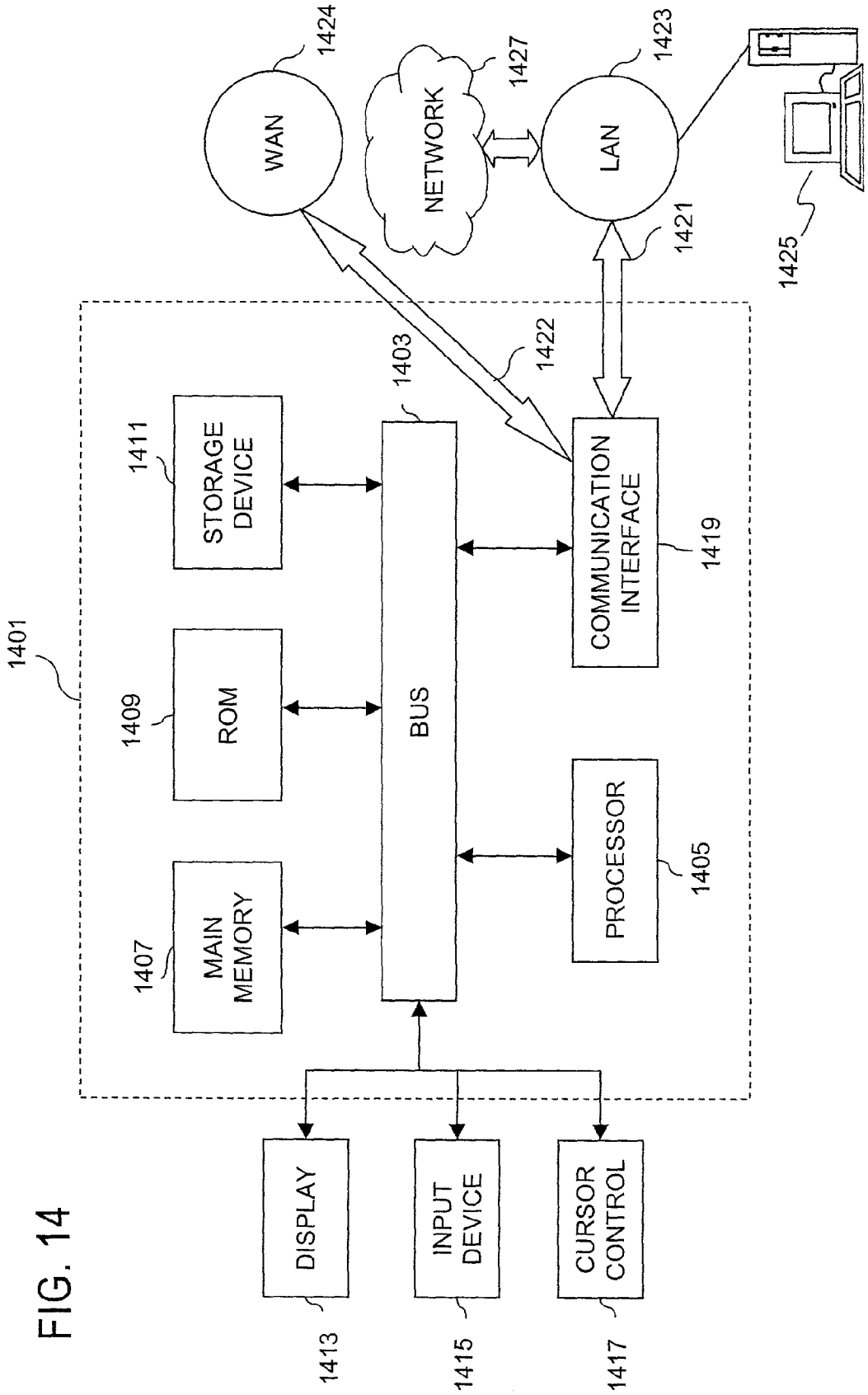

1424
WAN
1427
NETWORK
1423
LAN
1425
1421
1422
1403
1401
1411
STORAGE DEVICE
1409
ROM
1407
MAIN MEMORY
BUS
COMMUNICATION INTERFACE
1419
PROCESSOR
1405
DISPLAY
1413
INPUT DEVICE
1415
CURSOR CONTROL
1417

FIG. 14

FIG. 15

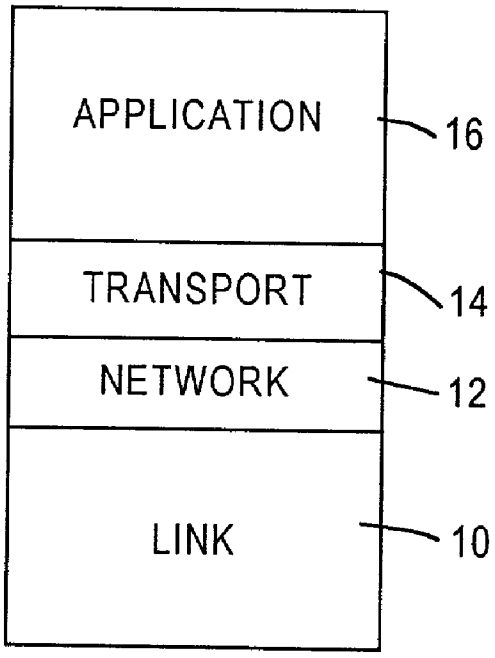
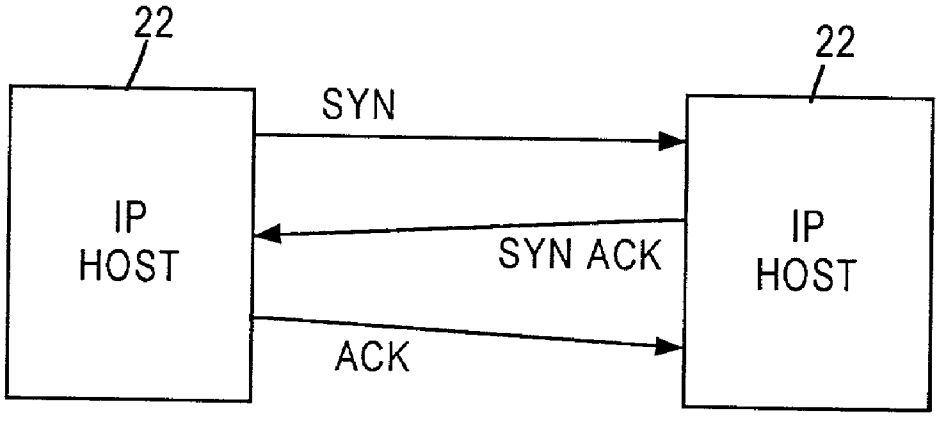


FIG. 16



TCP CONNECTION

NETWORK MANAGEMENT OF A PERFORMANCE ENHANCING PROXY ARCHITECTURE

CROSS-REFERENCES TO RELATED APPLICATION

[0001] This application is related to and claims the benefit of priority to: (i) U.S. Provisional Patent Application (Serial No. 60/220,026), filed Jul. 21, 2000, entitled "Performance Enhancing Proxy," and (ii) U.S. Provisional Patent Application (Serial No. 60/225,630), filed Aug. 15, 2000, entitled "Performance Enhancing Proxy"; all of which are incorporated herein by reference in their entirety.

BACKGROUND OF THE INVENTION

[0002] 1. Field of the Invention

[0003] The present invention relates to a communication system, and is more particularly related to a network management system utilized in a proxy architecture.

[0004] 2. Discussion of the Background

[0005] The entrenchment of data networking into the routines of modern society, as evidenced by the prevalence of the Internet, particularly the World Wide Web, has placed ever-growing demands on service providers to continually improve network performance. To meet this challenge, service providers have invested heavily in upgrading their networks to increase system capacity (i.e., bandwidth). In many circumstances, such upgrades may not be feasible economically or the physical constraints of the communication system does not permit simply "upgrading." Accordingly, service providers have also invested in developing techniques to optimize the performance of their networks. Because much of today's networks are either operating with or are required to interface with the Transmission Control Protocol/Internet Protocol (TCP/IP) suite, attention has been focused on optimizing TCP/IP based networking operations.

[0006] As the networking standard for the global Internet, TCP/IP has earned such acceptance among the industry because of its flexibility and rich heritage in the research community. The transmission control protocol (TCP) is the dominant protocol in use today on the Internet. TCP is carried by the Internet protocol (IP) and is used in a variety of applications including reliable file transfer and Internet web page access applications. The four layers of the TCP/IP protocol suite are illustrated in **FIG. 15**. As illustrated, the link layer (or the network interface layer) **10** includes device drivers in the operating system and any corresponding network interface cards. Together, the device driver and the interface cards handle hardware details of physically interfacing with any cable or whatever type of media that is being used. The network layer (also referred to as the Internet layer) **12** handles the movement of packets around the network. Routing of packets, for example, takes place at the network layer **12**. IP, Internet control message protocol (ICMP), and Internet group management protocol (IGMP) may provide the network layer in the TCP/IP protocol suite. The transport layer **14** provides a flow of data between two hosts, for the application layer **16** above.

[0007] In the TCP/IP protocol suite, there are at least two different transport protocols, TCP and a user datagram protocol (UDP). TCP, which provides a reliable flow of data between two hosts, is primarily concerned with dividing the data passed to it from the application layer **16** into appropriately sized segments for the network layer **12** below, acknowledging received packets, setting timeouts to make certain the other end acknowledges packets that are sent, and so on. Because this reliable flow of data is provided by the transport layer **14**, the application layer **16** is isolated from these details. UDP, on the other hand, provides a much simpler service to the application layer **16**. UDP just sends packets of data called datagrams from one host to another, with no guarantee that the datagrams will reach their destination. Any desired reliability must be added by a higher layer, such as the application layer **16**.

[0008] The application layer **16** handles the details of the particular application. There are many common TCP/IP applications that almost every implementation provides, including telnet for remote log-in, the file transfer protocol (FTP), the simple mail transfer protocol (SMTP) or electronic mail, the simple network management protocol (SNMP), the hypertext transfer protocol (HTTP), and many others.

[0009] As mentioned, TCP provides reliable, in-sequence delivery of data between two IP hosts. The IP hosts set up a TCP connection, using a conventional TCP three-way handshake and then transfer data using a window based protocol with the successfully received data acknowledged.

[0010] To understand where optimizations may be made, it is instructive to consider a typical TCP connection establishment. **FIG. 16** illustrates an example of the conventional TCP three-way handshake between IP hosts **20** and **22**. First, the IP host **20** that wishes to initiate a transfer with IP host **22**, sends a synchronize (SYN) signal to IP host **22**. The IP host **22** acknowledges the SYN signal from IP host **20** by sending a SYN acknowledgement (ACK). The third step of the conventional TCP three-way handshake is the issuance of an ACK signal from the IP host **20** to the other IP host **22**. At this point, IP host **22** is ready to receive the data from IP host **20** (and vice versa). After all the data has been delivered, another handshake (similar to the handshake described to initiate the connection) is used to close the TCP connection.

