(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 May 2003 (22.05.2003)

PCT

(10) International Publication Number
WO 03/043285 A2

(51) International Patent Classification⁷: H04L 29/00

(21) International Application Number: PCT/US02/36282

(22) International Filing Date: 13 November 2002 (13.11.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/333,608 13 November 2001 (13.11.2001) US

(71) Applicant: **EMS TECHNOLOGIES, INC.** [US/US]; 2850 Colonnades Court NW, Norcross, GA 30071 (US).

(72) Inventors: **NEALE, Jason**; 2850 Colonnades Court NW, Norcross, GA 30071 (US). **PETHER, Andrew, M.**; 2850 Colonnades Court NW, Norcross, GA 30071 (US). **MOHSEN, Abdul-Kader**; 2850 Colonnades Court NW, Norcross, GA 30071 (US). **BEGIN, Guy**; 2850 Colonnades Court NW, Norcross, GA 30071 (US).

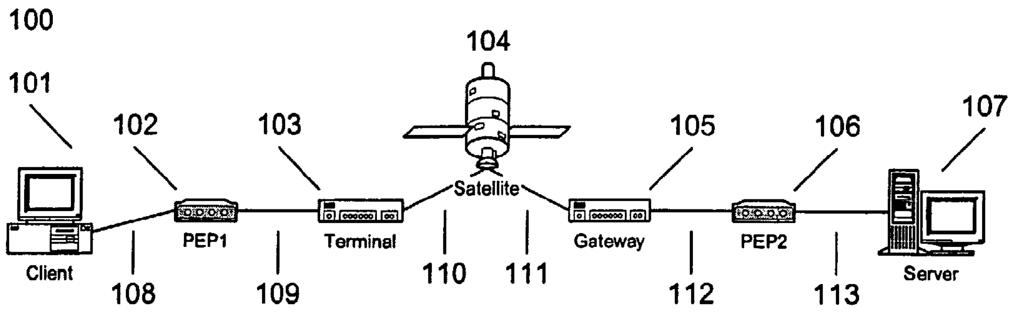
(74) Agents: **CROWSON, Celine, Jimenez** et al.; Hogan & Hartson, L.L.P., 555 Thirteenth Street, N.W., Washington, DC 20004 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

(54) Title: FLOW CONTROL BETWEEN PERFORMANCE ENHANCING PROXIES OVER VARIABLE BANDWIDTH SPLIT LINKS



(57) **Abstract:** The invention provides a method and system for dealing with the flow of data between one Performance Enhancement Proxy and another in the context of split links for TCP over satellite performance improvement. The environment according to the invention may include one or more intermediate nodes experiencing variable latency and bandwidth allocations. The method and system manage the data flow to ensure bandwidth fairness between competing TCP/IP connections, prevention of PEP receiver buffer overflow and near 100% usage of the available satellite bandwidth without the need for conventional TCP/IP ACK driven probing algorithms. The near 100% usage of capacity being achieved, in part, through PEP to intermediate node message exchange.

FLOW CONTROL BETWEEN PERFORMANCE ENHANCING PROXIES OVER VARIABLE BANDWIDTH SPLIT LINKS

CROSS-REFERENCE TO RELATED APPLICATIONS

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 November 13, 2001.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to data telecommunications satellites, and more specifically to the use of hardware and/or software, referred to as Performance Enhancing Proxies (PEPs), to optimize the performance of the Transmission Control Protocol (TCP) over satellite links with varying bandwidth.

2. Description of Related Art

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 Internet Protocol (IP) based networks have a variety of effects on the behavior of protocols in the IP suite.

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, providing reliability to applications and building 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 TCP 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, its use with geosynchronous satellite links being an example. 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) were introduced.

In any point-to-point communications system, such as PEP to PEP, flow control between the two elements is important to prevent starvation or over-flow of data in any other Intermediate Device (such as the Terminal) as well as the receiving point. For the purposes of describing the invention, a PEP shall be described as an Intermediate Node between the endpoints of a connection, and any network element between the PEPs, such as a Satellite Gateway, Satellite or Satellite Terminal, shall be described as an Intermediate Device. Furthermore, a connection refers to an end-to-end connection between a client and a server which is broken up into three connection segments, client to PEP, PEP to PEP and PEP to Server, such that the client or server remain largely aware of the splitting. Starvation of data implies inefficient use of available communications capacity, and overflow of data implies packet loss and retransmissions of data. Moreover, where the point-to-point communications system includes an intermediate device that is required to request bandwidth based on the amount of data in its queue, then ensuring the queue is adequately provisioned guarantees that capacity requests will be made.

Ensuring any particular intermediate device (such as the Terminal) between the two points (PEPs) is accurately supplied with an appropriate amount of data is difficult if the amount of bandwidth capacity allocated to the device is constantly changing. However a bandwidth-on-demand satellite system scheme will naturally change the allocated capacity to a given terminal (based on, but not limited to, Traffic conditions and Quality of Service (QoS) agreements).

TCP performance is typically degraded to some extent in terms of lowered throughout 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.

PEPs may function as one or more Intermediate Nodes 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 to 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.

A 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 connection splitting PEP uses more than one PEP in an end-to-end connection, most commonly, two PEPs are used, although the invention can be applied to systems using greater number of PEPs with the FP protocol of this invention. 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 Systems Control Protocol Suite – Transport Protocol (SSCPS-TP) or even non-standard modified TCP.

Current versions of TCP use a window mechanism and acknowledgement (ACK) driven algorithms, such as slow-start and congestion-avoidance algorithms, to manage the flow of data from the sender to the receiver to mitigate the effects of congestion and prevent over-flow of the receiver's buffers. These algorithms often mistake transmission errors as congestion and fail to fully-supply the satellite link with data. Although the TCP window-scaling option helps with the later and fast-retransmission / fast-recovery help with the former, the overall link usage often remains well below the available capacity. The aforementioned satellite protocols and PEPs address some of these problems by facilitating larger windows, discriminating between congestion losses and in some cases making use of link rate knowledge (where that rate is constant). However, all of the PEP solutions use some form of capacity probing procedure (akin to a slow-start technique) and thus are unable to immediately employ the full link capacity without fear of losses (due to packet discard at an Intermediate Device). More importantly, none of the other solutions are able to fully supply a variable bandwidth and/or latency link for the duration of a TCP transfer. Additionally, they are unable to communicate directly with Intermediate Devices, relying instead on PEP-to-PEP messages to determine capacity, with the inevitable reduction in throughput associated with a satellite round-trip delay message exchange. As a solution to this problem, the method and system described in this invention facilitate a near 100% usage of link capacity for the entire duration of the PEP-to-PEP transfer without the risk of packets being discarded or receiver buffer-overflow.

