US 20030131079A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2003/0131079 A1

Neale et al. (43) Pub. Date: Jul. 10, 2003

(54) PERFORMANCE ENHANCING PROXY TECHNIQUES FOR INTERNET PROTOCOL TRAFFIC

(75) Inventors: **Jason Neale**, Beaconsfield (CA); **Andrew M. Pether**, Pincourt (CA); **Abdul-Kader Mohsen**, Montreal (CA); **Guy Begin**, Verdun (CA)

Correspondence Address:
HOGAN & HARTSON LLP
IP GROUP, COLUMBIA SQUARE
555 THIRTEENTH STREET, N.W.
WASHINGTON, DC 20004 (US)

(73) Assignee: **EMS Technologies, Inc.**

(21) Appl. No.: **10/292,899**

(22) Filed: **Nov. 13, 2002**

Related U.S. Application Data

(60) Provisional application No. 60/333,608, filed on Nov. 13, 2001.

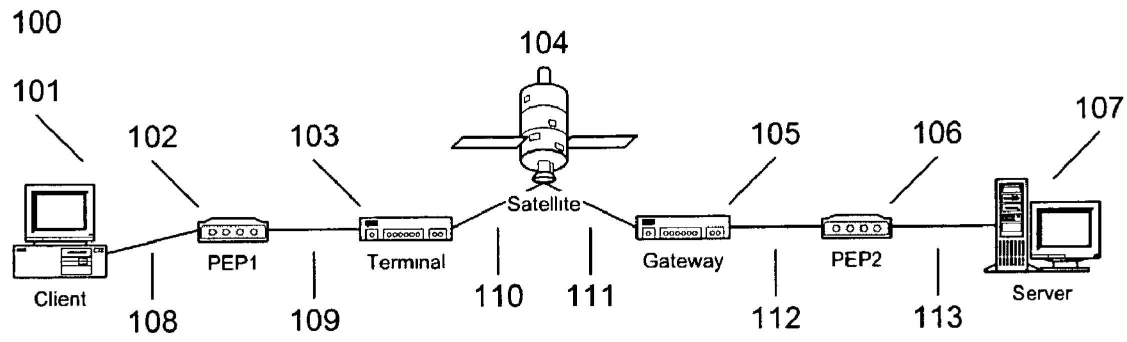
Publication Classification

(51) Int. Cl.^{7} **G06F 15/177**

(52) U.S. Cl. **709/220**

(57) **ABSTRACT**

Methods and systems are described that may be used in the context of a Performance Enhancing Proxy architecture for Internet traffic. A method of accelerated connection opening with error handling is disclosed. Also, a method of handling the Path MTU Discovery mechanism in the context of a distributed connection splitting PEP is described. A novel packet format for a proprietary inter-PEP protocol is described which allows for low packet header overhead. A new acknowledgement scheme that adapts to packet loss conditions to minimize bandwidth consumption by selecting an acknowledgement type from several possibilities is also detailed. A method whereby potentially spurious retransmissions are minimized by timing every transmission and retransmission is also described.



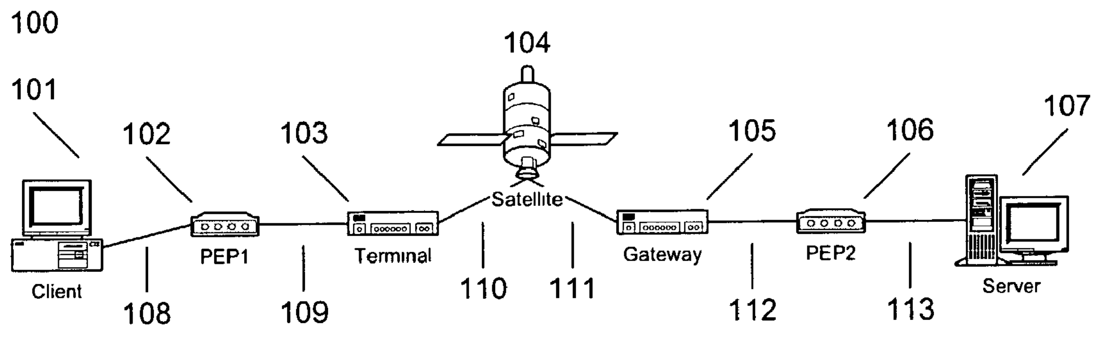
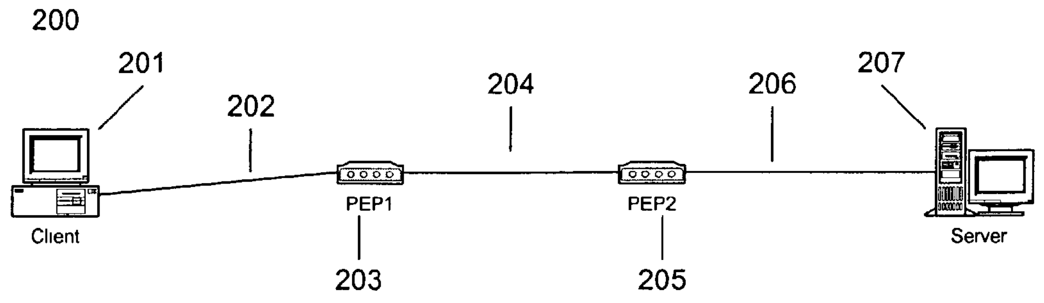

100
101
102
103
104
105
106
107
Client
PEP1
Terminal
Satellite
Gateway
PEP2
Server
108
109
110
111
112
113