[0011] TCP was designed to be very flexible and to work over a wide variety of communication links, including both slow and fast links, high latency links, and links with low and high error rates. However, while TCP (and other high layer protocols) works with many different kinds of links, TCP performance, in particular, the throughput possible across the TCP connection, is affected by the characteristics of the link in which it is used. There are many link layer design considerations that should be taken into account when designing a link layer service that is intended to support Internet protocols. However, not all characteristics can be compensated for by choices in the link layer design. TCP has been designed to be very flexible with respect to the links which it traverses. Such flexibility is achieved at the cost of sub-optimal operation in a number of environments vis-á-vis a tailored protocol. The tailored protocol, which is usually proprietary in nature, may be more optimal, but greatly lacks flexibility in terms of networking environments and interoperability.

[0012] An alternative to a tailored protocol is the use of performance enhancing proxies (PEPs), to perform a general

class of functions termed "TCP spoofing," in order to improve TCP performance over impaired (i.e., high latency or high error rate) links. TCP spoofing involves an intermediate network device (the performance enhancing proxy (PEP)) intercepting and altering, through the addition and/or deletion of TCP segments, the behavior of the TCP connection in an attempt to improve its performance.

[0013] Conventional TCP spoofing implementations include the local acknowledgement of TCP data segments in order to get the TCP data sender to send additional data sooner than it would have sent if spoofing were not being performed, thus improving the throughput of the TCP connection. Generally, conventional TCP spoofing implementations have focused simply on increasing the throughput of TCP connections either by using larger windows over the link or by using compression to reduce the amount of data which needs to be sent, or both.

[0014] Many TCP PEP implementations are based on TCP ACK manipulation. These may include TCP ACK spacing where ACKs which are bunched together are spaced apart, local TCP ACKs, local TCP retransmissions, and TCP ACK filtering and reconstruction. Other PEP mechanisms include tunneling, compression, and priority-based multiplexing.

[0015] Based on the foregoing, there is a clear need for improved approaches to optimizing network performance, while achieving network redundancy. There is also a need to enhance network performance, without a costly infrastructure investment. There is also a need to employ a network performance enhancing mechanism that complies with existing standards to facilitate rapid deployment. There is a further need to simplify the receiver design. Therefore, an approach for optimizing network performance using a proxy architecture is highly desirable.

SUMMARY OF THE INVENTION

[0016] The present invention addresses the above stated needs by providing a communication system with performance enhancing functionality. A network management system communicates with performance enhancing proxy (PEP) end point platforms to configure the platforms by utilizing profiles corresponding to the PEP end point platforms.

[0017] According to one aspect of the invention, a method for monitoring a communication system that includes a platform configured to perform a plurality of performance enhancing functions is provided. The method includes receiving information relating to configuration parameters as specified in a profile of the platform. The method also includes selectively modifying the profile in response to the received information, and forwarding the modified profile to the platform. Under this approach, system performance is enhanced.

[0018] According to one aspect of the invention, a communication system includes a platform that is configured to provide performance enhancing functions. The platform has a profile that specifies configuration parameters. The system also includes a network management system that communicates with the platform. The network management system is configured to receive information relating to the configuration parameters as specified in the profile. The network management system is configured to selectively modify the profile in response to the received information and to forward the modified profile to the platform. The above arrangement advantageously provides ease of configuration.

[0019] According to another aspect of the invention, a network apparatus for monitoring a communication system that includes a platform configured to perform a plurality of performance enhancing functions is disclosed. The apparatus includes means for receiving information relating to configuration parameters as specified in a profile of the platform. The apparatus also includes means for selectively modifying the profile in response to the received information, and means for forwarding the modified profile to the platform. The above arrangement advantageously provides an efficient approach to configuring of network elements.

[0020] In yet another aspect of the invention, a computer-readable medium carrying one or more sequences of one or more instructions for monitoring a communication system that includes a platform configured to perform a plurality of performance enhancing functions is disclosed. The one or more sequences of one or more instructions include instructions which, when executed by one or more processors, cause the one or more processors to perform the step of receiving information relating to configuration parameters as specified in a profile of the platform. Other steps include selectively modifying the profile in response to the received information, and forwarding the modified profile to the platform. This approach advantageously simplifies network management.

BRIEF DESCRIPTION OF THE DRAWINGS

[0021] A more complete appreciation of the invention and many of the attendant advantages thereof will be readily obtained as the same becomes better understood by reference to the following detailed description when considered in connection with the accompanying drawings, wherein:

[0022] **FIG. 1** is a diagram of a communication system in which the performance enhancing proxy (PEP) of the present invention is implemented;

[0023] **FIG. 2** is a diagram of a PEP end point platform environment, according to an embodiment of the present invention;

[0024] **FIG. 3** is a diagram of a TCP Spoofing Kernel (TSK) utilized in the environment of **FIG. 2**;

[0025] **FIGS. 4A and 4B** are flow diagrams of the connection establishment with three-way handshake spoofing and without three-way handshake spoofing, respectively;

[0026] **FIG. 5** is a diagram of a PEP packet flow between two PEP end points, according to an embodiment of the present invention;

[0027] **FIG. 6** is a diagram of an IP (Internet Protocol) packet flow through a PEP end point, in accordance with an embodiment of the present invention;

[0028] **FIG. 7** is a diagram of PEP end point profiles utilized in the platform of **FIG. 2**;

[0029] **FIG. 8** is a diagram of the interfaces of a PEP end point implemented as an IP gateway, according to an embodiment of the present invention;

[0030] **FIG. 9** is a diagram of the interfaces of a PEP end point implemented as a Multimedia Relay, according to an embodiment of the present invention;

[0031] **FIG. 10** is a diagram of the interfaces of a PEP end point implemented as a Multimedia VSAT (Very Small Aperture Terminal), according to an embodiment of the present invention;

[0032] **FIG. 11** is a diagram of the interfaces of a PEP end point implemented in an earth station, according to an embodiment of the present invention;

[0033] **FIG. 12** is a diagram of an exemplary network management system (NMS) for PEP end points, according to an embodiment of the present invention;

[0034] **FIG. 13** is a diagram of exemplary arrangements of parameter profiles utilized in the system of **FIG. 12**;

[0035] **FIG. 14** is a diagram of a computer system that can perform PEP functions, in accordance with an embodiment of the present invention;

[0036] **FIG. 15** is diagram of the protocol layers of the TCP/IP protocol suite; and

[0037] **FIG. 16** is diagram of a conventional TCP three-way handshake between IP hosts.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0038] In the following description, for the purpose of explanation, specific details are set forth in order to provide a thorough understanding of the invention. However, it will be apparent that the invention may be practiced without these specific details. In some instances, well-known structures and devices are depicted in block diagram form in order to avoid unnecessarily obscuring the invention.

[0039] Although the present invention is discussed with respect to the Internet and the TCP/IP protocol suite, the present invention has applicability to other packet switched networks and equivalent protocols.

[0040] **FIG. 1** illustrates an exemplary network **100** in which the performance enhancing proxy (PEP) of the present invention may be utilized. The network **100** in **FIG. 1** includes one or more hosts **110** connected to a network gateway **120** via TCP connections. The network gateway **120** is connected to another network gateway **140** via a backbone connection on a backbone link **130**. As seen in **FIG. 1**, the backbone link **130**, in an exemplary embodiment, is shown as a satellite link that is established over a satellite **101**; however, it is recognized by one of ordinary skill in the art that other network connections may be implemented. For example, these network connections may be established over a wireless communications system, in general, (e.g., radio networks, cellular networks, etc.) or a terrestrial communications system. The network gateway **140** is further connected to a second group of hosts **150**, also via TCP connections. In the arrangement illustrated in **FIG. 1**, the network gateways **120**, **140** facilitate communication between the groups of hosts **110**, **150**.

[0041] The network gateways **120**, **140** facilitate communication between the two groups of hosts **110**, **150** by performing a number of performance enhancing functions. These network gateways **120**, **140** may perform selective TCP spoofing, which allows flexible configuration of the particular TCP connections that are to be spoofed. Additionally, gateways **120**, **140** employ a TCP three-way handshake, in which the TCP connections are terminated at each end of the backbone link **130**. Local data acknowledgements are utilized by the network gateways **120**, **140**, thereby permitting the TCP windows to increase at local speeds.

[0042] The network gateway **120**, **140** further multiplexes multiple TCP connections across a single backbone connection; this capability reduces the amount of acknowledgement traffic associated with the data from multiple TCP connections, as a single backbone connection acknowledgement may be employed. The multiplexing function also provides support for high throughput TCP connections, wherein the backbone connection protocol is optimized for the particular backbone link that is used. The network gateways **120**, **140** also support data compression over the backbone link **130** to reduce the amount of traffic to be sent, further leveraging the capabilities of the backbone connection. Further, the network gateways **120**, **140** utilize data encryption in the data transmission across the backbone link **130** to protect data privacy, and provide prioritized access to backbone link **130** capacity on a per TCP connection basis. Each of the network gateways **120**, **140** may select a particular path for the data associated with a connection to flow. The above capabilities of the network gateways **120**, **140** are more fully described below.

[0043] **FIG. 2** illustrates a performance enhancing proxy (PEP) **200** as implemented in a network gateway **120**, **140**, according to one embodiment of the present invention. In this embodiment, the PEP **200** has a platform environment **210**, which includes the hardware and software operating system. The PEP **200** also includes local area network (LAN) interfaces **220** and wide area network (WAN) interfaces **230**. In the example in **FIG. 1**, the network gateway **120** may establish the TCP connections with the IP hosts **110**, via a local LAN interface **220** and may establish the backbone connection with the network gateway **140** via a WAN interface **230**. The PEP platform environment **210** may also include general functional modules: routing module **240**, buffer management module **250**, event management module **260**, and parameter management module **270**. As illustrated in **FIG. 2**, the network gateway also includes a TCP spoofing kernel (TSK) **280**, a backbone protocol kernel (BPK) **282**, a prioritization kernel (PK) **284**, and a path selection kernel (PSK) **286**. These four kernels essentially make up the functionality of the performance enhancing proxy **200**.

[0044] The platform environment **210** performs a number of functions. One such function is to shield the various PEP kernels **280**, **282**, **284**, **286** from implementation specific constraints. That is, the platform environment **210** performs functions that the various PEP kernels **280**, **282**, **284**, **286** cannot perform directly because the implementation of the function is platform specific. This arrangement has the advantageous effect of hiding platform specific details from the PEP kernels **280**, **282**, **284**, **286**, making the PEP kernels more portable. An example of a platform specific function is the allocation of a buffer. In some platforms, buffers are created as they are needed, while in other platforms, buffers are created at start-up and organized into linked lists for later use. It is noted that platform specific functions are not limited to functions generic to all of the kernels **280**, **282**,

284, **286**. A function specific to a particular kernel, for example, the allocation of a control block for TCP spoofing, may also be implemented in the platform environment to hide platform specific details from the kernel.

[**0045**] Additionally, the platform environment **210** may provide the task context in which the PEP kernels **280**, **282**, **284**, **286** run. In one exemplary embodiment, all PEP kernels **280**, **282**, **284**, **286** can run in the same task context for efficiency. However, this is not required.