In terms of the sharing of resources to guarantee fairness while not limiting hungry connections when satellite resources permit, conventional TCP merely dedicates a buffer per end-to-end connection and manages those connections independently. Thus TCP is unable to quickly make use of free capacity for hungry connections when other connections start to slowly supply the link, relying instead on a slow-ramp through congestion avoidance. Indeed, the overall throughput is often reduced as a result of a lack of acknowledgments. Although many of the PEPs and other protocols are able to prevent the flow from diminishing because of a lack of ACKs, none of them include a flow fairness technique to ensure 100% supply to the link and fairness (assuming TCP/IP traffic supplies allow). Herein, fairness is defined as the ideal condition where each connection has an equal share of the link, as long as it has enough traffic to use this portion of the resources.

If a connection is running more slowly, its unused share of capacity will be provided for the use of all other connections, in a fair way.

SUMMARY OF THE INVENTION

The invention provides methods and systems for the management of the flow of data between two PEPs where the link conditions, in terms of bandwidth resources and latency, are changing in relation to network traffic conditions and QoS profiles, among other things. This changing bandwidth condition is characteristic of a Bandwidth on Demand (BOD) satellite communications system.

The invention further provides systems and methods that enable the fair distribution of satellite resources between a number of competing TCP connections at a PEP, while facilitating the full usage of the available satellite bandwidth by accurately supplying a particular Intermediate Device with sufficient data and preventing the receiving PEP's buffers from overflowing. Additionally, the distribution process according to the invention prevents slow running TCP connections, at the receiver end, from unfairly being allocated excess link capacity whilst allowing for full usage of bandwidth at the transmitting PEP's end by distributing unused satellite capacity among hungry TCP connections.

The methods and systems in accordance with the invention allow satellite resources to be fairly shared between competing TCP connections while also facilitating the full use of the link capacity by hungry connections should resources allow. The overall flow strategy in accordance with the invention allows the fair sharing or satellite link resources between connections while achieving near 100% usage and not over-flowing the receiving end's buffers.

The invention provides satellite communication systems where the bandwidth between two PEPs is controlled by a Gateway and is subject to wide variations due to competing traffic, among other things. As an example, communication between two PEPs relies on an Intermediate Device (Terminal), forward and return satellite links and a Gateway. In such a network, numerous terminals could be operating with the Gateway, each terminal receiving data packets from a PEP, then requesting and receiving bandwidth allocations from the Gateway. Such a Bandwidth-on-Demand (BoD) scheme typically facilitates a fast variation in bandwidth allocation and substantial latency changes dependent on the number of terminals in operation, their traffic supplies and guaranteed quality of service agreements.

Thus, the invention provides for improved performance of TCP over a satellite link or other large bandwidth delay network. Moreover, the invention provides for the management of the flow of traffic from a PEP to an Intermediate Device in a satellite environment where the bandwidth and latency are fluctuating. This may be accomplished in part by replacing TCP with a new transport protocol, the EMS (proprietary) Flight Protocol (FP), over the wireless satellite link only and maintaining TCP connections over the terrestrial portions of the end-to-end connection. TCP performance over GEO links is traditionally very poor from a user perspective in terms of transfer time and throughput for web browsing and file transfer among other applications relying on a TCP transport layer.

The above aspects of the invention are achieved by addressing and in part accessing the characteristics of the satellite link, including available capacity, and treating a lack of acknowledgements from the receiver as errors and not congestion. The PEP invention described herein will improve the throughput and transfer times and achieve a higher utilization factor of the assigned link rates.

BRIEF DESCRIPTION OF THE DRAWINGS

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:

Figure 1 is an illustration showing the overall satellite network with the location of the PEPs, as exemplary of connections and equipment in a distributed, connection splitting PEP deployment;

Figure 2 illustrates the end-to-end satellite protocol stacks for TCP/IP utilizing the PEPs; and

Figure 3 shows an algorithm and procedures for making a PEP-to-intermediate device (Terminal) communication, and in particular, steps in making a query of the Terminal's buffer status and determining the amount of data to be sent to the Terminal.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The invention is described in the context of a bi-directional transfer of data between a client and a server over a communications link consisting of both terrestrial segments and up and down segments, or links, over satellite, as generally shown in Figure 1. 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.

The PEP component consists of two main parts: a TCP Emulator (TCP*) and a Flight Protocol (FP) Processor. While the operation of inventive components and methods of the invention are different depending on whether they are functioning in a Receive or a Transfer mode, i.e., depending on whether the data is being sent by the client or server, it is to be understood that the components and methods are reciprocal. A description of the operation for sending data, for example, from a client to a server is essentially equivalent to data being sent in the reverse direction, from server to client, in this respect the PEP system is reciprocal. Figure 2 presents an illustration generally showing where the TCP* and FP layers sit in a conventional stack. The PEP is designed to be used in pairs, one on each terrestrial side of the satellite link.

The TCP Emulator (TCP*) is present in the transmit and receive PEPs and behaves as if it is a TCP connection endpoint. The TCP Emulator transparently interrupts the TCP connection going from the client to the server (or server to client) and acts as a TCP endpoint. It translates TCP traffic into FP traffic, the inventive transport protocol used over the satellite link (or other large bandwidth * delay network) between the two PEPs. PEP1 is the first PEP in the transfer chain (the Transmitter) and PEP2 is the second PEP in the transfer chain (the Receiver), whichever direction of packet is being discussed. The names PEP1 and PEP2 do not necessarily apply to the terminal-side or gateway-side PEPs respectively. The role of the TCP emulator depends on the function of the PEP. In the Transmit PEP, the TCP Emulator converts TCP segments into FP packets. In the Receive PEP, the TCP Emulator (TCP*) receives the FP packets, converts them back to standard TCP packets and transmits them over a new TCP connection to their final destination: the communication's endpoint. The client and the server are the communication system's endpoints or end users.