Figure 1

Figure 2



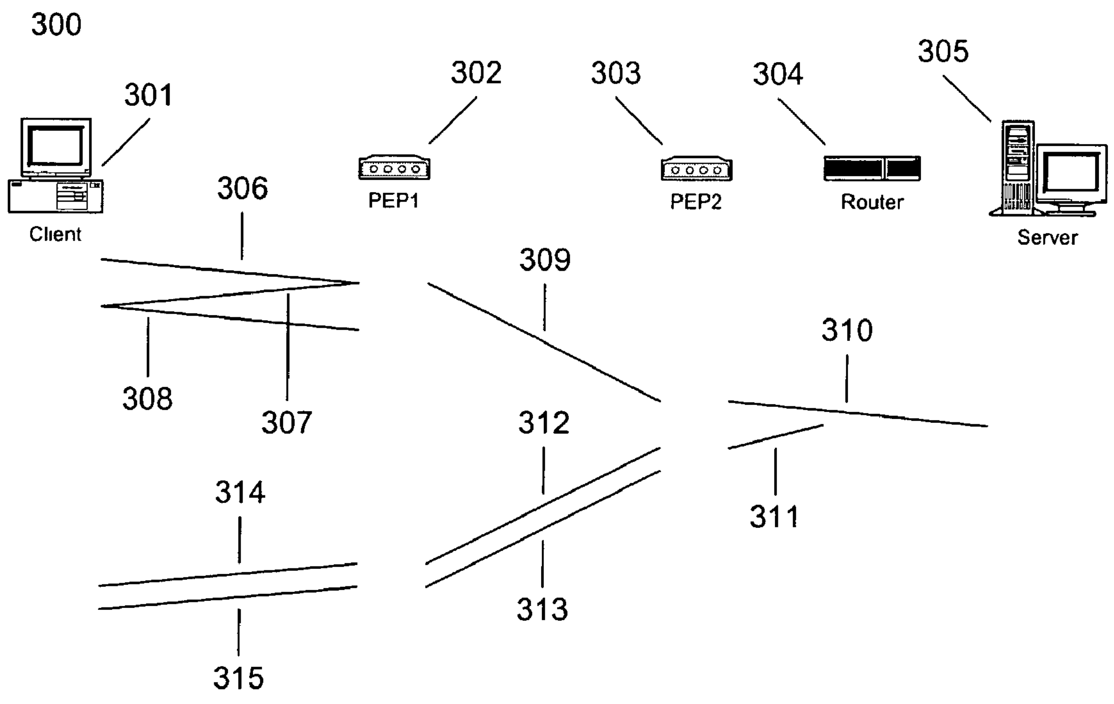

300
301
302
303
304
305
Client
PEP1
PEP2
Router
Server
306
307
308
309
310
311
312
313
314
315

FIGURE 3

Figure 4

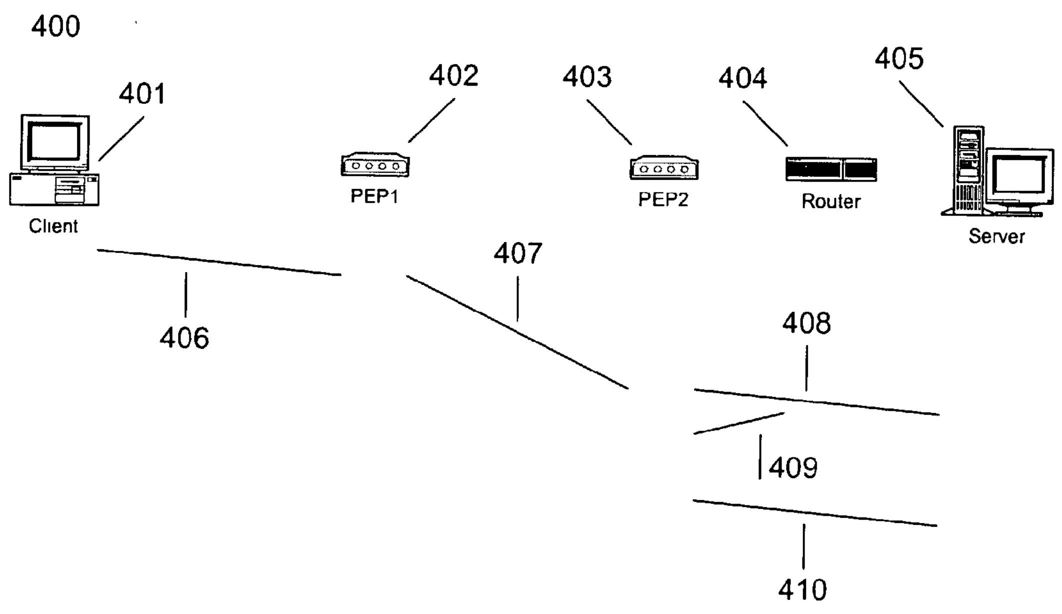
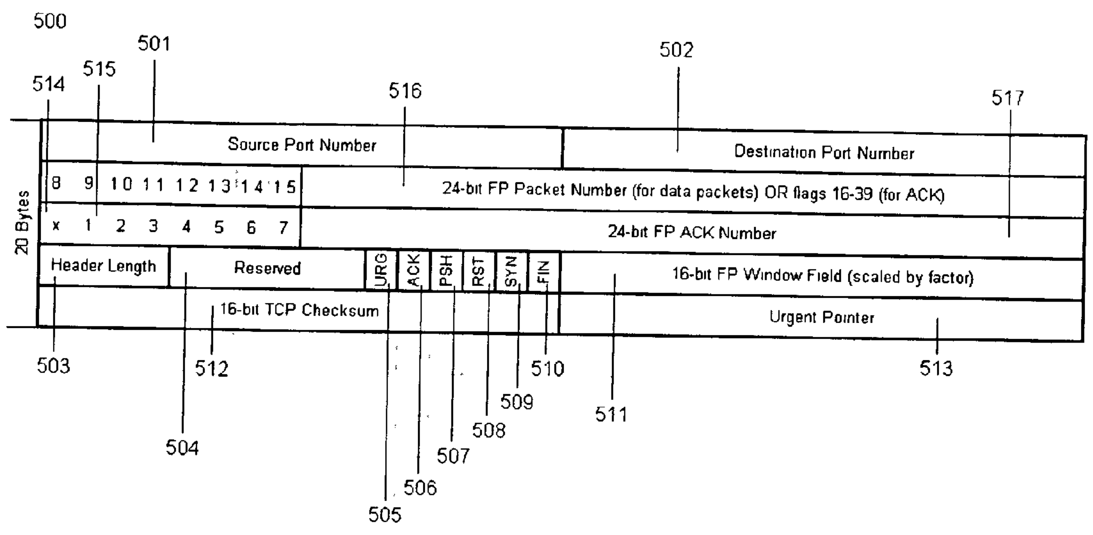

400
401
402
403
404
405
Client
PEP1
PEP2
Router
Server
406
407
408
409
410

Figure 5


500
501
502
514
515
516
517
Source Port Number
Destination Port Number
8 9 10 11 12 13 14 15
24-bit FP Packet Number (for data packets) OR flags 16-39 (for ACK)
x 1 2 3 4 5 6 7
24-bit FP ACK Number
20 Bytes
Header Length
Reserved
URG
ACK
PSH
RST
SYN
FIN
16-bit FP Window Field (scaled by factor)
16-bit TCP Checksum
Urgent Pointer
503
512
504
505
506
507
508
509
510
511
513

PERFORMANCE ENHANCING PROXY TECHNIQUES FOR INTERNET PROTOCOL TRAFFIC

CROSS-REFERENCE TO RELATED APPLICTIONS

[0001] This application claims priority to U.S. Provisional Application No. 60/333,608 to Jason D. Neale et. al., entitled "Performance Enhancing Proxies for Satellite Transmission Control Protocols," filed on Nov. 13, 2001.