[**0046**] Furthermore, the platform environment **210**, in an exemplary embodiment, provides an interface between the PEP functionality (embodied in kernels **280**, **282**, **284**, **286**) and the other functionality of the network gateway **120**, **140**. The platform environment **210** may provide the interface between the PEP functionality and the routing function **240**, as seen in **FIG. 2**. It is noted that the platform specific functions illustrated in **FIG. 2** are examples and are not considered an exhaustive list. It is further noted that the PEP kernels shown touching each other (**280**, **282** and **284**, **286**) in **FIG. 2** may have a direct procedural interface to each other. Further, the kernels **280**, **282**, **284**, **286** may include direct interfaces to improve performance, as opposed to routing everything through the platform environment **210** (as shown in **FIG. 2**).

[**0047**] In addition to the PEP kernels **280**, **282**, **284**, and **286**, the PEP end point platform **210** may utilize a data compression kernel (CK) **290** and an encryption kernel (EK) **292**. These kernels **280**, **282**, **284**, **286**, **290**, and **292**, as described above, facilitate communication between the two groups of hosts **110**, **150**, by performing a variety of performance enhancing functions, either singly or in combination. These performance enhancing functions include selective TCP spoofing, three-way handshake spoofing, local data acknowledgement, TCP connection to backbone connection multiplexing, data compression/encryption, prioritization, and path selection.

[**0048**] Selective TCP Spoofing is performed by the TSK **280** and includes a set of user configurable rules that are used to determine which TCP connections should be spoofed. Selective TCP spoofing improves performance by not tying up TCP spoofing-related resources, such as buffer space, control blocks, etc., for TCP connections for which the user has determined that spoofing is not beneficial or required and by supporting the use of tailored parameters for TCP connections that are spoofed.

[**0049**] In particular, the TSK **280** discriminates among the various TCP connections based on the applications using them. That is, TSK **280** discriminates among these TCP connections to determine which connection should be spoofed as well as the manner in which the connection is spoofed; e.g., whether to spoof the three-way handshake, the particular timeout parameters for the spoofed connections, etc. TCP spoofing is then performed only for those TCP connections that are associated with applications for which high throughput or reduced connection startup latency (or both) is required. As a result, the TSK **280** conserves TCP spoofing resources for only those TCP connections for which high throughput or reduced connection startup latency (or both) is required. Further, the TSK **280** increases the total number of TCP connections which can be active before running out of TCP spoofing resources, since any active TCP connections which do not require high throughput are not allocated resources.

[**0050**] One criterion for identifying TCP connections of applications for which TCP spoofing should and should not be performed is the TCP port number field contained in the TCP packets being sent. In general, unique port numbers are assigned to each type of application. Which TCP port numbers should and should not be spoofed can be stored in the TSK **280**. The TSK **280** is also re-configurable to allow a user or operator to reconfigure the TCP port numbers which should and should not be spoofed. The TSK **280** also permits a user or operator to control which TCP connections are to be spoofed based on other criteria. In general, a decision on whether to spoof a TCP connection may be based on any field within a TCP packet. The TSK **280** permits a user to specify which fields to examine and which values in these fields identify TCP connections that should or should not be spoofed. Another example of a potential use for this capability is for the user or operator to select the IP address of the TCP packet in order to control for which users TCP spoofing is performed. The TSK **280** also permits a user to look at multiple fields at the same time. As a result, the TSK **280** permits a user or operator to use multiple criteria for selecting TCP connections to spoof. For example, by selecting both the IP address and the TCP port number fields, the system operator can enable TCP spoofing for only specific applications from specific users.

[**0051**] The user configurable rules may include five exemplary criteria which can be specified by the user or operator in producing a selective TCP spoofing rule: Destination IP address; Source IP address; TCP port numbers (which may apply to both the TCP destination and source port numbers); TCP options; and IP differentiated services (DS) field. However, as indicated above, other fields within the TCP packet may be used.

[**0052**] As discussed above, in addition to supporting selective TCP spoofing rules for each of these criterion, AND and OR combination operators can be used to link criteria together. For example, using the AND combination operator, a rule can be defined to disable TCP spoofing for FTP data received from a specific host. Also, the order in which the rules are specified may be significant. It is possible for a connection to match the criteria of multiple rules. Therefore, the TSK **280** can apply rules in the order specified by the operator, taking the action of the first rule that matches. A default rule may also be set which defines the action to be taken for TCP connections which do not match any of the defined rules. The set of rules selected by the operator may be defined in a selective TCP spoofing selection profile.

[**0053**] As an example, assuming sufficient buffer space has been allocated to spoof five TCP connections, if four low speed applications (i.e., applications which, by their nature, do not require high speed) bring up connections along with one high speed application, the high speed connection has access to only ⅕ of the available spoofing buffer space. Further, if five low speed connections are brought up before the high speed connection, the high speed connection cannot be spoofed at all. Using the TSK **280** selective spoofing mechanism, the low speed connections are not allocated any spoofing buffer space. Therefore, the high speed connection always has access to all of the buffer space, improving its performance with respect to an implementation without the selective TCP spoofing feature of the TSK **280**.

[0054] The TSK **280** also facilitates spoofing of the conventional three-way handshake. Three-Way Handshake Spoofing involves locally responding to a connection request to bring up a TCP connection in parallel with forwarding the connection requests across the backbone link **130** (**FIG. 1**). This allows the originating IP host (for example, **110**) to reach the point of being able to send the data it must send at local speeds, i.e. speeds that are independent of the latency of the backbone link **130**. Three-way Handshake Spoofing allows the data that the IP host **110** needs to send to be sent to the destination IP host **150** without waiting for the end-to-end establishment of the TCP connection. For backbone links **130** with high latency, this significantly reduces the time it takes to bring up the TCP connection and, more importantly, the overall time it takes to get a response (from an IP host **150**) to the data the IP host **110** sends.

[0055] A specific example in which this technique is useful relates to an Internet web page access application. With three-way handshake spoofing, an IP host's request to retrieve a web page can be on its way to a web server without waiting for the end-to-end establishment of the TCP connection, thereby reducing the time it takes to download the web page.

[0056] With Local Data Acknowledgement, the TSK **280** in the network gateway **120** (for example) locally acknowledges data segments received from the IP host **110**. This allows the sending IP host **110** to send additional data immediately. More importantly, TCP uses received acknowledgements as signals for increasing the current TCP window size. As a result, local sending of the acknowledgements allows the sending IP host **110** to increase it TCP window at a much faster rate than supported by end to end TCP acknowledgements. The TSK **280** (the spoofer) takes on the responsibility for reliable delivery of the data which it has acknowledged.

[0057] In the BPK **282**, multiple TCP connections are multiplexed onto and carried by a single backbone connection. This improves system performance by allowing the data for multiple TCP connections to be acknowledged by a single backbone connection acknowledgement (ACK), significantly reducing the amount of acknowledgement traffic required to maintain high throughput across the backbone link **130**. In addition, the BPK **282** selects a backbone connection protocol that is optimized to provide high throughput for the particular link. Different backbone connection protocols can be used by the BPK **282** with different backbone links without changing the fundamental TCP spoofing implementation. The backbone connection protocol selected by the BPK **282** provides appropriate support for reliable, high speed delivery of data over the backbone link **130**, hiding the details of the impairments (for example high latency) of the link from the TCP spoofing implementation.

[0058] The multiplexing by the BPK **282** allows for the use of a backbone link protocol which is individually tailored for use with the particular link and provides a technique to leverage the performance of the backbone link protocol with much less dependency upon the individual performance of the TCP connections being spoofed than conventional methods. Further, the ability to tailor the backbone protocol for different backbone links makes the present invention applicable to many different systems.

[0059] The PEP **200** may optionally include a data compression kernel **290** for compressing TCP data and an encryption kernel **292** for encrypting TCP data. Data compression increases the amount of data that can be carried across the backbone connection. Different compression algorithms can be supported by the data compression kernel **290** and more than one type of compression can be supported at the same time. The data compression kernel **290** may optionally apply compression on a per TCP connection basis, before the TCP data of multiple TCP connections is multiplexed onto the backbone connection or on a per backbone connection basis, after the TCP data of multiple TCP connections has been multiplexed onto the backbone connection. Which option is used is dynamically determined based on user configured rules and the specific compression algorithms being utilized. Exemplary data compression algorithms are disclosed in U.S. Pat. Nos. 5,973,630, 5,955,976, the entire contents of which are hereby incorporated by reference. The encryption kernel **292** encrypts the TCP data for secure transmission across the backbone link **130**. Encryption may be performed by any conventional technique. It is also understood that the corresponding spoofer (in the example outlined above, the network gateway **140**) includes appropriate kernels for decompression and decryption, both of which may be performed by any conventional technique.

[0060] The PK **284** provides prioritized access to the backbone link capacity. For example, the backbone connection can actually be divided into N (N>1) different sub-connections, each having a different priority level. In one exemplary embodiment, four priority levels can be supported. The PK **284** uses user-defined rules to assign different priorities, and therefore different sub-connections of the backbone connection, to different TCP connections. It should be noted that PK **284** may also prioritize non-TCP traffic (e.g., UDP (User Datagram Protocol) traffic) before sending the traffic across the backbone link **130**.

[0061] The PK **284** also uses user-defined rules to control how much of the backbone link **130** capacity is available to each priority level. Exemplary criteria which can be used to determine priority include the following: Destination IP address; Source IP address; IP next protocol; TCP port numbers (which may apply to both the TCP destination and source port numbers); UDP port numbers (which may apply to both the UDP destination and source port numbers); and IP differentiated services (DS) field. The type of data in the TCP data packets may also be used as a criterion. For example, video data could be given highest priority. Mission critical data could also be given high priority. As with selective TCP spoofing, any field in the IP packet can be used by PK **284** to determine priority. However, it should be noted that under some scenarios the consequence of using such a field may cause different IP packets of the same flow (e.g., TCP connection) to be assigned different priorities; these scenarios should be avoided.

[0062] As mentioned above, in addition to supporting selective prioritization rules for each of these criteria, AND and OR combination operators can be used to link criteria together. For example, using the AND combination operator, a rule can be defined to assign a priority for SNMP data received from a specific host. Also, the order in which the rules are specified may be significant. It is possible for a connection to match the criteria of multiple rules. Therefore,

the PK **284** can apply rules in the order specified by the operator, taking the action of the first rule that matches. A default rule may also be set which defines the action to be taken for IP packets which do not match any of the defined rules. The set of rules selected by the operator may be defined in a prioritization profile.

[**0063**] As regards the path selection functionality, the PSK **286** is responsible for determining which path an IP packet should take to reach its destination. The path selected by the PSK **286** can be determined by applying path selection rules. The PSK **286** also determines which IP packets should be forwarded using an alternate path and which IP packets should be dropped when one or more primary paths fail. Path selection parameters can also be configured using profiles. The path selection rules may be designed to provide flexibility with respect to assigning paths while making sure that all of the packets related to the same traffic flow (e.g., the same TCP connection) take the same path (although it is also possible to send segments of the same TCP connection via different paths, this segment "splitting" may have negative side effects). Exemplary criteria that can be used to select a path include the following: priority of the IP packet as set by the PK **284** (should be the most common criterion): Destination IP address; Source IP address; IP next protocol; TCP port numbers (which may apply to both the TCP destination and source port numbers); UDP port numbers (which may apply to both the UDP destination and source port numbers); and IP differentiated services (DS) field. Similar to selective TCP spoofing and prioritization, the PSK **284** may determine a path by using any field in the IP packet.

