Draft
Frequently Asked Questions (FAQ) for Performance Enhancing Proxies (PEPS)
Hints on How to Configure PEPs
August 01, 2005
Patrick Feighery[1][2][3]

Topics

[1] Approved for Public Release; Distribution Unlimited

[3] The research described in this paper was carried out for the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration

Background on PEPs

Back in the late 1990s there were considerable discussions of the performance issues associated with TCP based applications traversing satellite environments. One popular mechanism to address performance concerns is a called a Performance Enhancing Proxy (PEP) also known as a transport layer proxy or gateway. This technique splits the TCP connection into three separate connections: 1) standard TCP connection between the local application and the local PEP; 2) an advanced protocol to transmit data across the satellite or otherwise challenged link; 3) standard TCP connection between the peer PEP and the peer application. This inter-PEP protocol is tuned and optimized to the characteristics of the satellite link. The advantage of using this technique is the end systems and applications are not modified and any way but still receive the performance enhancing benefits of the PEP. See following picture which describes the operation of a PEP.

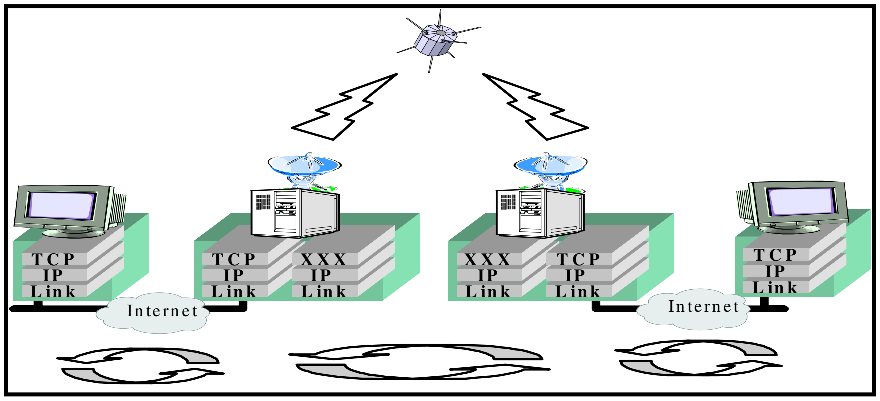


Figure 1 Sample PEP Scenario

How PEPS Improve Performance

PEPs, operating at the transport layer, increase the performance of TCP based applications where native performance suffers due to characteristics of a link or subnetwork in the path. These PEPs do not modify the application protocol, thus applications operate end-to-end. These PEP techniques are totally transparent to the applications. Technically, these gateways perform a technique called spoofing in which they intercept the TCP connection in the middle and terminate that connection as if the gateway were the intended destination. PEP, typically bracketing the satellite network, split a single TCP connection into three separate connections. These gateways communicate via standard TCP when talking to each of the end-systems and a third connection using an optimized rate based protocol is used when transferring the data between them. This technique allows one to isolate the properties of satellite networks that degrade performance from manifesting themselves to TCP. For example, corruption loss on the satellite network does not cause the transmission rate to be cut in half, and congestion loss on the terrestrial network does not cause data to be retransmitted consuming satellite bandwidth that may be a scarce resource. This also allows the deployment of protocols that can be tuned to match the characteristics of the satellite link without affecting the end-systems. Now the default TCP parameters on most end-systems are now applicable for the terrestrial environment in which they operate.

Placement of PEPs within a Network

In general PEPs should be placed as close to the satellite or otherwise challenged link as possible. As indicated above, the PEP operate best when they are the only entry point into a reserved capacity and therefore in a best case situation the PEP processes all traffic traversing satellite and only the traffic traversing the link. If the PEP needs to be placed further from the satellite resource, but still processes all data traversing the satellite resource then, the PEP needs the ability to classify which traffic will traverse the constrained resource.

Network Configuration #1 – Reserved Capacity Guaranteed Between PEPs

A rate-based emission strategy can be used by the PEP because it is typically in control of the inter-gateway network and is provided a guaranteed bandwidth capacity. Since the PEPs are the only entry point in to the reserved capacity, PEPs can ensure that network congestion is not possible. Since network congestion not possible for inter-gateway traffic only flow control and a retransmission strategy need to be deployed. This is the preferred topology for PEPs.