With the introduction of the two PEPs, the standard connection going from the server to the client will be replaced with the following three connections:

1. A TCP connection from the client to the first, terminal, PEP (PEP1)

2. A FP connection going from PEP1 to PEP2 (gateway PEP)
3. A TCP connection from PEP2 to the server

The TCP Emulator (TCP*) emulates a standard TCP connection between the external end user, the client and server, and converts TCP/IP packets into Flight Protocol (FP) packet 'shells'. The TCP Emulator filters TCP/IP packets entering from the outside world (the real end points) and emulates the TCP behavior of the destination transport layer. This behavior includes the TCP three-way handshake, acknowledgements, flow control, re-transmissions and all other TCP functionality. It also manages the flow of traffic to the Flight Protocol Processor section of the PEP.

The Flight Protocol FP is the inventive transport protocol that is used over the satellite link between the transmit and receive PEPs that are connected, respectively, to a terrestrial/satellite gateway and a user terminal. The FP is optimized to operate over this link by not using the TCP Slow Start and Congestion Avoidance algorithms, and instead, utilizes the full available satellite capacity immediately and consistently throughout the lifetime of the connection, also improving link utilization efficiency. This throughput utilization is achieved despite bandwidth fluctuations, prevalent in a BoD satellite network, by linking the PEP with the Intermediate Device experiencing the said fluctuations. Other future protocols may also be used, the FP should be considered as merely one current example of such suitable protocols. While this invention is described with reference to the FP, it is to be understood that the invention is not limited to this protocol. It can equally be used with and applied to a variety of other protocols.

The FP also avoids the delays associated with the TCP three-way handshake by not using a pre-data handshaking (negotiation) mechanism of any form. Instead, the FP connection Initiator simply informs the remote entity that a new connection has been created and then immediately begins to send data. The assumption is that the FP connection will be created successfully unless evidence demonstrates to the contrary. The alternative rationale, as implemented by conventional TCP, assumes failure until success is explicitly signaled. This feature of the FP of this invention removes an additional one-off delay (per connection) that is significant for very small files or short duration transfers such as those typical of current web pages that at present comprise the greatest volume of Internet traffic. This setup mechanism may be used with either a half or full duplex connection allowing for bi-directional data communication on a single connection or alternatively with two associated simplex connections.

The invention involves the principle of an Intermediate Device breaking an end-to-end TCP and splitting the connection into a TCP/IP protocol and TCP/IP connections by the use of PEPs; and then managing the flow of data between the PEP and the Intermediate device (Terminal) by knowledge of the status of the Terminal queue. One of ordinary skill in the art will recognize other variations, modifications, and alternatives.

The principle involves the PEP sending sufficient data to at least partially or fully fill the Terminal queue (to a pre-defined level), and after adequate time for some of the data to be transmitted, checking the status of the Terminal, using some protocol, and then re-filling the Terminal queue to the previous level. This method could be as simple as the PEP sending regular SNMP get messages to the terminal MIB, the terminal sending SNMP traps when its queue reaches a certain level or in the case where the PEP and Terminal are co-located in the same device some form of direct linkage. The flow of data from the PEP to the Terminal could be made more accurate by the PEP having knowledge of the current and future bandwidth allocations of the intermediate device (Terminal).

Beyond the management of the flow of data from the PEP to the intermediate device two other functions are key to the successful flow of data from the PEP to PEP; firstly not overflowing the receiver's window and secondly the fair distribution of the satellite bandwidth between competing TCP connections. The former is dealt with by the use of conventional window mechanisms, as in TCP, and the latter is achieved by the sharing of resources by the number (n) of connections by the use of transmission storage. For example where the maximum bandwidth-delay product of a communications link is (x), the amount of capacity given to a connection, before allowance for the PEP to Intermediate device flow, is (x/n). When the sum of all actual capacity usage by each individual connection (n) is less than (x), then the available capacity is shared by any connections wishing to exceed their allocations. This allows the full capacity of the link to be employed if several incoming TCP/IP connections (from the terrestrial side) are running slowly.

Additionally, when new connections arrive, they are guaranteed (x/n) capacity even if the result is a temporarily over-supply of capacity; the existing connections flows are naturally reduced to the new lower rate. This is achieved by the use of a buffer to manage the flow, with packets only being allowed storage in the buffer when the afore mentioned rules are met. For example, if 1Mb/s was previously shared between 10 connections a new connection would result in the over supply of 90.909Kb/s (10*100Kb/s + 1* 90.909Kb/s). However the existing connections are forced down to around 90.909Kb/s by preventing new incoming TCP/IP packets from the terrestrial side to progress from the TCP Emulator to the FP, until sufficient buffer space is created. Any temporary over-supply, until the system returns to the steady-state, is achieved by allowing the new connection packets to be stored in an output queue. The existing connections being denied further capacity until the new connection has reached its allocation and sufficient space exists in the output queue. The end result being a natural back-pressure to the TCP senders of the over-supplying connections and prevention of the new connection being starved of capacity when the link was previously fully occupied.

The transmission of data from the PEP to the intermediate device and onwards to the other PEP, depends on the passing of three conditions laid out above; namely: sufficient transmission buffer space (allowing for the rules above), sufficient space in the receiver window (allowing for the amount of data already transmitted since the last window update (as in TCP)) and sufficient capacity / buffer space at the intermediate device (Terminal). If any of these tests fail, data can be stored in an applicable Input or Output queue within the PEP (TCP Emulator or FP) thereby creating natural back pressure to the TCP sender in the regular window updates sent form the PEP to the TCP sender.

The invention provides for a return link in the Terminal-Satellite-Gateway direction that is shared between a number of Terminals and thus subject to varying bandwidth conditions. Likewise the output queue in the Terminal that is tested and then filled is a queue for all traffic. Variations are possible, for example, the Gateway could enjoy varying bandwidth conditions facilitating the need for such a scheme or the Intermediate Device could have several output queues, requiring a different parameter to be polled (e.g. the TCP/IP queue).

Referring now to the method of implementing the transfer of data in this environment according to an embodiment of the invention, Figure 1 illustrates a simplified view of a performance enhancing proxy (PEP) communications system **100** including equipment and links involved in a PEP deployment in a satellite communication link environment. In Fig. 1, a client **101** initiates a connection attempt to a server **107** via a satellite **104**. The client is connected by a local area network (LAN) segment **108** to a terminal-side PEP1 **102**, via another LAN segment **109** to a satellite terminal or satellite modem of some form **103**. Traffic from a terminal **103** passes over the communications links **110** and **111** via the satellite **104** to the Gateway, central hub equipment or other satellite modem **105**. Traffic leaving the Gateway passes via a LAN segment **112** to a gateway-side PEP2 **106**. The PEP2 106 then sends the traffic via a wide area network (WAN), such as part of the Internet, **113** to the server **107**. The traffic may be a client request, which could generate server response traffic in the reverse direction.