[**0064**] As with the prioritization criteria (rules) the AND and OR combination operators can be used to link criteria together. For example, using the AND combination operator, a rule can be defined to select a path for SNMP data received from a specific host. Also, the order in which the rules are specified may be significant. It is possible for a connection to match the criteria of multiple rules. Therefore, the PSK **286** can apply rules in the order specified by the operator, taking the action of the first rule that matches. A default rule may also be set which defines the action to be taken for IP packets which do not match any of the defined rules. The set of rules selected by the operator may be defined in a path selection profile.

[**0065**] By way of example, a path selection rule may select the path based on any of the following path information in which IP packets match the rule: a primary path, a secondary path, and a tertiary path. The primary path is be specified in any path selection rule. The secondary path is used only when the primary path has failed. If no secondary path is specified, any IP packets that match the rule can be discarded when the primary path fails. The tertiary path is specified only if a secondary path is specified. The tertiary path is selected if both the primary and secondary paths have failed. If no tertiary path is specified, any IP packets that match the rule can be discarded when both the primary and secondary paths fail. Path selection may be generalized such that the path selection rule can select up to N paths where the Nth path is used only if the (N−1)th path fails. The example above where N=3 is merely illustrative, although N is typically a fairly small number.

[**0066**] By way of example, the operation of the system **100** is described as follows. First, a backbone connection is established between the PEPs **200** of two network gateways **120**, **140** (i.e., the two spoofers), located at each end of the backbone link **130** for which TCP spoofing is desired. Whenever an IP host **110** initiates a TCP connection, the TSK **280** of the PEP **200** local to the IP host **110** checks its configured selective TCP spoofing rules. If the rules indicate that the connection should not be spoofed, the PEP **200** allows the TCP connection to flow end-to-end unspoofed. If the rules indicate that the connection should be spoofed, the spoofing PEP **200** locally responds to the IP host's TCP three-way handshake. In parallel, the spoofing PEP **200** sends a message across the backbone link **130** to its partner network gateway **140** asking it to initiate a TCP three-way handshake with the IP host **150** on its side of the backbone link **130**. Data is then exchanged between the IP host **110**, **150** with the PEP **200** of the network gateway **120** locally acknowledging the received data and forwarding it across the backbone link **130** via the high speed backbone connection, compressing the data as appropriate based on the configured compression rules. The priority of the TCP connection is determined when the connection is established. The BPK **282** can multiplex the connection with other received connections over a single backbone connection, the PK **284** determines the priority of the connection and the PSK **286** determines the path the connection is to take.

[**0067**] The PEP **200**, as described above, advantageously improves network performance by allocating TCP spoofing-related resources, such as buffer space, control blocks, etc., only to TCP connections for which spoofing is beneficial; by spoofing the three-way handshake to decrease data response time; by reducing the number of ACKs which are transmitted by performing local acknowledgement and by acknowledging multiple TCP connections with a single ACK; by performing data compression to increase the amount of data that can be transmitted; by assigning priorities to different connections; and by defining multiple paths for connections to be made.

[**0068**] **FIG. 3** shows an exemplary stack, which illustrates the relationship between the TCP stack and the PEP kernels **280**, **282**, **284**, **286** of the present invention. The TSK **280** is primarily responsible for functions related to TCP spoofing. The TSK **280**, in an exemplary embodiment, includes two basic elements: a transport layer that encompasses a TCP stack **303** and an IP stack **305**; and a TCP spoofing application **301**. The transport layer is responsible for interacting with the TCP stacks (e.g., **303**) of IP hosts **110** connected to a local LAN interface **220** of a PEP **210**.

[**0069**] The TSK **280** implements the TCP protocol, which includes the appropriate TCP state machines and terminates spoofed TCP connections. The TCP spoofing application **301** rests on top of the transport layer and act as the application that receives data from and sends data to the IP hosts **110** applications. Because of the layered architecture of the protocol, the TCP spoofing application **301** isolates the details of TCP spoofing from the transport layer, thereby allowing the transport layer to operate in a standard fashion.

[**0070**] As shown in **FIG. 3**, the TCP spoofing application **301** can also interface to the BPK **282** associated with the WAN interfaces **230**. The BPK **282** performs backbone protocol maintenance, implementing the protocol by which the network gateways **120**, **140** (in **FIG. 1**) communicate.

The BPK **282** provides reliable delivery of data, uses a relatively small amount of acknowledgement traffic, and supports generic backbone use (i.e., use not specific to the TSK **280**); one such example is the reliable data protocol (RDP).

[**0071**] The BPK **282** lies above the PK **284** and the PSK **286**, according to an exemplary embodiment. The PK **284** is responsible for determining the priority of IP packets and then allocating transmission opportunities based on priority. The PK **284** can also control access to buffer space by controlling the queue sizes associated with sending and receiving IP packets. The PSK **286** determines which path an IP packet should take to reach its destination. The path selected by the PSK **286** can be determined applying path selection rules. PSK **286** may also determine which IP packet should be forwarded using an alternate path and which packets should be dropped when one or more primary paths fail.

[**0072**] **FIGS. 4A and 4B** show flow diagrams of the establishment of a spoofed TCP connection utilizing three-way handshake spoofing and without three-way handshake spoofing, respectively. The TCP Spoofing Kernel **280** establishes a spoofed TCP connection when a TCP <SYN> segment is received from its local LAN or a Connection Request message from its TSK peer. It is noted that the three-way handshake spoofing may be disabled to support an end to end maximum segment size (MSS) exchange, which is more fully described below. For the purpose of explanation, the spoofed TCP connection establishment process is described with respect to a local host **400**, a local PEP end point **402**, a remote PEP end point **404**, and a remote host **406**. As mentioned previously, the TSK **280** within each of the PEP end points **402** and **404** provides the spoofing functionality.

[**0073**] In step **401**, the local host **400** transmits a TCP <SYN> segment to the local PEP end point **402** at a local LAN interface **220**. When a TCP segment is received from the local LAN interface **220**, the platform environment **402** determines whether there is already a TCP connection control block (CCB) assigned to the TCP connection associated with the TCP segment. If there is no CCB, the environment **402** checks whether the TCP segment is a <SYN> segment that is being sent to a non-local destination. If so, the <SYN> segment represents an attempt to bring up a new (non-local) TCP connection, and the environment **402** passes the segment to the TCP Spoofing Kernel **280** to determine the TCP connection's disposition. When a TCP <SYN> segment is received from the local LAN interface **220** for a new TCP connection, the TCP Spoofing Kernel **280** first determines if the connection should be spoofed. If the connection should be spoofed, TSK **280** uses (in an exemplary embodiment) the priority indicated in the selected TCP spoofing parameter profile and the peer index (provided by the environment **210** with the TCP <SYN> segment) to construct the handle of the backbone connection which should be used to carry this spoofed TCP connection. In the exemplary embodiment, the peer index is used as the 14 high order bits of the handle and the priority is used as the two low order bits of the handle. The backbone connection handle is then used (via the TSK control block (TCB) mapping table) to find the TCB associated with the backbone connection. TSK **280** of PEP end point **402** then checks whether the backbone connection is up. If the backbone connection is up, TSK **280** determines whether the number of spoofed TCP connections that are already using the selected backbone connection is still currently below the CCB resource limit. The CCB resource limit is the smaller of the local number of CCBs (provided as a parameter by the platform environment **210**) and the peer number of CCBs (received in the latest TSK peer parameters (TPP) message from the TSK peer) available for this backbone connection. If the number of connections is still below the limit, TSK **280** of PEP end point **402** assigns a unique TCP connection identifier (e.g., a free CCB mapping table entry index) to the connection and calls the environment **210** to allocate a TCP connection control block for the connection.

[**0074**] TSK **280** of PEP end point **402** returns the TCP <SYN> segment back to the environment **210** to be forwarded unspoofed if any of the above checks fail. In other words, the following conditions result in the TCP connection being unspoofed. First, if the selective TCP spoofing rules indicate that the connection should not be spoofed. Also, there is no backbone connection for the priority at which the TCP connection should be spoofed (indicated by the absence of a TCB for the backbone connection). No spoofing is performed if the backbone connection is down. Additional, if the number of spoofed TCP connections that are already using the backbone connection reaches or exceeds a predetermined threshold, then no spoofing is performed. Further, if there is no CCB mapping table entry available or there is no CCB available from the CCB free pool, then the TCP connection is forwarded unspoofed. For the case in which there is no backbone connection, TSK **280** of PEP end point **402** may also post an event to alert the operator that there is a mismatch between the configured TCP spoofing parameter profiles and the configured set of backbone connections.

[**0075**] Continuing with the example, if all of the above checks pass, TSK **280** of PEP end point **402** writes the backbone connection handle into the buffer holding the TCP <SYN> segment. It is noted that this is not done until a CCB is successfully allocated by the platform environment **402**, because the environment does not count the buffer unless a CCB is successfully allocated. TSK **280** then copies the parameters from the selected TCP spoofing parameter profile into the CCB. Consequently, relevant information (e.g., the maximum segment size that is advertised by the host (if smaller than the configured MSS), the initial sequence number, and etc.) is copied out of the TCP <SYN> segment and stored in the CCB. It is noted that the source and destination IP addresses and source and destination TCP port numbers will already have been placed into the CCB by the platform environment **402** when the CCB was allocated; the environment **402** uses this information to manage CCB hash function collisions.

[**0076**] After allocating and setting up the CCB, the TCP Spoofing Kernel **280** of PEP end point **402** constructs a Connection Request (CR) message, per step **403**, and sends it to its TSK peer associated with the remote PEP end point **404**. The CR message basically contains all of the information extracted from the TCP spoofing parameter profile and the TCP <SYN> segment and stored in the local CCB, e.g., the source and destination IP addresses, the source and destination TCP port numbers, the MSS value, etc., with the exception of fields that have only local significance, such as the initial sequence number. (The IP addresses and TCP port numbers are placed into a TCP connection header.) In other

words, the CR message contains all of the information that the peer TSK of PEP end point **404** requires to set up its own CCB. To complete the local connection establishment, the TCP Spoofing Kernel **280** of the local PEP end point **402** sends a TCP <SYN,ACK> segment to the local host **400** in response to the <SYN> segment received, per step **405**. TSK **280** of PEP end point **402** performs step **405** simultaneously with the step of sending the Connection Request message (i.e., step **403**), if three-way handshake spoofing is enabled. Otherwise, TSK **280** of **402** waits for a Connection Established (CE) message from its TSK peer of the remote PEP end point **404** before sending the <SYN,ACK> segment. In an exemplary embodiment, TSK **280** of PEP end point **402** selects a random initial sequence number (as provided in IETF (Internet Engineering Task Force) RFC **793**, which is incorporated herein by reference in its entirety) to use for sending data.