BACKGROUND OF THE INVENTION

[0002] 1. Field of the Invention

[0003] The invention relates to the field of multiplex communications techniques. More particularly, the invention provides methods and systems for improving the performance, efficiency and user experience of systems transporting Internet Protocol (IP) traffic, by the use of Performance Enhancing Proxies (PEPs).

[0004] 2. Background of the Invention

[0005] The Internet is a world-wide computer super-network, which is made up of a large number of component networks and their interconnections. Computer networks may consist of a wide variety of connected paths or network links serving to transport user information in the form of data between a diverse array of computer end systems. Different network links are more or less suitable for different network requirements. For example, a fiber optic cable typically provides a high bandwidth, low per bit cost, low error rate and low delay point-to-point network link. Alternatively, for example, a satellite link typically provides a lower bandwidth, higher per bit cost, higher error rate and longer delay point-to-multi-point network link. The wide variety of links and thus link characteristics encountered on the Internet, or other private (IP) based networks, have a variety of effects on the behavior of protocols in the IP suite.

[0006] IP primarily provides the routing functionality for packets (bits or bytes of data) over a network. It acts at the network layer to direct packets from their sources to their destinations. Transmission Control Protocol (TCP) is the reliable transport layer protocol of the IP suite of protocols and as such, layers on top of IP, provides reliability to applications, and builds on IP's unreliable datagram (packet) service. TCP underlies the vast majority, estimated to be around 90%, of all the traffic on the Internet. TCP supports the World Wide Web (WWW), electronic mail (email) and file transfers, along with other common applications. TCP was introduced in 1981 and since then has evolved in many ways, but today still provides reliable and largely efficient service over a wide variety of links as evidenced by its omnipresent nature. However, there are a variety of conditions under which TCP may perform below expectations, geosynchronous satellite links being one prime example. The problems of TCP over satellites has been previously documented. TCP performance is typically degraded to some extent in terms of lowered throughput and link utilization by, but not limited to, the following link characteristics: long delay, high bandwidth, high error rate, link asymmetry and link variability, all of which may be encountered on satellite and similar links.

[0007] In response to the established use of TCP and also of certain link types (such as satellite) which are not ideal for TCP, Performance Enhancing Proxies (PEPs) have been introduced. PEPs may function as one or more devices or pieces of software placed in the end-to-end path that suffers TCP performance degradation. PEP units may, for example, surround a satellite link. PEPs modify the traffic flow in an attempt to alleviate the issues of TCP traffic on a specific link. PEPs may use many methods, either alone or in concert, to enhance performance.

[0008] One type of PEP, known as a distributed, connection splitting PEP is commonly chosen due to that fact that it allows for the use of a proprietary protocol across the satellite link. This protocol can then be chosen or designed to mitigate problems specific to the link. A distributed PEP uses more than one PEP device in an end-to-end connection, often two PEP devices are used. If two PEP devices are used, the end-to-end connection may be split into 3 connection segments. The end connections must remain TCP for compatibility, but the inter-PEP connection may be any protocol. Several protocols are available for use on the satellite link that provide improved performance over that of TCP. Examples of these protocols are Xpress Transport Protocol (XTP), Satellite Transport Protocol (STP), Space Communications Protocol Standards—Transport Protocol (SCPS-TP), standard or enhanced User Datagram Protocol (UDP) or even non-standard (modified) TCP. In addition to the protocol used, there are also many ways in which a PEP device may handle processing between connection segments in this type of system.

[0009] One of the link characteristics that affects TCP performance is delay. Links such as those over GEO satellites have long delays, for example, around 500 ms or more. Several TCP mechanisms that control connection setup, flow control and error correction through retransmission may be adversely affected by long delay links.

[0010] For transfers that are typically short in duration, such as web pages, the delays involved in establishing TCP connections make a proportionally larger contribution to the transfer time, and therefore to the mean data throughput rate. Additionally, a user typically begins to view a web page as it is downloading so an initial delay before any material is displayed may frustrate a user and also consequently, potentially cause re-requests which lower system efficiency.

[0011] The delay in connection opening is caused by a mechanism known as the TCP Three Way Handshake (3WHS). The purpose of this exchange of messages is to ensure that the other end point is present, and thereby to promote the reliability of a transfer. A connection initiator sends a packet with the SYN (synchronize) flag set. A responding system sends back a packet with the SYN and ACK (acknowledgement) flags set. The ACK flag acknowledges the initiator's SYN. The initiator then sends a final ACK packet acknowledging the responder's SYN. From this point on the initiator may send data. Thus, the delay from initiation to sending data on a TCP connection is a whole RTT [Note to inventor—please define RTT].

[0012] When opening a TCP connection in a distributed split-connection PEP implementation, there are two main options and then variants thereof. For preserving end-to-end behavior of the connection and reliability, the connection should be opened end-to-end and the connection should be opened by the endpoints and not the PEP devices. Although more reliable than alternatives, however, this method involves a full RTT of overhead during which no data is transferred. An alternative method involves accelerating the opening of certain connections, such as web connections, which are of short duration and thus more heavily affected by extra RTTs.

[**0013**] An initiator sends a SYN packet to a PEP and the PEP responds locally with the SYN/ACK packet to the initiator. The initiator then responds with the ACK packet and the first data packet, which in the case of a web transfer is an HTTP request packet. The PEP then combines the original SYN packet (which it has held) and the first data packet and sends them over the satellite link to the other PEP device. The lower RTT on the terrestrial link means that the time taken to send the first request is reduced.