Data transfer from the client **101** to the terminal PEP1 **102**, and from the gateway PEP2 **106** to server **107** uses known TCP protocols. Data transfer over the satellite link from the PEP1 **102** to the PEP2 **106** uses the flight protocol (FP) described in greater detail above. The data transfer is thus sent over TCP-FP-TCP protocol links.

Figure 2 shows the stacks involved in the various elements in a TCP-FP-TCP transfer. Fig. 2 shows the stacks of a transfer end point application which include an application/presentation/session layer **285**, a TCP layer **206**, an IP layer **212**, an Ethernet or equivalent layer **222** and a UTP or equivalent layer **233**. Fig. 2 also shows the stacks in a first PEP1 (first PEP in the transfer direction front end) which includes a UTP or equivalent layer **234**, an Ethernet or equivalent layer **223**, a modified IP layer **213**, a modified TCP layer (TCP*) **207**, an application/presentation/session layer **201**, an FP layer **208**, a modified IP layer **214**, an Ethernet or equivalent layer **224** and a UTP or equivalent layer **235**. A transfer originating from an end point **285** is acted upon using conventional data communication rules by the TCP layer **206**, the IP layer **212**, the Ethernet **222**, the UTP layer **233**, the UTP layer **234** and the Ethernet **223** before it is grabbed by the modified IP layer **213** and passed up the stack to the modified TCP layer **207**, translated by the application/presentation/session layer **201** and then managed by the FP layer **208** before being sent through the modified IP layer **214**, the Ethernet **224** and UTP layer **235**.

Following the arrows in Figure 2 illustrates the conventional stacks of the satellite gateway and terminal and also the stacks of a PEP2. In PEP2, incoming FP packets are grabbed by a modified IP layer **219**, after passing through the physical UTP layer **240** and link Ethernet **229** layers, and then managed by the FP layer **209** before passing, via application/presentation/session/layer **204** to TCP* **210** which along with the modified IP layer **220** manages the terrestrial TCP/IP connection.

As described earlier, the invention allows for the exchange of information between the Intermediate Device (the "Terminal") and PEP1.

This is to aid overall flow control and the sharing of bandwidth within the overall flow, between incoming (terrestrial network-to-PEP) and outgoing (PEP-to-terrestrial network) TCP connections by managing buffer resources associated to the link Bandwidth Delay Product (BDP). Additionally, the novel overall flow control scheme ensures near 100% throughput, assuming sufficient TCP/IP traffic as an input, while allowing for the receiving PEPs buffer requirements, through conventional window mechanisms, and fairly sharing the capacity between a number of connections. The following section describes further aspects of the invention in greater detail.

From the PEP-Intermediate Device (Terminal) point of view, actual return link capacity available for PEP traffic at any given time is normally an unknown. This issue mainly relates to the terminal population and its access to return link capacity. When packets are queued in the terminal **103** shown in Figure 1, capacity is requested related to the number of packets being queued, however a capacity request may be only partially fulfilled, or not fulfilled at all. In this case, it becomes possible for terminal PEP1 **102** to oversupply or under supply the terminal which may result in dropping or low usage. This dropping may increase buffering at Intermediate Devices and lead to greater jitter and latency that results in low bandwidth usage and hence a loss of revenue potential. In brief, the PEP is unaware of the return link capacity because of :

1. Limited knowledge about terminal queue status and size.

2. Dynamic capacity assignment to the terminal.
3. Non zero probability of loss on the local PEP-terminal link.
4. No confirmation of data transfer between PEP and terminal.
5. Unknown amount of traffic originating from the terminal itself, notably Operations Administration & Management (OA&M) traffic.

For accurate flow control purposes, what is really needed is information concerning the status of the Intermediate Device output queue, and for the PEP to supply data packets to fill the queue to a pre-determined level. We note that this queue could be an output queue for all packets or purely for TCP/IP traffic.

It is possible to obtain some valuable information by taking advantage of functionality often provided for network management. What follows is a description of how a PEP device could have knowledge of the Intermediate Devices (Terminal) queue status, and use this information to ensure that the queue is neither over-supplied nor under-supplied. Although the solution uses Simple Network Management Protocol (SNMP), it should be understood that other protocols could be used to exchange information and, moreover, the Intermediate Device and PEP could be co-located in the same unit with some form of direct linkage to erase the need for protocol driven communication. Furthermore, if the PEP were located in the terminal, it could make use of known future capacity allocations to manage the flow of incoming TCP/IP connections more accurately. For example, if the PEP knows of any Constant

Rate Assignments (CRAs) to the Terminal PEP1 **102** (i.e. not dynamic and varying capacity) it can at least ensure that sufficient TCP/IP flow is maintained to fill its future assignments and possibly buffer space.

The Simple Network Management Protocol (SNMP) is a request-reply protocol running over User Datagram Protocol (UDP). SNMP is an asymmetric protocol, operating between a management station and an agent. Intermediate Devices (Terminals) are expected to support SNMP messaging for network management. The Management Information Base (MIB) specified for Terminals incorporates several (but not all) generic MIB-II (MIB version 2) objects.

The following assumptions are made:

1. The PEP behind the terminal can assume the role of a local management station, thereby getting read access to MIB-II objects of the Intermediate Device (Terminal) using the appropriate community name.
2. Buffering in the terminal is limited by number of packets rather than by volume of data (although we note that data bytes could be used if the Intermediate Device output queue length in bytes were known).
3. The PEP either knows (or otherwise, can determine) the maximum size (in number of packets) of the Intermediate Device output queue. This parameter may be called MaxQlen.

Potentially useful MIB-II objects for deriving buffer occupancy information are in the interfaces group (the group of MIB-II objects related to the network interfaces of the device), specifically, for any of the interfaces given below. While these are specific, it should be understood that one of ordinary skill in the art would recognize possible use of other interface variations, modifications, and alternatives.

1. IfOutQlen: is a gauge indicating the number of packets in the outbound queue.
2. IfOutOctets: is a counter indicating the total number of octets transmitted out of the interface including framing octets.
3. IfOutUcastpkts: is a counter indicating the number of unicast packets whose transmission to a single address was requested.
4. IfOutNUcastpkts: is a counter indicating the total number of packets whose transmission to a multicast or broadcast address was requested.