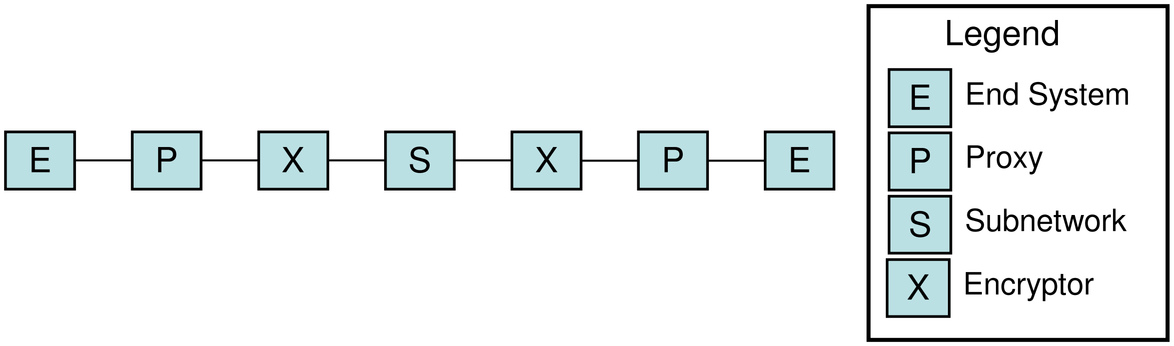
Network Configuration #2 – Unreserved Capacity between PEPs

When there is a unreserved, unknown, or uncontrolled bandwidth between the PEPs, the possibility of network congestion is always possible. Therefore, the PEP should run some from of congestion control packet emission policy. Whether this policy is standard VJ congestion control algorithm of the Vegas congestion control algorithm depends on the various in the latency of the path between the two paths. Vegas should typically not be used if there is any real variance of round trip times, since Vegas assumes the variation if round trip times is due to queueing delay and thus Vegas will adjust it rate according.

Interaction with Network Encryptors

Because the PEPs need to terminate TCP connections to/from the source/destination, they must be placed in the network path between the end hosts and any network encryptor device (if present). If the PEP is placed in the network so that it sees traffic before network layer encryptors (e.g., IPSEC, TACLanes, etc.), it has the ability to terminate the TCP connections and improve performance. However the PEP needs to be aware of the encryptor in the network, in particular any overhead that is added by the encryptor. For example some encryptors operate by encrypting the entire input IP packet and encapsulating it inside another packet. The PEP must be aware of this for the following reasons.

- **Fragmentation Avoidance** IP fragmentation occurs when an IP packet is too large for the Maximum Transmission Unit (MTU) of the link to be traversed. When this occurs the IP packet is divided into smaller packets to traverse the link. These smaller segments are then reassembled at the end point, which in this case is the peer PEP. Fragmentation in general is undesirable for three reasons. First, the overhead involved in the fragmented packets increases the number of octets traversing the link. Second, if any of the fragmented packets is lost (either due to congestion or corruption) then the entire packet must be resent not just the lost fragment. Third, reassembly from the receiver's perspective requires addition queueing and timers. Therefore fragmentation should be avoided. PEPs have the ability to decrease the Maximum Segment Size (MSS) to avoid any down stream fragmentation.
- **Overhead Accountability** One common performance enhancement technique PEPs use is to disable congestion control and to emit data at or just below the line rate of the challenged resource. When downstream devices add overhead, , the PEP may overdrive the network and cause self congestion. With encryptors, the additional overhead associated with the IP encapsulation plus any encryption overhead needs to be accounted for. Additionally, some encryptors may prevent traffic analysis by padding all packets to their MTU size. The PEP must be made aware and must account for the additional overhead incurred by the encryptor.



Sample Topology for Placement of Network Encryptors

Interaction with GRE and other Tunneling Techniques

As with network encryption devices, PEPs need to be aware of any network layer overhead that can augment size of the packets transmitted by the PEP. See Interaction with Network Encryptors problems associated with ‘Fragmentation Avoidance’ and ‘Overhead Accountability.’ This also includes interaction with an MPLS cloud, since the MPLS ingress point adds an additional 4 octets of overhead. Moreover, PEPs and most other Middleware boxes typically can not be inserted into an MPLS cloud, because the ethernet type of not of IP, but rather of MPLS which may cause confusion in these devices. When designing network architectures, PEPs should not be placed within a tunnel. From a theoretical perspective, it could be possible (assuming encryption is not present) for a PEP to walk down the levels of encapsulation to reach the transport layer header. When the PEP transmits packets it would have to be aware of the tunnel and recreate the tunneling structure. Most PEPs that I am aware of do not support this feature.