[**0014**] A problem with the above accelerated opening is that it is possible to open a connection locally that might then fail to establish end-to-end, resulting in a desynchronized state. This state will eventually time-out. However, during the time that the two endpoints are desynchronized, the user will be confused, as the connection appears to be established but no data will be transferred, which again could lead to the user re-attempting the connection several times and wasting bandwidth.

[**0015**] As described earlier, the Internet is a collection of networks and interconnections. These interconnections and network links each have their own characteristics. One characteristic is the Maximum Transmission Unit (MTU) size. This value, often expressed in bytes, is the maximum data payload that may be encapsulated and carried over the OSI/ISO 7-layer model link layer without being broken down into a smaller unit. Two common technologies for LAN links are Ethernet and the similar, but not identical, IEEE 802.3 standards. Ethernet allows for the encapsulation of a 1500 byte IP packet (1500 byte MTU) while 802.3 encapsulation allows for a 1492 byte MTU. It can be imagined that in a network of heterogeneous links there will, sometimes, not be one common MTU for any path between points A and B in a given network or path through the Internet.

[**0016**] In response to the recognition that any given path through a network may not have a consistent MTU for all hops, the IP protocol allows for fragmentation of IP packets. If the IP layer at a host or router is unable to send a packet of the desired size onto the link, the IP layer will split that packet up into several smaller packets. When this behavior occurs at a router between ports, it is known as fragmentation and is commonly recognized to have detrimental side effects, such as lowering maximum data rate (through additional header bytes and also packet processing overhead at network nodes) and impacting efficiency. However, fragmentation is necessary to allow the data to pass end-to-end.

[**0017**] In an attempt to avoid fragmentation, the process of Path MTU Discovery (PMTUD) was introduced. The purpose of this process is to try to detect the minimum MTU in the path from source to destination. This value is dynamic if the route changes. The IP header has a flag, which may be set to inform intermediate network nodes (i.e. any devices in the network between the source and destination) not to fragment a packet. This flag is known as the Don't Fragment (DF) flag. When the DF flag is set, a router should discard the packet if it is too large to forward on the outgoing interface. The router should also send an Internet Control Message Protocol (ICMP) Can't Fragment (ICMP type **3**[destination unreachable], code **4**[fragmentation needed but don't-fragment bit set]) message back to the originator of the packet. This packet should contain the MTU of the outgoing interface on the router to inform the sender of the limiting MTU. Through this mechanism, a sender may adapt to the path MTU and avoid fragmentation. This mechanism is therefore desirable for efficiency reasons.

[**0018**] Currently, there is little guidance on how PMTUD should function in the presence of PEPs. In the absence of guidance, it is currently left to the decision of each PEP designer or manufacturer on how to handle the PMTUD mechanism at a PEP. One solution requires that ICMP messages pass through its PEP devices without modification. This allows for the sender to adapt its path MTU estimate and send smaller packets in the future.

[**0019**] However, a problem exists in a connection-splitting distributed PEP, due to the fact that the PEP devices are often buffering packets that are in transit between the endpoints. These packets have been acknowledged to the sending endpoint and are, therefore, no longer buffered by the endpoint itself for retransmission. Therefore, if a router drops a packet after the second PEP in the connection and an ICMP Can't Fragment message is sent to the originator, a problem occurs. The originator is able to lower the Path MTU estimate but cannot retransmit the data in the original packet. The second PEP in the connection has a copy of the packet buffered so may retransmit when no TCP acknowledgement arrives, but will not understand that the packet must be resized to a smaller packet to arrive successfully at the destination. Therefore, a deadlock may occur until several retransmissions of the packet have failed and the connection has to be reset.

[**0020**] One solution to this problem is that PMTUD may be disabled when a PEP is included in the end-to-end connection to allow the connections using the PEP to function correctly. This however is not ideal for the reasons stated above. Hence, problems exist in the current technique for PMTUD when PEPs are used.

[**0021**] Each protocol used on the Internet has its own packet format, which specifies the way that information is encoded in headers and where data begins in a packet, among other things. The TCP packet format includes the TCP header and space in the header for optional fields known as TCP options. Distributed connection splitting PEPs may use other (non-TCP) standard protocols and possibly proprietary protocols between the two PEP devices. These non-TCP protocols are used to gain performance advantages over end-to-end TCP and even split connection TCP, performance however is only one, although the most important, aspect of a PEP. A PEP must also be compatible with the end hosts and the TCP protocol. If the PEP to PEP protocol does not support the transfer of certain TCP information from end-to-end then functionality will be lost; the TCP urgent pointer which is used to expedite transfer of portions of the data stream being one example.

[**0022**] When choosing or designing a protocol for the problematic link there is, therefore, a tradeoff between efficiency and compatibility. If using an entirely different protocol, it may be necessary to carry the TCP information in extra header structures, which may increase the packet overhead on each packet. Increasing packet overhead may also trigger IP fragmentation for packets that were originally the maximum size for the link; this should be avoided. Also, the end-to-end path over which the connection travels may have intermediate equipment that does not know how to handle unknown protocols. For example, Network Address Translation (NAT) devices may perform translation of the IP address fields and sometimes layer 4 protocol port numbers also. These types of operations can then require the checksum fields to be updated. If a protocol is not recognized, it may not be able to function properly at, for example, the NAT device or packets may pass the NAT device but be unrecognizable at the receiver. Additionally, the functionality of a newly designed protocol will impose constraints on the information that must be carried in each packet. For the proprietary protocol chosen for use with the PEP design of this invention, no pre-existing packet structure was considered appropriate.

[**0023**] For problematic links, TCP has been improved by several different mechanisms to address different issues. For the case of packet and acknowledgement loss, TCP has been improved by the addition of the Selective Acknowledgement (SACK) option. This allows TCP packet headers to carry information on contiguous blocks of packets that have been successfully received. This mechanism adds overhead to each packet and although the overhead is only a small percentage on large packets (around 1% on a 1500 byte packet), the percentage overhead on a standard acknowledgement packet is much larger. For a 40-byte packet, an extra 12 to 20 bytes of SACK information is between an extra 30 and 50% of the original packet size. More seriously, if the TCP acknowledgements are carried over a link layer protocol such as Asynchronous Transfer Mode (ATM), a TCP acknowledgement with SACK information may no longer fit within a single ATM cell. If, instead, two cells are required for the acknowledgement then acknowledgement traffic volume is, in effect, doubled. If, for example, this is the return channel on a satellite system such as the Digitial Video Broadcast-Return Channel Satellite (DVB-RCS) where most traffic may be acknowledgement traffic, then the total traffic volume may also be nearly doubled.