[**0077**] If three-way handshake spoofing is disabled, the MSS value sent in the <SYN,ACK> segment is set equal to the MSS value received in the CE message. If three-way handshake spoofing is enabled, the MSS value is determined from the TCP spoofing parameter profile selected for the connection (and the configured path maximum transmission unit (MTU)). For this case, TSK **280** of PEP end point **402** then compares the MSS value received in the Connection Established message, when it arrives, to the value it sent to the local host in the TCP <SYN,ACK> segment. If the MSS value received in the CE message is smaller than the MSS value sent to the local host, a maximum segment size mismatch exists. (If an MSS mismatch exists, TSK may need to adjust the size of TCP data segments before sending them.) After sending the TCP <SYN,ACK> segment (step **405**), TSK **280** of the local PEP end point **402** is ready to start accepting data from the local host **400**. In step **407**, the local host **400** transmits an <ACK> segment to the TSK **280** of PEP end point **402**; thereafter, the local host forwards, as in step **409** data to the TSK **280** of PEP end point **402** as well. When three-way handshake spoofing is being used, TSK **280** does not need to wait for the Connection Established message to arrive from its TSK peer before accepting and forwarding data. As seen in **FIG. 4**A, in step **411**, TSK **280** of the local PEP end point **402** sends an <ACK> segment to the local host and simultaneously sends the TCP data (TD) from the local host **400** to the peer TSK of PEP end point **404** (per step **413**) prior to receiving a CE message from the peer TSK of PEP end point **404**.

[**0078**] However, TSK **280** of PEP end point **402** does not accept data from its TSK peer of PEP end point **404** until after the CE message has been received. TSK **280** of PEP end point **402** does not forward any data received from its TSK peer of PEP end point **404** to the local host **400** until it has received the TCP <ACK> segment indicating that the local host has received the <SYN,ACK> segment (as in step **407**).

[**0079**] When a Connection Request message is received from a peer TSK (step **403**), the TCP Spoofing Kernel **280** allocates a CCB for the connection and then stores all of the relevant information from the CR message in the CCB. TSK **280** of PEP end point **404** then uses this information to generate a TCP <SYN> segment, as in step **415**, to send to the remote host **406**. The MSS in the <SYN> segment is set to the value received from the TSK peer of PEP end point **404**. When the remote host responds with a TCP <SYN, ACK> segment (step **417**), TSK **280** of PEP end point **402** sends a Connection Established message to its TSK peer of the remote PEP end point **404** (step **419**), including in the CE message the MSS that is sent by the local host in the <SYN,ACK> segment. TSK **280** of PEP end point **402** also responds, as in step **421**, with a TCP <ACK> segment to complete the local three-way handshake. The peer TSK of PEP end point **404** then forwards the data that is received from TSK **280** to the host, per step **423**. Concurrently, in step **425**, the remote host **406** sends data to the peer TSK of PEP end point **404**, which acknowledges receipt of the data by issuing an <ACK> segment to the remote PEP end point **404**, per step **427**. Simultaneously with the acknowledgement, the data is sent to TSK **280** of PEP end point **402** (step **429**).

[**0080**] At this point, TSK **280** is ready to receive and forward data from either direction. TSK **280** forwards the data, as in step **431** to the local host, which, in turn, sends an <ACK> segment (step **433**). If the data arrives from its TSK peer before a <SYN,ACK> segment response is received from the local host, the data is queued and then sent after the <ACK> segment is sent in response to the <SYN, ACK> segment (when it arrives).

[**0081**] Turning now to **FIG. 4B, a** spoofed TCP connection is established with the three-way handshake spoofing disabled. Under this scenario, the local host **400** transmits a TCP <SYN> segment, as in step **451**, to the TSK **280** within the local PEP end point **402**. Unlike the TCP connection establishment of **FIG. 4**A, the local PEP end point **402** does not respond to the a TCP <SYN> segment with a <SYN, ACK> segment, but merely forwards a CR message to the remote PEP end point **404** (step **453**). Next, in step **455**, sends a TCP <SYN> segment to the remote host **406**. In response, the remote host **406** transmit a TCP <SYN,ACK> segment back to the remote PEP end point **404** (per step **457**). Thereafter, the remote PEP end point **404**, as in step **459**, forwards a CE message to the local PEP end point **402**, which subsequently issues a <SYN,ACK> segment to the local host **400**, per step **461**. Simultaneous with step **459**, the remote PEP end point **404** issues an <ACK> segment to the remote host **406** (step **463**).

[**0082**] Upon receiving the <ACK> segment, the remote host **406** may begin transmission of data, as in step **465**. Once the PEP end point **404** receives the data from the remote host **406**, the remote PEP end point **404** simultaneously transmits, as in step **467**, the TD message to the local PEP end point **402** and transmits an <ACK> segment to the remote host **406** to acknowledge receipt of the data (step **469**).

[**0083**] Because the local host **400** has received a <SYN, ACK> segment from the local PEP end point **402**, the local host **400** acknowledges the message, per step **471**. Thereafter, the local host **400** transmits data to the local PEP end point **402**. In this example, before the local PEP end point **402** receives the data from the local host **400**, the local PEP end point **402** forwards the data that originated from the remote host **406** via the TD message (step **467**) to the local host **400**, per step **475**.

[**0084**] In response to the data received (in step **473**), the local PEP end point **402** issues an <ACK> segment, as in step **477**, and forwards the data in a TD message to the remote PEP end point **404**, per step **479**. The local host **400**

responds to the received data of step **475** with an <ACK> segment to the local PEP end point **402** (step **481**). The remote PEP end point **404** sends the data from the local host **400**, as in step **483**, upon receipt of the TD message. After receiving the data, the remote host **406** acknowledges receipt by sending an <ACK> segment back to the remote PEP end point **404**, per step **485**.

[**0085**] **FIG. 5** shows the flow of packets with the PEP architecture, according to one embodiment of the present invention. As shown, a communication system **500** includes a hub site (or local) PEP end point **501** that has connectivity to a remote site PEP end point **503** via a backbone connection. By way of example, at the hub site (or local site) and at each remote site, PEP end points **501** and **503** handle IP packets. PEP end point **501** includes an internal IP packet routing module **501***a* that receives local IP packets and exchanges these packets with a TSK **501***b* and a BPK **501***c*. Similarly, the remote PEP end point **503** includes an internal IP packet routing module **503***a* that is in communication with a TSK **503***b* and a BPK **503***c*. Except for the fact that the hub site PEP end point **501** may support many more backbone protocol connections than a remote site PEP end point **503**, hub and remote site PEP processing is symmetrical.

[**0086**] For local-to-WAN traffic (i.e., upstream direction), the PEP end point **501** receives IP packets from its local interface **220** (**FIG. 2**). Non-TCP IP packets are forwarded (as appropriate) to the WAN interface **230** (**FIG. 2**). TCP IP packets are internally forwarded to TSK **501***b*. TCP segments which belong to connections that are not be spoofed are passed back by the spoofing kernel **501***b* to the routing module **501***a* to be forwarded unmodified to the WAN interface **230**. For spoofed TCP connections, the TCP spoofing kernel **501***a* locally terminates the TCP connection. TCP data that is received from a spoofed connection is passed from the spoofing kernel **501***a* to the backbone protocol kernel **501***c*, and then multiplexed onto the appropriate backbone protocol connection. The backbone protocol kernel **501***c* ensures that the data is delivered across the WAN.

[**0087**] For WAN-to-local traffic (i.e., downstream direction), the remote PEP end point **503** receives IP packets from its WAN interface **230** (**FIG. 2**). IP packets that are not addressed to the end point **503** are simply forwarded (as appropriate) to the local interface **220** (**FIG. 2**). IP packets addressed to the end point **503**, which have a next protocol header type of "PBP" are forwarded to the backbone protocol kernel **503***c*. The backbone protocol kernel **503***c* extracts the TCP data and forwards it to the TCP spoofing kernel **503***b* for transmission on the appropriate spoofed TCP connection. In addition to carrying TCP data, the backbone protocol connection is used by the TCP spoofing kernel **501***b* to send control information to its peer TCP spoofing kernel **503***b* in the remote PEP end point **503** to coordinate connection establishment and connection termination.

[**0088**] Prioritization may be applied at four points in the system **500** within routing **501***a* and TSK **501***b* of PEP end point **501**, and within routing **503***a*, and TSK **503***b* of PEP end point **503**. In the upstream direction, priority rules are applied to the packets of individual TCP connections at the entry point to the TCP spoofing kernel **501***b*. These rules allow a customer to control which spoofed applications have higher and lower priority access to spoofing resources. Upstream prioritization is also applied before forwarding packets to the WAN. This allows a customer to control the relative priority of spoofed TCP connections with respect to unspoofed TCP connections and non-TCP traffic (as well as to control the relative priority of these other types of traffic with respect to each other). On the downstream side, prioritization is used to control access to buffer space and other resources in the PEP end point **503**, generally and with respect to TCP spoofing.

[**0089**] At the hub (or local) site, the PEP end point **501** may be implemented in a network gateway (e.g. an IP Gateway), according to one embodiment of the present invention. At the remote site, the PEP end point **503** may be implemented in the remote site component, e.g. a satellite terminal such as a Multimedia Relay, a Multimedia VSAT or a Personal Earth Station (PES) Remote.

[**0090**] The architecture of system **500** provides a number of advantages. First, TCP spoofing may be accomplished in both upstream and downstream directions. Additionally, the system supports spoofing of TCP connection startup, and selective TCP spoofing with only connections that can benefit from spoofing actually spoofed. Further, system **500** enables prioritization among spoofed TCP connections for access to TCP spoofing resources (e.g., available bandwidth and buffer space). This prioritization is utilized for all types of traffic that compete for system resources.

[**0091**] With respect to the backbone connection, the system **500** is suitable for application to a satellite network as the WAN. That is, the backbone protocol is optimized for satellite use in that control block resource requirements are minimized, and efficient error recovery for dropped packets are provided. The system **500** also provides a feedback mechanism to support maximum buffer space resource efficiency. Further, system **500** provides reduced acknowledgement traffic by using a single backbone protocol ACK to acknowledge the data of multiple TCP connections.

[**0092**] **FIG. 6** illustrates the flow of IP packets through a PEP end point, according to an embodiment of the present invention. When IP packets are received at the local LAN interface **220**, the PEP end point **210** determines (as shown by decision point A), whether the packets are destined for a host that is locally situated; if so, the IP packets are forwarded to the proper local LAN interface **220**. If the IP packets are destined for a remote host, then the PEP end point **210** decides, per decision point B, whether the traffic is a TCP segment. If the PEP end point **210** determines that in fact the packets are TCP segments, then the TSK **280** determines whether the TCP connection should be spoofed. However, if the PEP end point **210** determines that the packets are not TCP segments, then the BPK **282** processes the traffic, along with the PK **284** and the PSK **286** for eventual transmission out to the WAN. It should be noted that the BPK **282** does not process unspoofed IP packets; i.e., the packets flow directly to PK **284**. As seen in **FIG. 6**, traffic that is received from the WAN interface **230** is examined to determine whether the traffic is a proper PBP segment (decision point D) for the particular PEP end point **210**; if the determination is in the affirmative, then the packets are sent to the BPK **282** and then the TSK **280**.