The basic idea for PEP to Intermediate Device communication is for the PEP to regularly send SNMP GetRequest queries to the Intermediate Device (Terminal) for the IfOutQlen of the satellite link interface, although it is to be noted that the Intermediate Device could send the PEP an SNMP Set when its IfOutQlen reaches a pre-defined limit. The returned value allows the PEP to control the amount of data that is fed to the terminal.

From the point of view of a PEP, the algorithm is packet-driven: no actions are performed unless there are packets to send on the satellite link. A flow chart **300** for the algorithm is illustrated in Figure 3. The algorithm proceeds as follows: In step 301, the variables FreeQ_i and FreeQ_(i-1) are first initialized. The process then moves to step 302. In step 302, the PEP then waits until there are packets to send to the terminal (Packets_to_send >0). The process then moves to step 303. In step 303, when there are packets to send, a test is performed to check whether the PEP is already aware of a certain amount of buffer space available on the terminal (FreeQ_(i-1)). If there is buffer space, the process moves to step 304, otherwise the process moves to step 305. In step 304, the PEP sends packets up to (FreeQ_(i-1)) packets. The process then moves to step S305. In step S305, the PEP sends a SNMP GetRequest for IfOutQlen. The terminal responds with a SNMP GetResponse containing a value for IfOutQlen. The process then moves to step S306. In step 306, the PEP computes FreeQ_i = (MaxQlen - OutQlen – Margin). The process then moves to step 307. In step 307, the PEPs send the minimum of {FreeQ_I, Packets_to_send} packets to the terminal. The process then moves to step 308.

In step 308, the system determines whether FreeQ_i > Packets_to_send. If yes, the process moves to step 309 where there is more buffer space available on the terminal, and FreeQ_(i-1) is updated to the remaining space, i.e., FreeQ_(i-1)=FreeQ_i - Packets_to_send and the process returns to step 302. If no, the process continues to step 310. In step 310, If (FreeQ_i >= Thresh) then the process returns to step 305. Otherwise, the process moves to step 311 to wait Query_timer and then returns to step 305.

The following parameters employed by the process above are described:

1. MaxQlen: maximum size (in number of packets) of the Intermediate Device (Terminal) satellite link outbound queue.
2. Query_timer: a delay to avoid querying the terminal too frequently. Its value may be set based on the time required to empty a full queue at a given return link maximum rate.
3. Thresh: a threshold (in number of packets) for deciding whether the queue is full enough to wait before going back to query. A suitable low value should be used.
4. Margin: a security margin to account for an unknown amount of traffic originating from the terminal itself, for inaccuracies in the OutQlen values reported. This should be set in such a way that the probability of dropping packets due to overflows at the terminal is suitably low to keep the costs of lost FP packets low.

The following variables are also defined:

1. FreeQ_i: free satellite link buffer space maximum on the terminal.
2. FreeQ_(i-1): remaining free satellite link buffer space available.

In a situation where packets to send are few and far between, FreeQ_i will be much larger than Packets_to_send and the PEP will go through steps 302 to 308 each and every time a packet to send arrives. This entails one SNMP query per run, since step 303 avoids one SNMP query when the PEP is already aware of some buffer space available on the terminal. The PEP could be made to wait for a certain delay before returning to step 302 after step

308, thereby avoiding sending a certain number of SNMP queries. However, there is no simple way to avoid this delay when a large number of packets to send arrive in a burst. Since the delay would slow transmission down in that very important case, it is more efficient to go back to step 302 and run through step 308 without delay. At worst, there will therefore be one query per packet if packets arrive one by one, but in such a case, SNMP queries will not hamper traffic since there will be virtually no traffic to interfere with.

In step 311 the PEP will wait until the expiration of a query timer if FreeQ_i is not greater than a threshold. This is to prevent un-necessary SNMP queries being sent to an Intermediate Device when there is little chance of the PEP sent packets having been sent beyond the Intermediate Device (Terminal). Another, lower timer, could be employed between the tests of step 310 and step 305 should FreeQ_i be greater than the threshold to allow the Intermediate Device (Terminal) time to process the packets.

The PEP should also make use of some form of PEP rate control clocking-out mechanism to ensure that the transmission rate from the PEP to the Intermediate Device (Terminal) does not over run the Terminal input queue and processing rate when sending bursts to fill up the said queue. Finally, note that packets arriving at the PEP from the terrestrial side, while the packet-driven algorithm is in process, are queued and subsequently drawn from the queue when the algorithm returns to step 302, this process being consistent with the packet driven format of the PEP.

Management of individual competing TCP/IP connection flows within the general flow is achieved by sharing a buffer space equivalent to the Bandwidth-Delay Product of the satellite link. In principle, every time a packet arrives from the terrestrial side it is associated with a connection and, before transgressing from the TCP* layer **207** to the FP layer **208** shown in Fig. 2, three tests are undertaken. This section describes the "buffer utilization" test that is employed to share resources among the different connections.

The PEP includes both a Transmit and a Receive Buffer. On arrival at the PEP, a packet from the terrestrial world is queued in the TCP* **207** layer's buffer. This buffer is equivalent to a conventional TCP buffer at the receiving end point, with the space in the buffer being linked to the advertised window on the TCP connection. This 'linkage' allows the TCP connection on the terrestrial link to slow down the flow of packets to the PEP as the TCP* buffer becomes full, in much the same way as TCP would slow down the flow of packet as the receiving ends buffers become full.

Before a packet can pass from the TCP* buffer to the FP layer it must pass the following rules to ensure that sufficient space exists in the FP buffer:

1. If the connection buffer utilization is less than its allowed buffer space (allowed buffer space = FP total buffer space / n connections, where the total buffer space is sufficient space to fill one bandwidth-delay product) then the packet can be pushed; or

2. If the total buffer utilization (among all n connections) is less than the FP total buffer space allowed, then the packet can be pushed.
3. Otherwise the packet remains in the TCP* buffers.

Once a packet is allowed to pass from the TCP* layer to the FP layer, it is also stored in the FP buffer and associated to a connection. The FP buffer facilitates re-transmissions if necessary and the sharing of the satellite link resources between competing TCP connections, as described.

It is important to note that the FP total buffer space can become artificially high. For example, if the global size is 3 Mbytes, then if three connections are opened, each connection will be allowed 1 Mbyte for its individual transmit buffer.