[**0024**] TCP also uses a cumulative acknowledgement scheme to signal correct reception of packets to the sender. Optionally, TCP may use the SACK option described earlier if higher packet loss rates are expected, as may often be the case over satellite links, for example. Whether standard TCP acknowledgements are used or whether the SACK option is used, the same method of acknowledgement must be used throughout the duration of the connection. If the error conditions on the link change during the course of the transfer, the connection performance may be adversely impacted if an inappropriate acknowledgement method is chosen. For example, if the standard TCP acknowledgement scheme is selected, the TCP transfer may suffer very poor performance or even failure under heavy error conditions. If the SACK scheme is chosen, the additional overhead, as described above, may be incurred even if the SACK scheme is not needed. TCP is unable to adapt the acknowledgement scheme to changing error conditions during the course of the connection. This problem exists with the conventional systems in the area of acknowledgement of packets.

[**0025**] TCP also uses a timer as one method of detecting lost packets and triggering retransmissions. However, in the conventional systems, only one timer is used regardless of how many packets are being sent. TCP uses the timer in the following manner. When there are no packets in transit, the timer is off. When the first packet is transmitted, the timer is set. When a packet is acknowledged and other packets are still in transit, the timer is reset. Therefore it may take different amounts of time to detect a packet loss depending upon which packet in a group is lost. In the worst case it may take up to the timer timeout value plus the round trip time to detect a loss. This time period may be almost twice as long as the detection period for loss of the first packet. In the ideal case, every loss should be detected as quickly as possible.

[**0026**] Additionally, and perhaps more importantly, if an acknowledgement scheme is used in which repeated retransmission triggers occur for the same packet, the single packet timer provides no indication of how long an individual packet has been in transit. This means that it is not possible to know if a transmitted packet has had time to be acknowledged or not. In this case, it is possible to retransmit a packet before it has bad time to reach the destination and an acknowledgement be returned and received. This scenario lowers the efficiency of the link as packets are transmitted multiple times unnecessarily. This is a problem in the conventional systems related to controlling or limiting unnecessary retransmissions.

SUMMARY OF THE INVENTION

[**0027**] The invention provides solutions to the problems of the conventional systems as described above in the following areas: connection opening, path MTU discovery, satellite protocol packet format, acknowledgement of satellite protocol packets and unnecessary re-transmissions. The invention provides systems and methods for connection opening in a distributed split-connection PEP implementation. The systems and methods in accordance with the invention use an accelerated opening that allows for minimizing the time spent in the desynchronized state to avoid unnecessary re-request transfers in the case where the connection does not establish properly. When a connection attempt fails, the current invention intercepts the commonly resulting ICMP messages at the PEP, and handles them to provide a faster tear down of the failed connection. This minimizes the time the endpoints spend in a desynchronized state and reduces the period in which a user may become frustrated and generate multiple re-requests. Thus, one aspect of the invention is to reduce unnecessary request transmissions, for example for web pages, and to improve the user experience by avoiding long idle or dead periods which cause the connection to seem to have stopped responding.

[**0028**] The invention also provides a solution to the problems regarding the use of the PMTUD mechanisms in the presence of PEPs, specifically where the PEPs locally acknowledge packets and buffer them for retransmission. The invention offers a system and method where the PEPs involved in an end to end connection filter for ICMP “Can’t Fragment” messages for the PEP enhanced TCP connections “If Can’t Fragment” messages are received then the path MTU estimate at the PEP is adjusted and the data contained in the dropped packet is locally retransmitted in multiple subsequent packets. This avoids a potential deadlock that may prove fatal to affected connections. Therefore, a further aspect of the invention is to provide a reduction of unnecessary processing and retransmissions from the PEP devices.

[0029] The invention also provides a novel packet structure to support the unique needs and functionality of a proprietary protocol designed to mitigate certain detrimental link characteristics. The packet structure in accordance with the invention is compact, being the same size as the minimal TCP header, which also means that fragmentation is avoided between the PEP devices. The inventive packet structure supports some of the TCP functionality and also new features of the proprietary protocol. The protocol uses the TCP protocol type in the IP header so that it will be treated like TCP by intermediate equipment such as routers and NATs.

[0030] The port numbers are not altered, as they are used by both TCP and the proprietary protocol for connection identification. The port numbers may be modified by a NAT device if necessary, as they are located at the same place in the packet structure. The TCP checksum is also used, again for compatibility, primarily with NAT devices. Sequence and acknowledgement numbers are modified to allow for behaviour different to TCP. The TCP flags are shared by both the TCP end connections and the proprietary protocol. The TCP reserved field is maintained for compatibility with future uses. The TCP window size field is reused for communicating a satellite protocol PEP-to-PEP flow control window. The urgent pointer is maintained.

[0031] A packet number field which is 24-bits in length replaces the TCP sequence number field. An acknowledgement number field is also included which is also 24-bits in length. The additional spared bits from the packet and acknowledgement fields are used for identifying acknowledgement type, and also as bit flags to represent packets being acknowledged. Each bit flag indicates the presence or absence of a packet at the receiver. Depending on the value of the TCP acknowledgement flag and the acknowledgement type flag, different acknowledgement formats may be used. The acknowledgement scheme also allows for sending multiple acknowledgements in one packet, without the need for a larger packet header, through the use of options.

[0032] The invention also offers a system and method whereby a proprietary protocol may use several different types of acknowledgement packets. The packet types allow for positively, and possibly negatively, acknowledging a different number and pattern of received and possibly missing packets. Additionally, a method for choosing the most suitable acknowledgement to use each time an acknowledgement is sent is described, with the intention of reducing the volume of bytes transmitted to acknowledge a pattern of received and lost or errored packets. This scheme allows for the possibility of dynamically adapting to packet loss conditions to lower the volume of acknowledgement bytes otherwise necessary. The scheme also has lower overhead than schemes in protocols such as TCP. The inventive scheme may also employ an acknowledgement timer to ensure a minimum rate of acknowledgements and an upper bound to the real RTT.