[**0093**] Routing support includes routing between the ports of the PEP End Point **210** (**FIG. 2**), e.g., from one Multi-

media VSAT LAN port to another. Architecturally, the functionalities of TCP spoofing, prioritization and path selection, fit between the IP routing functionality and the WAN. PEP functionality need not be applied to IP packets which are routed from local port to local port within the same PEP End Point **210**. TCP spoofing, prioritization and path selection are applied to IP packets received from a local PEP End Point interface that have been determined to be destined for another site by the routing function.

[**0094**] **FIG. 7** shows the relationship between PEP End Points and PEP End Point profiles, in accordance with an embodiment of the present invention. PEP parameters are primarily configured via a set of profiles **701** and **703**, which are associated with one or more PEP end points **705**. In an exemplary embodiment, PEP parameters are configured on a per PEP End Point basis, such as whether TCP spoofing is globally enabled. These parameters are configured in the PEP End Point profiles **701** and **703**. It is noted that parameters that apply to specific PEP kernels may be configured via other types of profiles. Profiles **701** and **703** are a network management construct; internally, a PEP End Point **705** processes a set of parameters that are received via one or more files.

[**0095**] Whenever the PEP End Point **705** receives new parameters, the platform environment compares the new parameters to the existing parameters, figures out which of the PEP kernels are affected by the parameter changes, and then passes the new parameters to the affected kernels. In an exemplary embodiment, all parameters are installed dynamically. With the exception of parameters that are component specific (such as the IP addresses of a component), all parameters may be defined with default values.

[**0096**] As mentioned previously, the PEP end point **210** maybe implemented in a number of different platforms, in accordance with the various embodiments of the present invention. These platforms may include an IP gateway, a Multimedia Relay, a Multimedia VSAT (Very Small Aperture Terminal), and a Personal Earth Station (PES) Remote, as shown in FIGS. **8-11**, respectively. In general, as discussed in **FIG. 2**, the PEP end point **210** defines a local LAN interface **220** an interface through which the PEP End Point **210** connects to IP hosts located at the site. A WAN interface **230** is an interface through which the PEP End Point **210** connects to other sites. It is noted that a WAN interface **230** can physically be a LAN port. FIGS. **8-11**, below, describe the specific LAN and WAN interfaces of the various specific PEP End Point platforms. The particular LAN and WAN interfaces that are employed depend on which remote site PEP End Points are being used, on the configuration of the hub and remote site PEP End Points and on any path selection rules which may be configured.

[**0097**] **FIG. 8** shows the interfaces of the PEP end point implemented as an IP gateway, according to one embodiment of the present invention. By way of example, an IP Gateway **801** has a single local LAN interface, which is an enterprise interface **803**. The IP Gateway **803** employs two WAN interfaces **805** for sending and receiving IP packets to and from remote site PEP End Points: a backbone LAN interface and a wide area access (WAA) LAN interface.

[**0098**] The backbone LAN interface **805** is used to send IP packets to remote site PEP End Points via, for example, a Satellite Gateway (SGW) and a VSAT outroute. A VSAT outroute can be received directly by Multimedia Relays (**FIG. 9**) and Multimedia VSATs (**FIG. 10**) (and is the primary path used with these End Points); however, IP packets can be sent to a PES Remote (**FIG. 11**) via a VSAT outroute.

[**0099**] **FIG. 9** shows a Multimedia Relay implementation of a PEP end point, in accordance with an embodiment of the present invention. A Multimedia Relay has two or three local LAN interfaces **903**. A Multimedia Relay **901** has up to two WAN interfaces **905** for sending IP packets to hub site PEP End Points: one of its LAN interfaces and a PPP serial port interface, and four or five interfaces for receiving IP packets from hub site PEP End Points, a VSAT outroute, all of its LAN interfaces, and a PPP serial port interface. It is noted that a PPP (Point-to-Point Protocol) serial port interface and a LAN interface are generally not be used at the same time.

[**0100**] A Multimedia Relay **901** supports the use of all of its LAN interfaces **903** at the same time for sending and receiving IP packets to and from hub site PEP End Points. Further, a Multimedia Relay **905** supports the use of a VADB (VPN Automatic Dial Backup) serial port interface for sending and receiving IP packets to and from the hub site PEP End Points.

[**0101**] **FIG. 10** shows a Multimedia VSAT implementation of the PEP end point, according to one embodiment of the present invention. A Multimedia VSAT **1001**, in an exemplary embodiment, has two local LAN interfaces **1003**. Support for one or more local PPP serial port interfaces may be utilized. The Multimedia VSAT **1001** has two WAN interfaces **1005** for sending IP packets to hub site PEP End Points: a VSAT inroute and one of its LAN interfaces. The Multimedia VSAT **1001** thus has three interfaces for receiving IP packets from hub site PEP End Points, the VSAT outroute and both of its LAN interfaces **1003**. A Multimedia VSAT **1003** may support uses of both of its LAN interfaces **1003** at the same time for sending and receiving IP packets to and from hub site PEP End Points. The Multimedia VSAT **1003** further supports the use of a VADB serial port interface for sending and receiving IP packets to and from the hub site PEP End Points.

[**0102**] **FIG. 11** shows a PES Remote implementation of a PEP end point, according to one embodiment of the present invention. A PES Remote **1101** may have a local LAN interface and/or several local IP (e.g. PPP, SLIP, etc.) serial port interfaces, collectively denoted as LAN interfaces **1103**. The particular LAN interfaces **1103** depend on the specific PES Remote platform. PES Remote **1101**, in an exemplary embodiment, has up to five WAN interfaces **1105** for sending IP packets to hub site PEP End Points, an ISBN inroute, a LAN interface, a VADB serial port interface, a Frame Relay serial port interface and an IP serial port interface, and up to five existing interfaces for receiving IP packets from hub site PEP End Points: an ISBN outroute, a LAN interface, a VADB serial port interface, a Frame Relay serial port interface, and an IP serial port interface. The physical Frame Relay serial port interface may be supporting multiple Permanent Virtual Circuits (PVCs); some of which are equivalent to local interfaces **1103** and some of which are WAN interfaces **1105**.

[**0103**] **FIG. 12** shows a diagram of an exemplary network management system (NMS) for PEP end points, according to an embodiment of the present invention. As shown, a

communication system **1200** includes a hub (or local) site PEP end point **1201** that contains a SNMP agent **1203**. As previously discussed, hub (or local) site PEP end point **1201** may communicate via a WAN **1205** to a remote PEP end point **1207**, which similarly provides a SNMP agent **1209**. In an exemplary embodiment, hub (or local) site PEP end point **1201** connects to a LAN **1211**. A network management system **1213** receives data from SNMP agents **1203** and **1209**. The NMS **1213** maintains a database **1215** that stores an event log to assist in debugging of either of the hub (or local) site PEP end point **1201** or the remote PEP end point. Also, the NMS **1213** includes an operator console **1217** to support logging in of events.

[**0104**] The NMS **1213** may be used to configure and control the IP gateway **801** (**FIG. 8**), the Multimedia Relay **901** (**FIG. 9**), the Multimedia VSAT **1001** (**FIG. 10**), and the PES remote **1101**.

[**0105**] NMS parameters are stored by the NMS **1213** in a database (not shown); the stored data are used to create parameter files. Accordingly, NMS parameters can be downloaded to components as ASCII files with self-defining parameters. The self-defining aspect refers to the fact that parameters are, in general, downloaded in the form Parameter Name=Parameter Value. Other formats may be used, depending on the particular PEP End Point implementation.

[**0106**] As indicated previously, profiles are the primary means of configuring PEP parameters. Generally, most types of PEP profiles may be shared by all PEP End Point platforms **1201** and **1207**. The creation of PEP profiles is supported by the NMS **1213**.

[**0107**] The NMS **1213** supports the use of network management domains (NMDs), which partition the system **1200** into logical domains that are used to control access network management information. In an exemplary embodiment, each component in the system **1200** belongs to exactly one NMD. Similarly, PEP profiles may also belong to exactly one NMD. In the exemplary embodiment, a PEP End Point platform component **1201**, **1207** can only be configured to use profiles that are in the same domain as the component. In an exemplary embodiment, NMDs are primarily used to partition by customer in a shared hub context.

[**0108**] As seen in **FIG. 12**, each PEP End Point platform **1201** and **1207** includes an SNMP Agent **1203** and **1209**, respectively, for processing get and set access to the SNMP MIB variables that are defined for the particular platform **1201** and **1207**. To support access to PEP specific MIB variables, each of the SNMP Agents **1203** and **1209** accesses information kept in corresponding platform environments and PEP kernels. Access to PEP kernel information is provided via the platform environment, which represents the PEP feature to the rest of the platform software (i.e., the rest of the platform software should not even be aware that the PEP kernels exist). And, the mechanism used to access MIB information may not be consistent across all of the PEP End Point platforms; for example, a shared memory mechanism may be used in the IP Gateway, whereas a procedural interface mechanism may be used in a Multimedia VSAT. Therefore, platform specific support may be required (and providing platform specific support is one of the functions of the platform environment).

[**0109**] PEP parameters are generally an addition to the existing configuration for each type of PEP End Point platform. Most, if not all, PEP specific parameters are configured by means of profiles, with the possible exception of the configuration of the backbone connections between PEP End Points **1201**, **1207**.

[**0110**] As noted, profiles are used as the primary means for configuring PEP parameters since most PEP specific parameters are common to multiple PEP End Points. The use of profiles minimizes the amount of repetitive configuration that is required and allows an operator to simultaneously make changes to multiple components by simply changing the appropriate PEP profile. In an exemplary embodiment, the types of parameter profiles include connectivity profiles, PEP End Point profiles, TCP spoofing selection profiles, TCP spoofing parameter profiles, priority profiles, and path selection profiles.

[**0111**] To simplify configuration, the NMS **1213** may maintain a default version for each type of PEP profile (with the exception of the connectivity profiles). The default profiles are identified as "DEFAULT" and are selected automatically whenever a component that uses the type of profile is created, unless overridden by the operator. A DEFAULT profile may not be deleted by the operator, but may be modified. A reset function may be included for each profile type which, when invoked by the operator, resets all of the profile parameters to their original default values. The reset function may be supported for use with any profile, not just a DEFAULT profile. Also, it is noted that profiles are a network management construct that are designed to make configuration easier.

[**0112**] Parameters are downloaded, as parameter files, to PEP End Points **1201**, **1207**. A separate file for each type of PEP profile used a PEP End Point **1201**, **1207** may be downloaded to some types of PEP end points. On the other hand, a single file with the concatenation of all of the profile parameters may be downloaded to other types of PEP end points. The various file configurations are shown in **FIG. 13**, below. The download can be tailored by the NMS **1213** to meet the requirements of the particular PEP end point implementation. More than one parameter file may be utilized, in which multiple profiles may be downloaded in a single file. For example, there can be up to 16 TCP spoofing parameter profiles that are required per PEP End Point **1201**, **1207**. The number of TCP spoofing parameter profiles depends, in part, on the number of selective TCP spoofing rules that are supported by a particular PEP end point. Rather than send all of these profiles as individual files, the TCP spoofing parameter profiles that are referenced by a particular TCP spoofing selection profile can be bundled with the parameters of the TCP spoofing selection profile into a single file. In addition to reducing the total number of files required, this approach eliminates the need for a potentially complicated mechanism for mapping the parameters of one file to the parameters of another. Exemplary file configurations as shown in **FIG. 13**.