When opening a fourth connection, even if the three connections are using their whole transmit buffer, the new connection will immediately use its whole share of the transmit buffer. The FP global transmit buffer size will then become 3.75 Mbytes (1 + 1 + 1 + 0.75). The buffer utilization test allows to exceed the global transmit buffer size, but this situation will never last very long. The buffer utilization mechanism will redistribute equally the global transmit buffer size between the four opened connections. Each connection will be allowed to use a new individual transmit buffer size of 0.75 Mbyte (3/4), thus pausing the flow of data on the three excessive connections. Eventually, the global transmit buffer size will decrease to 3 Mytes (0.75 +

0.75 + 0.75 + 0.75) and therefore, the size of the global transmit buffer will fall back to 3 Mbytes in the steady state.

When space becomes available in the FP buffer (because of an acknowledgement signal, ACK, from the receiving ending indicating the successful arrival of a packet and thereby removing the packet from the FP Buffer), the PEP can pull a packet from the applicable TCP* buffer. This mechanism ensures that when the FP Buffer and even TCP* buffers are full, then the flow of a connection can be re-triggered by the arrival of ACKs. (note that, although the mechanism is not described as it is a known mechanism, the PEP includes a re-transmission procedure, based on timers, to ensure that data is re-transmitted if presumed lost, thereby guaranteeing the arrival of ACKs to re-trigger the flow or, if the satellite link is lost, a connection tear-down).

The following is an illustrative example for calculating default buffer sizes for the gateway and terminal-side PEP transmit and receive buffers. For purposes of this example, it is presumed that the total forward link rate is 60 mega-bits per second (Mbps) and the total return link rate is 48Mbps. Each terminal can be assumed to operate at a maximum receive rate in the forward link of 8Mbps and a maximum transmit rate in the return link of 2Mbps.

In the forward direction the transmit bandwidth and thus transmit buffer should assume 60Mbps. At the terminal (forward link receiver) the bandwidth is 8Mbps. In the return link, the transmit and receive bandwidths are 2Mbps and 48Mbps respectively. To calculate the Bandwidth Delay Product (BDP) and hence the buffer requirements, 2 more factors are required: the Round Trip Time (RTT) and what can be called the utilization factor.

The RTT is actually the time for a packet to be sent and the associated FP ACK to be sent back and clear the packet from the transmit buffer or open up more receive buffer space through window advertisements. This is the RTT between the PEPs and includes any ACK delay timers used to provide a minimum ACK frequency. From satellite testing, a value of 600ms is reasonable for Terminals that have been allocated a constant rate of bandwidth (Constant Rate Assignment – CRA) for this example and a delayed ACK timer of 500ms could be assumed. For Terminals operating in a pure BoD environment using Variable Bandwidth Dynamic Capacity (VBDC), the mean RTT measured was around 1400ms which would obviously require a larger buffer.

The utilization factor adjusts the calculated buffer sizes to maintain PEP/FP performance under heavier buffer utilization due to packet loss/corruption. From simulation and theory, we expect transmit buffer utilization to be around 102-105% of the calculated buffer size depending upon error conditions and packet sizes. Each FP packet that is lost must remain buffered for an additional RTT to allow for successful retransmission and acknowledgement.

At the receiver, a decision must be made regarding how many retransmissions of any individual packet must be supported before FP performance is impacted. A single BDP sized buffer allows the link to stay fully utilized as long as the receiver is processing the packets quickly enough and there are no errors. If a packet loss occurs, the missing packet (hole) will progress to the left edge of the receive window (as the receiver processes data) and the buffer will begin to fill. After one RTT the receive buffer will be full and transmission of new packets must be halted while the lost packet is retransmitted. Effectively, a single RTT pause is inserted for any packet lost once. If a double BDP buffer is used, then virtually any number of packets can be lost once with the FP and the connection will still send new data at full speed if available.

The following calculations are for example only, but an attempt has been made to make this example as efficient as possible. The two values given are for CRA and VBDC, respectively. It is important to note that one of ordinary skill in the art would recognize other variations, modifications, and alternatives

GW PEP TX BUFFER = 60Mbps / 8 bits per byte * 105% * 1.1s to 1.9s = 8.7M bytes to 14.9M bytes

GW PEP RX BUFFER = 48Mbps / 8 bits per byte * 200% * 1.1s to 1.9s = 13.2M bytes to 22.8M bytes

Terminal PEP TX BUFFER = 2Mbps / 8 bits per byte * 105% * 1.1s to 1.9s = 288K bytes to 499K bytes

Terminal PEP RX BUFFER = 8Mbps / 8 bits per byte * 200% * 1.1s to 1.9s = 2.2M bytes to 3.8M bytes

The description above has dealt with the use of the "buffer utilization" test to share resources between competing TCP/IP connections in the terrestrial to PEP direction. However, the same technique is used to manage the flow of resources between the PEP and the terrestrial world using a Receive Buffer. This allows for the fair sharing of resources between connections, the distribution of free resources if one or more connections are not employing their allocation and the prevention of slow connections dominating the link. (In simulations it was found that if connections are not limited in their allocation, then slow TCP/IP connections would tend to dominate the link). The scheme works using the same rules but in reverse, this time the intention being to decide whether or not to accept incoming FP packets from the satellite. FP packets that are allowed to be stored in the FP Receiver buffer are cleared by a TCP acknowledgement. In much the same way, as TCP operates, a window mechanism is used from the receiving PEP to the transmitting PEP to indicate the status of each connection's buffer space and thus prevent over-flow.

The task for an overall flow control mechanism applicable for a large bandwidth-delay product environment with changing bandwidth and latency, such as a satellite system, is to achieve the following goals:

1. Maximize the flow of data from the sender to the satellite network without under-supplying or over-supplying the intermediate node (i.e. the terminal that requests and receives bandwidth capacity).
2. Share satellite and terrestrial bandwidth within the overall flow so as to ensure fairness while allowing unused capacity to be employed by hungry connections and preventing slow connections from dominating the link capacity.
3. Prevent the overflow of the receiving end PEPs buffer associated per connection.

As used in this section, transmission of data means the passing of a packet from the TCP* layer to the FP layer, the re-transmission of a FP data packet or an acknowledgment packet signaling the successful arrival of data.