[0033] The invention also allows for the timing of every packet in transit. Each time a packet is transmitted, a copy is buffered to allow for retransmission if necessary. Each buffered copy of a packet has, stored with it, a timestamp that records the time of transmission. Alternatively, the buffered copy of the packet may have stored with it a timestamp indicating the expected time of acknowledgement. In the first scheme, the stored timestamp may be compared against the current time to see if the time difference is greater than the expected roundtrip time, including all delays. In the second scheme, the current time is compared against the stored time to see if an acknowledgement for the packet was expected by this time. Both methods, therefore, can be used to prevent re-transmitting a packet if a copy of the packet is already in transit on the link or an acknowledgement is in transit in the return direction. The units of time used may be real or pseudo time units. When packets are retransmitted the timestamps are updated.

BRIEF DESCRIPTION OF THE DRAWINGS

[0034] The accompanying drawings, which are included to provide a further understanding of the invention and are incorporated in and constitute a part of this specification, illustrate embodiments of the invention. Together with the written description, these drawings serve to explain the principles of the invention. In the drawings:

[0035] **FIG. 1** shows a block diagram of an exemplary PEP deployment with the connections and equipment involved;

[0036] **FIG. 2** shows a split-connection, distributed PEP implementation in accordance with an embodiment of the invention;

[0037] **FIG. 3** illustrates the system for reducing desynchronised time by handling ICMP messages at the PEPs in accordance with an embodiment of the invention;

[0038] **FIG. 4** shows PEP intervention in the Path MTU Discovery mechanism in accordance with an embodiment of the invention; and

[0039] **FIG. 5** shows the proprietary protocol packet format in accordance with an embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

[0040] This application incorporates by reference the disclosures of provisional patent application No. 60/333,608, entitled "Performance Enhancing Proxies For Satellite Transmission Control" from which this application claims priority, an application entitled "Flow Control Between Performance Enhancing Proxies Over Variable Bandwidth Split Links" and an application entitled "Performance Enhancing Proxies" both of which are concurrently filed herewith on Nov. 13, 2002 and have common ownership.

[0041] **FIG. 1** illustrates a simplified view of a system **100** that includes equipment and links involved in a PEP deployment in a satellite environment in accordance with an embodiment of the invention. It is important to note that the use of a satellite is merely illustrative of one embodiment of the invention and that the invention is applicable to both terrestrial hard-wired and terrestrial wireless applications. In **FIG. 1, a** client **101** will make a connection attempt to server a **107** via a satellite **104**. The client **101** is connected by a LAN segment **108** to a first or terminal side PEP**1102** and the PEP**1102** is connected by another LAN segment **109** to a satellite terminal **103** (or alternatively a satellite modem of some form). Traffic from the terminal **103** passes over communications links **110** and **111** via the satellite **104** to the gateway **105** (alternatively central hub equipment or another satellite modem). Traffic leaving the gateway passes via

LAN segment **112** to a gateway side or second PEP2**106**. This second or gateway side PEP2**106** then sends the traffic via a WAN such as part of the Internet **113** to a server **107**. The traffic may be a client request, which could generate server response traffic in the reverse direction. It should also be recognized that the terminal side or first PEP1**102** could be combined with terminal device **103**.

[**0042**] **FIG. 2** shows a split connection PEP implementation **200**. In **FIG. 3, a** client **201** connects to a first PEP1**203** via a first TCP connection **202**. The PEP1**203** then connects to a second PEP2**205** via a proprietary protocol connection **204**. The PEP2**205** then connects to a server **207** via a second TCP connection **206**.

[**0043**] **FIG. 3** shows a system of handling ICMP messages **300** to reduce desynchronized time in the event of the failure of the accelerated connection establishment method in accordance with an embodiment of the invention. In **FIG. 3, a** client **301** attempts to establish a TCP connection to a server **305** via a distributed PEP implementation that includes a first PEP1**302** and a second PEP2**303**. Other network equipment may be present such as a router **304** between the second PEP2**303** and the server **305**. The client **301** initiates the request with a TCP SYN segment **306** addressed to the server. The PEP1**302** intercepts this segment and replies to the client **301** with the standard TCP SYN/ACK segment **307**. The original TCP SYN segment is now converted to a connection-opening packet for the satellite protocol **309** and sent over the satellite link. While the satellite protocol is opening its connection, the TCP 3WHS completes with the sending of a TCP ACK segment **308** addressed to the server and intercepted by the PEP1**302**. At this point, the local TCP connection from the client **301** to the PEP1**302** is now fully open and the client **301** perceives that the connection has been established to the server **305**.

[**0044**] The process continues when the proprietary protocol connection-opening packet arrives at the PEP2**303**. The PEP2**303** then converts back to a TCP SYN segment **310** and attempts to open a second TCP connection, this time from the PEP2**303** to the server **305**. After sending the second TCP SYN to the server, the PEP2**303** monitors for ICMP message **311** related to that TCP connection. At this point a variety of responses may be obtained. If the connection sets up successfully, none will be received. If the connection fails, an ICMP message may be received from a router **304**, or from the server **305** itself if, for example, the TCP protocol is not implemented.

[**0045**] The process then concludes by the PEP2**303** forwarding the received ICMP message **312** to the PEP1**302**. The PEP1**302** then forwards the ICMP message **314** once more, to the client **301** for informational and diagnostic purposes and its receipt may also trigger connection teardown depending upon the implementation of the ICMP protocol at the client **301**. When the PEP2**303** detects an ICMP error message **311** for one of its connections, it may also generate a satellite protocol reset packet **313** to send over the satellite link, following the ICMP message. This packet will close down the satellite protocol connection and then be converted to a TCP reset packet **315** to guarantee teardown of the desynchronised TCP connection. After receiving the ICMP message, the PEP2**303** will close both the TCP and satellite protocol connections.

[**0046**] Variations of the above mechanism include the PEP2**303** only forwarding the ICMP packet or only a reset packet to reduce mechanism overhead bytes on the inter-PEP link, but at the expense of information as to the cause of the error or certainty as to the reaction of the client **301**. Alternatively, the PEP2**303** may forward the ICMP packet to the PEP1**302** and the PEP1**302** may also detect ICMP messages from the satellite link direction that are related to its connections. Then the PEP1**302** may generate a TCP reset locally and send it to the client **301**. In this last case, the PEP2**303** would close the second TCP connection and the related satellite connection and the PEP1**302** would use the ICMP message passing through to close the first TCP connection. Should the ICMP message be lost, the client **301** will not receive the diagnostic information it contains, but the reset packets will ensure all points close the connection segments end-to-end. Should one or all reset packet(s) be lost, mechanisms in the PEPs and endpoints will detect this and teardown the connections end-to-end.