[**0113**] **FIG. 13** shows a diagram of parameter files that are supported by the system of **FIG. 12**. As discussed above, the parameter profiles may be captured as a single file, individual files, or a combination thereof. In the example of **FIG. 13, a** single parameter file **1301** stores multiple profiles. Additionally, multiple files **1303** may be used, wherein one of the files stores multiple profiles. Further, the parameter profiles may each be stored in separate parameter files **1305**.

[0114] Returning to the example of **FIG. 12**, the operator configures backbone connections, matching hub (or local) site PEP End Points **1201** (e.g., IP Gateways) to remote site PEP End Points (e.g., Multimedia Relays, Multimedia VSATs and PES Remotes). By way of example, up to four connections per PEP End Point pair may be configured, at one per priority level. The hub (or local) PEP end points **1201** and remote site PEP End Points **1207** do not have to be using the same set of profiles.

[0115] The use of multiple backbone connections provides, in and of itself, a means of prioritizing spoofed TCP connections. A connectivity profile is the mechanism used by the operator to configure backbone connections. In an exemplary embodiment, connectivity profiles are associated with remote site PEP End Points **1207** with remote site PEP End Point configuration used to derive configuration information for the hub (or local) site PEP end points **1201** (e.g., IP gateways).

[0116] A connectivity profile specifies a variety of information. The connectivity profile indicates the hub (or local) site PEP End Point (e.g. IP Gateway) **1201** to which the remote site PEP End Point **1207** should establish its backbone connections. Additionally, the connectivity profile information about the hub (or local) and remote site PEP End Point IP addresses, as specified by the operator in terms of interfaces, which are to be used for the backbone connections. For example, for an IP Gateway **801**, either the enterprise interface **803** or wide area access interface **805** can be specified. The connectivity profile further provides for each priority level, an indication of whether a backbone connection should be created for this priority and, if so, what percentage of spoofing resources should be given to the connection. For instance, an assignment of 0% of the spoofing resources to a backbone connection may be designated to indicate that the connection should not be created. IP addresses are specified in terms of component interfaces in connectivity profiles to support sharing the profile. If specific IP addresses were specified, the profile could only be used with the component to which the IP address applied. However, when downloading individual parameter files to components, the NMS **1213** includes the appropriate IP addresses, not an interface indication. An interface indication may be provided, instead of an IP address, when the particular file is capable of being sent via multicast to multiple components.

[0117] A PEP End Point **1201**, **1207** has multiple IP addresses, usually one for each of its LAN interfaces and an additional IP address for its WAN interface. When a PEP backbone connection is defined, one of the IP addresses of each PEP End Point must be specified for the connection. And, the IP addresses selected must have meaning on all of the potential paths (including paths to all of the components of a redundancy group) that the PEP Backbone Protocol segments of the backbone connection might take. Specifically, the IP address used for the hub (or local) site PEP End Point **1201** must be reachable via the routing function in the remote site PEP End Point **1207**. Further, the IP address used for the remote site PEP End Point **1207** must be reachable via the routing function in the hub (or local) site PEP End Point.

[0118] Connectivity profiles define which backbone connections are created; however, the backbone connections that are used by spoofed TCP connections are determined by TCP spoofing selection profiles and the TCP spoofing parameter profiles that they reference. Additionally, if a TCP connection is mapped to a backbone connection which does not exist, it cannot be spoofed.

[0119] The NMS **1213** implements extensive cross-checking for profile and PEP End Point creation and modification scenarios to ensure that a PEP End Point **1201**, **1207** is never configured with conflicting connectivity and selective TCP spoofing criteria. In particular, the NMS **1213** performs cross checking for most scenarios; for the other scenarios, a PEP End Point **1201**, **1207** may post an event to alert the operator if a TCP connection gets mapped to a non-existent backbone connection. For example, when a connectivity profile is created or modified, the profile is cross-checked against the TCP spoofing selection profile and TCP spoofing parameter profiles that are used by the IP Gateway **1201** which the connectivity profile references. When a remote site PEP End Point **1207** is created or the connectivity profile and/or TCP spoofing selection profile being used by an existing remote site PEP End Point **1207** is changed, the connectivity profile associated with the PEP end point **1207** is cross-checked against the TCP spoofing selection profile and TCP spoofing parameter profiles it is using. When the TCP spoofing selection profile being used by a hub (or local) site PEP End Point **1201** is changed, the new TCP spoofing selection profile is be cross-checked against all connectivity profiles which reference the PEP End Point. Also, when the NMS **1213** does a cross-check, if an inconsistency is found, the NMS **1213** may not automatically reject the create or modify operation. Instead, the NMS **1213** may simply provide a warning to the operator and prompt the operator to determine whether the create or modify operation should proceed anyway. Taking this approach eliminates any "chicken and egg" issues which might otherwise have to be addressed regarding the order of changing parameters. And, the approach advantageously provides flexibility if it turns out that intentionally having mismatched parameters is useful. For example, it might be useful to remove a backbone connection from a connectivity profile as an easy way to compare spoofed versus unspoofed performance. And, removing the backbone connection may often be easier than modifying the applicable selective TCP spoofing rules.

[0120] PEP End Point profiles are used to configure parameters that only need to be configured once per PEP End Point. This includes TCP Spoofing Kernel and PEP Backbone Protocol Kernel parameters, which are not connection specific. In an exemplary embodiment, PEP End Point profiles are created by the operator and then can be assigned for use by a PEP End Point when the PEP End Point is created or modified. A PEP End Point profile cannot be deleted if any PEP End Point is currently configured to use it.

[0121] As mentioned previously, PEP End Point profiles are common to all types of PEP End Points **1201**, **1207**; for example, the same profile can be used by IP Gateways **801**, Multimedia Relays **901**, Multimedia VSATs **1001** and PES Remotes **1101**. A single PEP End Point profile can be used by all of the PEP End Points **1201**, **1207** in the system **1200**. On the other hand, the operator could create a separate PEP End Point profile for each and every PEP End Point.

[0122] To simplify configuration, a DEFAULT PEP End Point profile is utilized. The DEFAULT profile is selected automatically whenever a PEP End Point is created, unless overridden by the operator.

[0123] A TCP spoofing selection profile is used to configure selective TCP spoofing parameters (i.e. rules) for one or more PEP End Points **1201**, **1207**. TCP spoofing selection profiles are created by the operator and then can be assigned for use by a PEP End Point **1201**, **1207** when the PEP End Point **1201**, **1207** is created or modified. A TCP spoofing selection profile cannot be deleted if any PEP End Point **1201**, **1207** is currently configured to use it. TCP spoofing selection profiles are common to all types of PEP End Points. For example, the same profile can be used by IP Gateways **801**, Multimedia Relays **901**, Multimedia VSATs **1001** and PES Remotes **1101**. The NMS **1213** allows the operator to configure rule numbers along with the rules themselves. This allows the operator to change the order of rules which are already configured by simply changing their rule numbers. The NMS **1213** sorts the rules before downloading them to the components using the profile.

[0124] The definition of selection rules for selective TCP spoofing includes the ability to support AND and OR combination operators to link criteria together. An OR combination may be implemented by requiring the operator to create multiple back to back rules which map to the same selection (e.g., the same TCP spoofing parameter profile). An AND combination may be implemented by including all fields in every rule and then allowing the operator to indicate which fields apply. For example, if the operator specifies a non-zero value for a destination IP address mask and a non-zero value for a source IP address mask in a rule, the rule requires a match of both the destination and source IP address fields. The NMS **1213** allows any combination of fields in a rule to be "ANDed" together. The default value for each field in a rule should make that field a "don't care" value in order to allow the operator to just have to change the values in a rule that are a "do care". It is noted that combinations of AND and OR can be implemented with the AND part of the combination implemented within a rule and the OR part of the combination implemented by the use of multiple rules.

[0125] A TCP spoofing parameter profile is used to configure a set of TCP spoofing parameters for TCP connections, which are selected by a particular selective TCP spoofing rule. TCP spoofing parameter profiles are created by the operator and then can be assigned for use by selective TCP spoofing rules when a TCP spoofing selection profile is created or modified. TCP spoofing parameter profiles are referenced by name in the configuration of selective TCP spoofing rules. However, the names may be converted into numbers which are used to indexTCP spoofing parameter profiles in the downloaded parameters; i.e., profile names exist for the use of the operator, not the components. A TCP spoofing parameter profile cannot be deleted if any selective TCP spoofing rule is currently configured to use it. TCP spoofing parameter profiles are common to all types of PEP End Points and, thus, do not place any constraints on the use of TCP spoofing selection profiles by all types of PEP End Points.

[0126] In addition to the DEFAULT TCP spoofing parameter profile, an UNSPOOFED TCP spoofing parameter profile may always be specified. The UNSPOOFED TCP spoofing parameter profile is selected by a selective TCP spoofing rule for TCP connections which should not be spoofed. The UNSPOOFED TCP spoofing parameter profile does not actually exist as an entity (and cannot be read or modified) because none of the normal TCP spoofing parameter profile parameters has any meaning for unspoofed TCP connections. The TCP spoofing parameter profile name UNSPOOFED may simply be selected in a selective TCP spoofing rule and the NMS may map this to, for example, TCP spoofing parameter profile number **255** in the PEP End Point's configuration. PEP End Points **1201**, **1207** are aware that a selective TCP spoofing rule which indicates the use of TCP spoofing parameter profile number **255** means that the connection should not be spoofed.

[0127] PEP End Point platforms **1201**, **1207** provide status and statistics information via SNMP. Maintaining statistics on a per TCP connection basis poses a challenge because TCP connections are dynamic; therefore, statistics for individual TCP connections need not be kept unless desired.

[0128] Backbone connections, on the other hand, are more or less permanent (i.e., these connections are terminated if removed by the operator). Per backbone connection statistics are important for problem debugging; therefore, TCP and PBP statistics are kept on a per backbone connection basis. With respect to the TCP, the per backbone connection statistics represents the accumulated totals for all of the TCP connections which used and are using the backbone connection. Kernel wide statistics are also kept to track per PEP End Point statistics.

[0129] In addition to status and statistics MIB variables which can be read using the NMS **1213**, the PEP End Point platforms **1201**, **1207** also post events and alarms using SNMP traps. In general, events are only posted to provide exceptional information to the operator. Events that are informational in nature are, in general, tracked via statistics. This keeps the number of events that are being posted to a minimum (which makes them more likely to be useful); and, in the case of the remote site PEP End Points **1207**, reduces the load on the network (since, in general, bandwidth used to post events comes out of the same pool of bandwidth used to carry traffic). In some cases, some occurrences may be posted as events by the hub (or local) site PEP End Points **1201**, but be counted via statistics in remote site PEP End Points **1207** because the impact on the system **1200** is less significant at the hub (or local) site.