In general (with the exception of re-transmissions and ACKs) packets are pushed, upon arrival from the terrestrial world, from the TCP* **207** to the FP layer **208** as shown on Fig. 2, and then sent via the satellite with a copy being stored in the FP Transmit buffer. Before packets can be pushed (or re-transmitted / acknowledged) they have to pass three tests at PEP1, the transmitter PEP over satellite where the flow is client to server: a local (transmit) buffer space test (Distribution of Satellite Resources amongst TCP/IP connections at the PEP), a PEP-to-Intermediate Node Communication test, and a sufficient buffer space test. Sufficient buffer space at the receiving PEP is signalled using a conventional window strategy in packets (data or ACKs) flowing from the receiving PEP to the sending PEP.

When space becomes available (either because of a changed window parameter or an ACK removing a packet from the FP Tx Buffer), the PEP pulls a packet from the applicable TCP* buffer **207**, providing that the three flow control rules allow this. This technique, in combination with the above rules and large initial windows, allows nearly 100% usage of available capacity, while managing the distribution of bandwidth in a variable bandwidth / latency system. Natural back pressure from the TCP* buffers slows down the terrestrial TCP/IP connection flow.

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.

CLAIMS:

1. In a distributed connection splitting system, which comprises at least two intermediate nodes, a method for managing data flow between the intermediate nodes and through an intermediate device, comprising the steps of:

obtaining status data of the intermediate device; and

managing the flow of data between the intermediate nodes and the intermediate device by determining the status of an intermediate device queue.

2. The method of claim 1, wherein the amount of at least one of data and packet space in the intermediate device is used to determine the quantity of at least one of data and packets which should be sent to the intermediate device.

3. The method of claim 1, wherein the intermediate nodes send sufficient data to at-least partially fill the intermediate device queue to a pre-defined level, and after a pre-determined delay to allow data to be transmitted, checks the status of the intermediate device and then re-fills the intermediate device queue to a previous level.

4. The method of claim 1, wherein the intermediate nodes send regular SNMP get messages to an intermediate device MIB or the intermediate device sends SNMP traps when a queue reaches a pre-detemined level to indicate the current queue status.

5. The method of claim 1, wherein the flow of data from the intermediate nodes to the intermediate device is more accurate by the intermediate nodes having knowledge of current and future bandwidth information from the intermediate device.

6. The method of claim 5, wherein the intermediate nodes and the intermediate device are co-located in the same device with some form of direct linkage between the allocated bandwidth / queue and the intermediate nodes.

7. A method of fairly distributing bandwidth among connections in a communications system, comprising the step of sharing of transmission storage amongst a number N connections.

8. The method according to claim 7, wherein the step of sharing further includes the steps of:

calculating an amount of capacity given to a connection before allowance for any applicable intermediate node to Intermediate device flow is X/N, where X is the maximum bandwidth-delay product of a communication and N is the number of connections;

determining if a sum of all the actual capacity usage by each individual connection N is less than X, wherein the available capacity is shared by any connections wishing to exceed an allocation; and determining, if new connections arrive, whether they are guaranteed (X/N) capacity even if the result is a temporarily over-supply of capacity to a link resulting in buffering at an output queue of the intermediate node.

9. A method for managing the flow of data from an intermediate node to an intermediate device so that the transmission of data depends upon: sufficient transmission buffer space in the sending intermediate node; sufficient space in the receiver window of the receiving intermediate node; and sufficient capacity / buffer space at one of the intermediate devices.

10. The method of claim 9, wherein the failure of a test causes data to be stored in an applicable input or output queue within the intermediate node thereby creating natural back pressure to a TCP sender in the regular window updates sent from the intermediate node to the TCP sender.

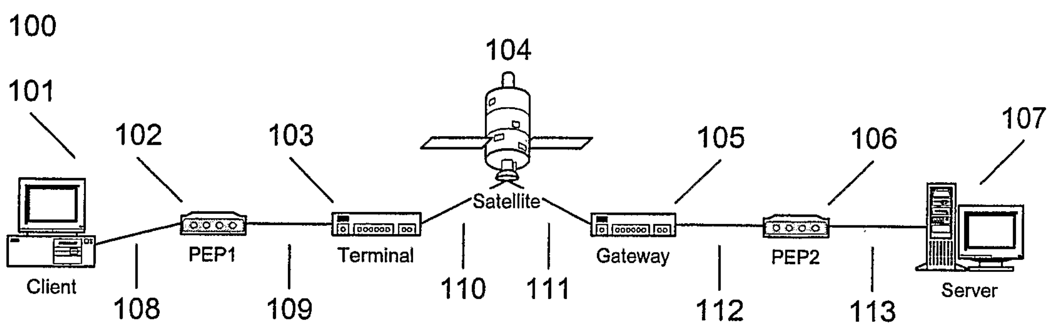
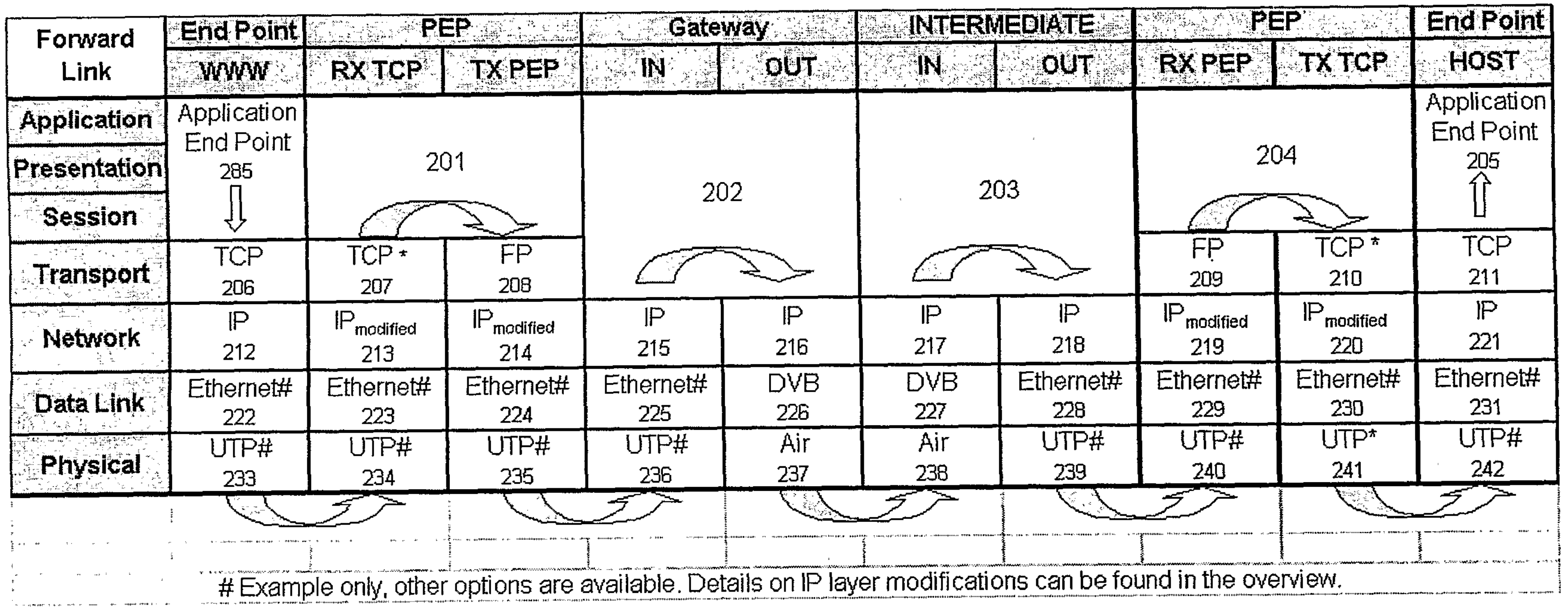