[**0047**] **FIG. 4** depicts a mechanism **400** by which the invention interacts with a PMTUD mechanism in accordance with an embodiment of the invention. In **FIG. 4, a** client **401** is attempting to transfer data to a server **405** through two PEPs, a first PEP1**402** and a second PEP2**403** in a path that includes a router **404** with differing input interface and output interface MTUs. The client **401** is using the PMTUD mechanism so each IP header has the DF bit set to avoid fragmentation and provide ICMP feedback from intermediate devices. The client **401** sends packet **406** to the server **405**. The packet **406** is intercepted by the PEP1**402** and converted to a satellite protocol packet **407** which is received by the PEP2**403** and converted back to a TCP packet **408**, which is sent by the PEP2**403** and intended for the server **405**. The router **404**, however, cannot forward the packet because it is not allowed to fragment it, soothe router **404** drops the packet and sends back an ICMP "Can't fragment" message **409** to the client. The PEP2**403** intercepts this message and removes it from the data stream. The PEP2**403** reduces its path MTU estimate and retransmits the data in smaller packets **410** to the server **405**, which can now be forwarded by the router **404**.

[**0048**] This method by which the original packet is segmented and re-packetized has two possible variants in accordance with the invention. The first variant involves merely segmenting the original large packet into multiples of the new path MTU and a remainder number of bytes. The second variant involves the PEP devices treating the data in the packets as a byte stream and combining any remainder, as described immediately above, with data bytes from the next packet in the data stream to form the maximum number of new path MTU estimate sized packets. This second variant may also be combined with a timer which controls the maximum time a remainder of a large packet may remain buffered while waiting for a following contiguous packet to arrive.

[**0049**] **FIG. 5** shows the overall inventive protocol packet format **500** according to an embodiment of the invention. The packets are the same size as TCP packets without TCP options. Two 16-bit port numbers, a source port number **501** and a destination port number **502**, are used at the beginning of the packet. These port numbers take the same format as the TCP port numbers. A header length field **503** taking the same format as TCP is included. This is followed by a reserved field **504**, which again preserves the TCP packet format. An urgent flag **505** is preserved from TCP. An acknowledgement flag **506** is reused by the inventive protocol. A push flag **507** is also preserved from TCP. A reset flag **508** may be preserved from TCP or reused by the inventive protocol. A synchronise flag **509** is preserved from TCP. A finish flag **510** is also preserved from TCP.

[**0050**] A 16-bit TCP window field (not shown) is reused by the inventive protocol as an inter-PEP flow-control window field **511**. A 16-bit TCP checksum **512** functions in the same way as the TCP checksum. An urgent pointer **513** is preserved from TCP. In order to distinguish between the multiple different acknowledgement types, an acknowledgement type bit flag **514** is used along with the acknowledgement flag. Packet bit flags, for example **515**, are used to indicate the receipt or loss of individual packets. A 24-bit protocol packet number **516** is used when the packet contains data. If the packet is a pure acknowledgement packet, the bits of the 24-bit packet number field may instead be used as further bit flags for indicating the loss or receipt of packets. An inventive protocol acknowledgement field **517** is used as a reference point for the individual bit flags. For example, this acknowledgement field may indicate the newest packet acknowledged (highest packet number) or the oldest packet acknowledged (lowest packet number) and the bit fields could indicate contiguous packets, older or newer, than this value, respectively.

[**0051**] In accordance with the embodiments of the invention, when a data packet arrives at its receiver, an acknowledgement packet is formed if one is not already being constructed. For each subsequent received packet, the acknowledgement packet is updated with positive and negative acknowledgement information from the packets received or missing. On a point-to-point link, missing packets may be assumed to have been lost. A timer may be used to bound the RTT and generate a minimum acknowledgement rate. If the timer expires, the acknowledgement will be sent. Each time a new acknowledgement is constructed, the most efficient format for the current pattern of packets received and missing will be chosen. As other packets are received or found to be missing, the acknowledgement format will be changed so as to always use the most efficient format according to current information. Once an acknowledgement can hold no more information, it is sent.

[**0052**] For every inventive protocol packet transmitted between PEPs, a copy must be buffered in a retransmission buffer to allow for retransmission, if necessary. This scheme allows for reliable transfer of data from end to end. For each buffered packet in the transmission buffer, a timestamp is also stored, the timestamp being based on any real or pseudo time clock with fine enough resolution. When a packet is retransmitted, the timestamp is updated.

[**0053**] The operation of the invention is now described in greater detail. The connection-opening mechanism for minimizing periods of endpoint desynchronization would operate in the following manner, as shown in **FIG. 1**. The client **101** and the first PEP1**102** would complete the local TCP 3WHS; the TCP SYN segment being converted to the satellite protocol and sent across the satellite link. The second PEP2**106** would initiate the second TCP connection and monitor for any ICMP messages in response. If an ICMP message were detected, it would be forwarded across the satellite link and the second TCP connection closed at the second PEP2**106**. The ICMP message would again be processed, this time by the PEP1**102**, which would forward the message, and then send a following TCP reset segment to guarantee the connection teardown at the client **101**. This provides the client **101** with the maximum information while minimizing the packets over the satellite link.

[**0054**] One embodiment of the invention would also include the method by which the PEP devices interact with the PMTUD mechanism. In this embodiment, remainders of packets may be combined with data bytes from following packets to maximize the number of path MTU estimate sized packets transmitted. The combination of this treatment of the packet stream as a byte stream at the PEP devices and a timer to wait for following packets should minimize further small (less than path MTU estimate) packets being sent. The result would be increased efficiency due to less header overhead and, consequently, improved throughput. Other node processing would also be reduced.

[**0055**] The inventive packet format may be used with the acknowledgement number as either the newest or oldest packet acknowledged with very little difference in performance. If the acknowledgement number is the newest packet being acknowledged, then the bit flags represent older packets and may re-acknowledge already acknowledged packets which may increase processing at the acknowledgement receiver. If the acknowledgement number used is the oldest, then a convention must be established as to which bit flag acknowledges the newest packet. For example, a timer may limit the maximum time between acknowledgements so that there may not be enough packets to be positively or negatively acknowledged to require all the packet bit flags. In this case, if there were no convention as to which of the bit flags were valid, the acknowledgement could trigger the retransmission of any packets not yet received.