[0130] Events report the occurrence of an event that is worthy of note to the operator. Alarms are a special type of event that indicates an adverse condition which has duration. Because it has duration, an event is posted when the condition is encountered and a corresponding event is posted when the condition is cleared. In some cases, where it is known in advance that there is no existing way for an alarm to be cleared without restarting the PEP End Point **1201**, the clearing event may not be explicitly defined if the PEP end point implementation cannot recall the alarm condition across a platform restart. The event associated with the platform restart than acts as the alarm clearing event. The following examples illustrate the difference between a simple event and an alarm. Receiving a TCP segment with an MD5 signature option in it is a simple event because it is an instantaneous occurrence with no duration; that is, post-

ing a later event indicating the condition has cleared is meaningless. In contrast, completely running out of buffers has duration in that eventually memory might be freed, even without operator intervention. An alarm event can be posted when the condition is encountered and an alarm clearing event can be posted when the condition disappears. Therefore, this occurrence is categorized as an alarm. Alarms are assigned a severity to aid the operator. If an alarm event is defined, its alarm clearing event should also be defined (unless, as indicated above, it is known in advance that the alarm cannot be cleared without a platform restart.)

[0131] Events and alarms are posted to the NMS **1213**. However, to aid in debugging, the platform environment **1201**, **1207** may also support the ability to log events locally. By way of example, two options exist: logging to a local disk **1219**, **1221** (i.e., to a file on a local disk drive) and logging to the operator console **1217**. When one or both of these options are present, use of the options is controlled via platform specific parameters, set via the NMS or platform specific debugging tools. In general, the default value for these debugging parameters is local logging to be disabled. Special software builds may sometimes be used in the field which have local logging enabled by default. When logging to disk in the PEP end point **1201**, **1207** is enabled, all events posted by the PEP kernels, regardless of severity, are logged to disk **1219**, **1221**. Normal events (i.e., events with a severity other than LOG TO DISK or LOG TO CONSOLE) may also be sent to the NMS **1213**. If logging to disk is disabled, then an event posted with a severity of LOG TO DISK is simply ignored. When logging to the operator console is enabled, an event posted with a severity of LOG TO CONSOLE is posted to the operator console **1217**. In a particular PEP End Point platform **1201**, **1207**, the platform environment **1201**, **1207** may choose to allow the logging of other event severities (e.g., all alarms) to the operator console **1217** (in addition to their normal disposition), based on additional debugging parameters. Platform environments **1201**, **1207** may accept LOG TO DISK and LOG TO CONSOLE as valid event severities even if the only action taken for these severities is to discard (i.e., ignore) the event. Alternatively, each of the PEP end platforms **1201**, **1207** may forward the events to the NMS **1213**, which may be stored in an event log **1215**.

[0132] **FIG. 14** illustrates a computer system **1401** upon which an embodiment according to the present invention may be implemented. Such a computer system **1401** may be configured as a server to execute code that performs the PEP functions of the PEP end point **210** as earlier discussed. Computer system **1401** includes a bus **1403** or other communication mechanism for communicating information, and a processor **1405** coupled with bus **1403** for processing the information. Computer system **1401** also includes a main memory **1407**, such as a random access memory (RAM) or other dynamic storage device, coupled to bus **1403** for storing information and instructions to be executed by processor **1405**. In addition, main memory **1407** may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor **1405**. Notably, downloaded parameters may be stored in main memory **1407**. Computer system **1401** further includes a read only memory (ROM) **1409** or other static storage device coupled to bus **1403** for storing static information and instructions for processor **1405**. A storage device **1411**, such as a magnetic disk or optical disk, is provided and coupled to bus **1403** for storing information and instructions.

[0133] Computer system **1401** maybe coupled via bus **1403** to a display **1413**, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device **1415**, including alphanumeric and other keys, is coupled to bus **1403** for communicating information and command selections to processor **1405**. Another type of user input device is cursor control **1417**, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor **1405** and for controlling cursor movement on display **1413**.

[0134] Embodiments are related to the use of computer system **1401** to perform the PEP functions of the PEP end point **210**. According to one embodiment, this automatic update approach is provided by computer system **1401** in response to processor **1405** executing one or more sequences of one or more instructions contained in main memory **1407**. Such instructions may be read into main memory **1407** from another computer-readable medium, such as storage device **1411**. Execution of the sequences of instructions contained in main memory **1407** causes processor **1405** to perform the process steps described herein. One or more processors in a multi-processing arrangement may also be employed to execute the sequences of instructions contained in main memory **1407**. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions. Thus, embodiments are not limited to any specific combination of hardware circuitry and software.

[0135] The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor **1405** for execution the PEP functions of the PEP end point **210**. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device **1411**. Volatile media includes dynamic memory, such as main memory **1407**. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus **1403**. Transmission media can also take the form of acoustic or light waves, such as those generated during radio wave and infrared data communications.

[0136] Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

[0137] Various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to processor **1405** for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions relating to execution of the PEP functions of the PEP end point **210** into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system **1401** can receive the data on the telephone line and use an infrared transmitter to convert the data to an infrared signal. An infrared detector coupled to

bus **1403** can receive the data carried in the infrared signal and place the data on bus **1403**. Bus **1403** carries the data to main memory **1407**, from which processor **1405** retrieves and executes the instructions. The instructions received by main memory **1407** may optionally be stored on storage device **1411** either before or after execution by processor **1405**.

[**0138**] Computer system **1401** also includes one or more communication interfaces **1419** coupled to bus **1403**. Communication interfaces **1419** provide a two-way data communication coupling to network links **1421** and **1422**, which are connected to a local area network **14**(LAN) **1423** and a wide area network (WAN) **1424**, respectively. The WAN **1424**, according to one embodiment of the present invention, may be a satellite network. For example, communication interface **1419** may be a network interface card to attach to any packet switched LAN. As another example, communication interface **1419** may be an asymmetrical digital subscriber line (ADSL) card, an integrated services digital network (ISDN) card, a cable modem, or a modem to provide a data communication connection to a corresponding type of telephone line. Wireless links may also be implemented. In any such implementation, communication interface **1419** sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

[**0139**] Network link **1421** typically provides data communication through one or more networks to other data devices. For example, network link **1421** may provide a connection through local area network **1423** to a host computer **1425** or to data equipment operated by an Internet Service Provider (ISP) **1427**. ISP **1427** in turn provides data communication services through the Internet **505**. In addition, LAN **1423** is linked to an intranet **1429**. The intranet **1429**, LAN **1423** and Internet **505** all use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link **1421** and through communication interface **1419**, which carry the digital data to and from computer system **1401**, are exemplary forms of carrier waves transporting the information.

[**0140**] Computer system **1401** can send messages and receive data, including program code, through the network(s), network link **1421** and communication interface **1419**. In the Internet example, a server **1431** might transmit a requested code for an application program through Internet **505**, ISP **1427**, LAN **1423** and communication interface **1419**. The received code may be executed by processor **1405** as it is received, and/or stored in storage device **1411**, or other non-volatile storage for later execution. In this manner, computer system **1401** may obtain application code in the form of a carrier wave. Computer system **1401** can transmit notifications and receive data, including program code, through the network(s), network link **1421** and communication interface **1419**.

[**0141**] The techniques described herein provide several advantages over prior approaches to improving network performance, particularly in a packet switched network such as the Internet. A local PEP end pint and a remote end point communicate to optimize the exchange of data through a TCP spoofing functionality. A network management system provides ease of configuration of the end points through the use of profiles.

[**0142**] Obviously, numerous modifications and variations of the present invention are possible in light of the above teachings. It is therefore to be understood that within the scope of the appended claims, the invention may be practiced otherwise than as specifically described herein.

What is claimed is:

1. A method for monitoring a communication system that includes a platform configured to perform a plurality of performance enhancing functions, the method comprising:

receiving information relating to configuration parameters as specified in a profile of the platform;

selectively modifying the profile in response to the received information; and

forwarding the modified profile to the platform.

2. The method according to claim 1, wherein the modified profile is forwarded as a single file.

3. The method according to claim 1, wherein the communication system is partitioned into a plurality of network management domains to control access network management information.

4. The method according to claim 1, further comprising:

maintaining a default profile for the platform.

5. The method according to claim 1, wherein the profile in the receiving step includes at least one of a TCP spoofing kernel parameter, a backbone protocol kernel parameter, a prioritization kernel parameter, and a path selection parameter.

6. The method according to claim 1, further comprising:

selectively storing the information at least within the platform and within a database that is separate from the platform.

7. A communication system comprising:

a platform configured to provide performance enhancing functions, the platform having a profile that specifies configuration parameters; and

a network management system communicating with the platform, the network management system being configured to receive information relating to the configuration parameters as specified in the profile, wherein the network management system is configured to selectively modify the profile in response to the received information and to forward the modified profile to the platform.

8. The system according to claim 7, wherein the modified profile is received as a single file by the network management system.

9. The system according to claim 7, wherein the communication system is partitioned into a plurality of network management domains to control access network management information.

10. The system according to claim 7, wherein the network management system maintains a default profile of the platform.

11. The system according to claim 7, wherein the profile includes at least one of a TCP spoofing kernel parameter, a backbone protocol kernel parameter, a prioritization kernel parameter, and a path selection parameter.

12. The system according to claim 7, wherein the platform includes a local disk configured to storing the information from the management agent.

13. The system according to claim 7, wherein the network management system includes a database configured to storing the information from the management agent.

14. A network apparatus for monitoring a communication system that includes a platform configured to perform a plurality of performance enhancing functions, the apparatus comprising:

means for receiving information relating to configuration parameters as specified in a profile of the platform;

means for selectively modifying the profile in response to the received information; and

means for forwarding the modified profile to the platform.

15. The system according to claim 14, wherein the modified profile is forwarded as a single file.

16. The system according to claim 14, wherein the communication system is partitioned into a plurality of network management domains to control access network management information.

17. The system according to claim 14, further comprising:

means for maintaining a default profile for the platform.

18. The system according to claim 14, wherein the profile includes at least one of a TCP spoofing kernel parameter, a backbone protocol kernel parameter, a prioritization kernel parameter, and a path selection parameter.

19. The system according to claim 14, further comprising:

means for selectively storing the information at least within the platform and within a database that is separate from the platform.

20. A computer-readable medium carrying one or more sequences of one or more instructions for monitoring a communication system that includes a platform configured to perform a plurality of performance enhancing functions, the one or more sequences of one or more instructions including instructions which, when executed by one or more processors, cause the one or more processors to perform the steps of:

receiving information relating to configuration parameters as specified in a profile of the platform;

selectively modifying the profile in response to the received information; and

forwarding the modified profile to the platform.

21. The computer-readable medium according to claim 20, wherein the modified profile is forwarded as a single file.

22. The computer-readable medium according to claim 20, wherein the communication system is partitioned into a plurality of network management domains to control access network management information.

23. The computer-readable medium according to claim 20, wherein the one or more processors further perform the step of:

maintaining a default profile for the platform.

24. The computer-readable medium according to claim 20, wherein the profile in the receiving step includes at least one of a TCP spoofing kernel parameter, a backbone protocol kernel parameter, a prioritization kernel parameter, and a path selection parameter.

25. The computer-readable medium according to claim 20, wherein the one or more processors further perform the step of:

selectively storing the information at least within the platform and within a database that is separate from the platform.

* * * * *