100
101
102
103
104
105
106
107
Client
PEP1
Terminal
Satellite
Gateway
PEP2
Server
108
109
110
111
112
113

Figure 1



Forward Link	End Point	PEP		Gateway		INTERMEDIATE		PEP		End Point
	WWW	RX TCP	TX PEP	IN	OUT	IN	OUT	RX PEP	TX TCP	HOST
Application	Application End Point 285 ⇩	201		202		203		204		Application End Point 205 ⇧
Presentation										
Session										
Transport	TCP 206	TCP * 207	FP 208					FP 209	TCP * 210	TCP 211
Network	IP 212	$IP_{modified}$ 213	$IP_{modified}$ 214	IP 215	IP 216	IP 217	IP 218	$IP_{modified}$ 219	$IP_{modified}$ 220	IP 221
Data Link	Ethernet# 222	Ethernet# 223	Ethernet# 224	Ethernet# 225	DVB 226	DVB 227	Ethernet# 228	Ethernet# 229	Ethernet# 230	Ethernet# 231
Physical	UTP# 233	UTP# 234	UTP# 235	UTP# 236	Air 237	Air 238	UTP# 239	UTP# 240	UTP* 241	UTP# 242

Example only, other options are available. Details on IP layer modifications can be found in the overview.

Figure 2

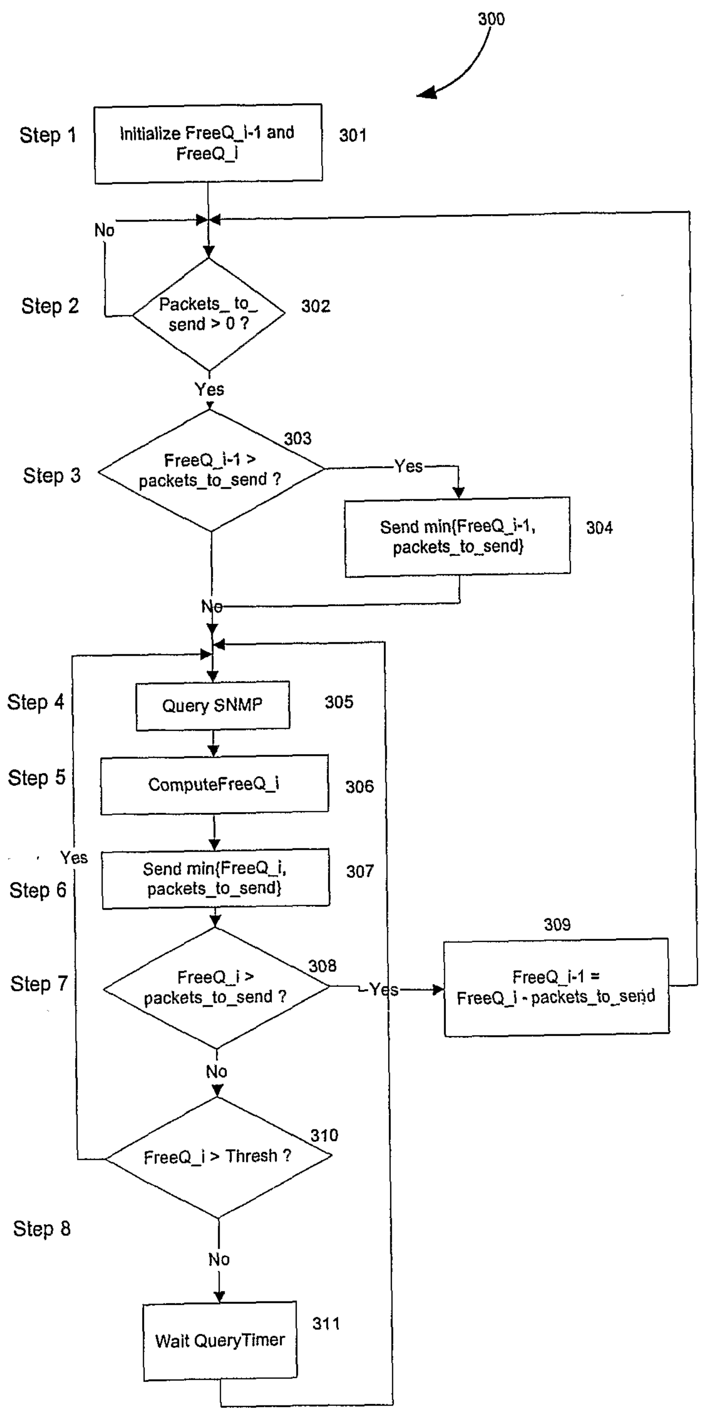

300
Step 1
Initialize FreeQ_i-1 and FreeQ_i
301
No
Step 2
Packets_ to_ send > 0 ?
302
Yes
Step 3
FreeQ_i-1 > packets_to_send ?
303
Yes
Send min{FreeQ_i-1, packets_to_send}
304
No
Step 4
Query SNMP
305
Step 5
ComputeFreeQ_i
306
Yes
Step 6
Send min{FreeQ_i, packets_to_send}
307
Step 7
FreeQ_i > packets_to_send ?
308
Yes
309
FreeQ_i-1 = FreeQ_i - packets_to_send
No
FreeQ_i > Thresh ?
310
Step 8
No
Wait QueryTimer
311

Figure 3