[**0056**] One embodiment of the invention utilizes an acknowledgement scheme that uses a timer to guarantee a minimum acknowledgement rate and a maximum RTT. A value of between 200 and 500 ms, for example, would be typical, adding minimal time to the RTT but enough time to allow an acknowledgement to acknowledge multiple packets.

[**0057**] The time-stamping mechanism to prevent unnecessary retransmissions may be used with either a time of transmission or expected time of acknowledgement, with very little difference. If the expected time of acknowledgement is calculated and then stored, a direct comparison may be made to the current time which may be more efficient computationally than storing the time of transmission and making a calculation and comparison each time the packet must be checked.

[**0058**] It will be apparent to those skilled in the art that various modifications and variations can be made to this invention without departing from the spirit or scope of the invention. Thus, it is intended that the present invention covers the modifications and variations of this invention provided that they come within the scope of any claims and their equivalents.

1. A method for accelerating an opening of a connection in a communications system for transmitting and receiving data packets, the method comprising the steps of:

intercepting a TCP SYN connection request from a client, the TCP SYN intercepted by a first Performance Enhancing Proxy (PEP);

responding to the intercepted TCP SYN connection with a TCP SYN/ACK, the step of responding performed by the client;

responding to the first PEP with the TCP ACK, the responding step performed by the client;

converting the TCP SYN to another protocol and sending a message over a link, the converting and sending steps performed by the first PEP;

receiving the message over a link and converting it back to the TCP SYN, the receiving and converting steps performed by the second PEP;

sending the TCP SYN to a server, the sending step perfomed by the second PEP; and

listening for ICMP messages generated in response to the TCP SYN, the listening step performed by the second server.

2. The method of claim 1, further comprising the steps of:

receiving the ICMP message related to a connection that is currently being established, the receiving step performed by the second PEP;

removing said ICMP message from the network; generating a reset message;

forwarding the reset message to the first PEP; and

forwarding the reset message to the client, the forwarding step performed by the first PEP.

3. The method of claim 1, further comprising the steps of:

receiving an ICMP message related to a connection that is currently being established, the ICMP message received by the second PEP;

forwarding the ICMP message to the first PEP; and

forwarding the ICMP message to said client, the ICMP message forwarded by the first PEP.

4. The method of claim 3, further comprising the steps of:

generating a reset packet and sending it over the link to the first PEP, the generating and sending steps performed by the second PEP; and

forwarding the reset message upon returning the ICMP message to the client, the forwarding step performed by the first PEP.

5. The method of claim 3, further comprising the steps of:

generating a reset message upon receiving the ICMP message, the generating step performed by the first PEP; and

sending the reset message to the client after the forwarding of the ICMP message, the sending step performed by the first PEP.

6. A method of supporting an end-to-end PMTUD mechanism in a communications system for transmitting and receiving data packets, comprising the steps of:

identifying messages to intercept via a destination IP address and an encapsulated packet fragment;

intercepting incoming ICMP "Fragmentation Needed, DF Set" messages from a TCP end connection;

re-sizing a path MTU estimate for an indicated destination IP address at a PEP to a value indicated by the ICMP messages or if no value is indicated to a default of 576 bytes; and

re-packetizing an original packet according to a newly set path MTU estimate; and

re-transmitting from PEP previously discarded data.

7. The method of claim 6, wherein the step of re-packetizing further comprises:

dividing data of a packet requiring subdivision into path MTU sized portions and a possible remainder of less than path MTU bytes;

creating a new set of TCP headers for subdivisions of the original packet that preserve TCP connection information contained in the original packet; and

creating a new set of IP headers for the subdivisions of the original packet that preserve the IP information contained in the original packet.

8. The method of claim 6, wherein the step of re-packetizing futher comprises:

treating the data as a byte stream; and

combining portions of multiple large packets into smaller packets.

9. A packet header for transmitting and receiving data packets in a communications system for transmitting and receiving data packets, comprising:

a first field containing a source port number;

a second field containing a destination port number;

a third field comprising 8 individual bit flags;

a forth field containing a packet number;

a fifth flag field;

a sixth field including either 0 or more individual bit flags;

a seventh, field containing an acknowledgement number;

an eighth field containing the header length in units of 4 bytes;

a ninth reserved field;

a tenth flag field;

an eleventh flag field;

a twelfth flag field;

a thirteenth flag field;

a fourteenth flag field;

a fifteenth flag field;

a sixteenth field containing an advertised window size;

a seventeenth checksum field; and

an eighteenth field containing an urgent pointer.

10. The packet header in accordance with claim 9, wherein the fourth header field is no greater than 31 bits and the seventh header field is no greater than 31 bits.

11. The packet header in accordance with claim 9, wherein the fourth field is utilized as 24 1-bit flags for acknowledging individual packets.

12. A method for use in a communications system for sending and receiving data packets for acknowledging a plurality of data packets to automatically reduce the number of acknowledgement packets under differing link conditions, the method comprising the steps of:

constructing an acknowledgement packet having a first format;

updating the acknowledgement packet as the packets are considered at least one of received or missing;

converting the acknowledgement packet to an alternative format in order to minimize the packets required to be sent; and

sending the packet when it is full.

13. The method of claim 12, further comprising the step of using a timer to limit a maximum delay between the acknowledgement packets.

14. A method for maintaining information on transmitted packets in a communications system for sending and receiving data packets, the method comprising the steps of:

transmitting a packet;

locally storing a timestamp representing a time of transmission of the packet;

determining a packet loss; and

determining if a retransmission may take place based upon the determined packet loss.

15. The method of claim 14, wherein the timestamp represents an expected time of reception of a confirmation of a successful receipt.

16. The method of claim 14, wherein the step of determining a packet loss entails a failure to receive an acknowledgment as indicated by the expiration of a timer or receipt of a negative acknowledgement or an acknowledgement for a later packet.

17. The method of claim 14, further comprising the step of determining which packet to retransmit.

18. The method of claim 17, wherein the step of determining which packet to retransmit further includes the step of finding all packets where a packet number is less than the packet determined missing and a time stamp indicates that an acknowledgement should have been received.

* * * * *