Interaction with End Systems

The SCPS PEP uses a superset of the TCP protocol and thus uses TCP options that are properly registered with IANA to negotiate the SCPS features. The TCP standard states that if an option is well formed but not implemented that option should be ignored. Therefore if a TCP protocol receives a TCP SYN with the SCPS option enabled, but does not understand the SCPS options it must ignore that option when responding with a SYN-ACK. Thus the SCPS features will not be used. Some operating systems, Microsoft Window variations in particular, unfortunately improperly handle TCP options not understood by their implementation. In this case, these implementations will not establish a connection if the TCP SYN contain the SCPS option. To work around this bug, PEPs are typically configured to not offer the SCPS option on the LAN side. If one PEP goes down, however, the peer PEP will be talking directly to the end system. When this occurs the PEP should try a few times to establish the connection with the SCPS option set on the SYN and if that fails, it should try to establish a connection without the SCPS option present. In addition to this the PEP should cache this information locally so that next time it attempts to establish a connection, it will not use the SCPS options if it was previously not accepted. Therefore for a period of time all connection to that destination will not include the SCPS option to speed up the connection establishment time.

Interaction with Microsoft’s Implementation of TCP

To date, most if not all Microsoft implementations of TCP have a software bug in the processing of TCP options that can prohibit some interactions with a SCPS based PEP.

Most implementations of TCP are developed to be flexible as TCP matures. In particular when new TCP options are developed implementations must be able to handle it - (e.g., window scaling, SACK, etc.) By this I mean to properly negotiate the usr or non use of that option. According to RFC 1122 entitled "Requirements for Internet Hosts -- Communication Layers"

4.2.2.5 TCP Options: RFC-793 Section 3.1

> A TCP MUST be able to receive a TCP option in any segment. A TCP MUST ignore without error any TCP option it does not implement, assuming that the option has a length field (all TCP options defined in the future will have length fields). TCP MUST be prepared to handle an illegal option length (e.g., zero) without crashing; a suggested procedure is to reset the connection and log the reason.

Therefore if a TCP implementation receives a properly formatted TCP option that it does not implement it MUST simply ignore that option.

It appears that for options that Microsoft's implementation of TCP understands, it will properly negotiate the use or non-use of that option. For example, it does properly negotiate the use of timestamps and window scaling defined in RFC 1323. However it appears that most Microsoft implementations of TCP will NOT accept TCP connections if they contains TCP options that the OS does not understand. This is NOT in accordance with the Internets 'Requirements of Internet hosts – Communication Layers' as stated above. Not only is the SCPS options well formed but it is also registered with IANA. See http://www.iana.org/assignments/tcp-parameters - KIND 20.

To overcome the problem in Microsoft's TCP implementation, SCPS based PEPs need to be configured to optionally not include the SCPS TCP option when communicating to systems on the LAN side.

Interaction with TDMA, DAMA and Other Dynamic Bandwidth Environments

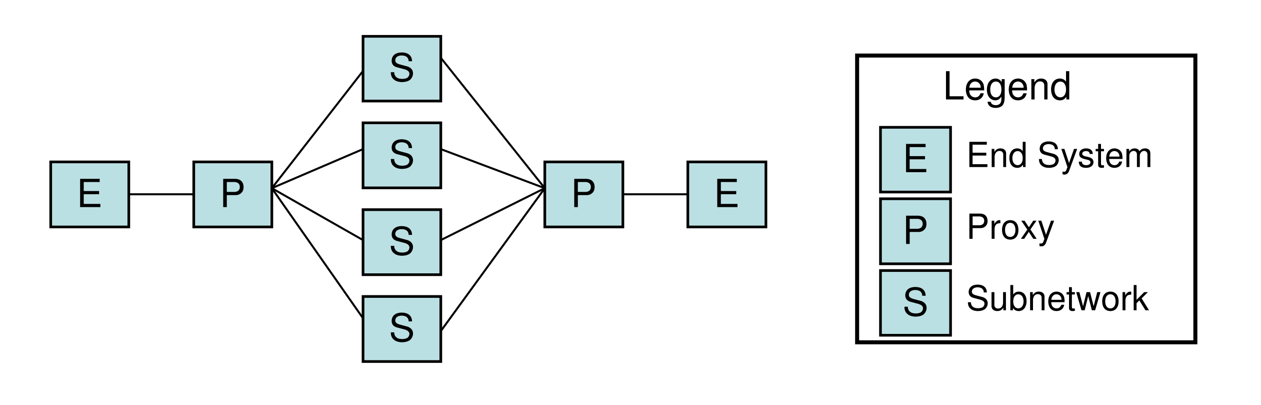
In some environments, in particular DAMA or TDMA based environments; the bandwidth of a network is dynamically share among the number of active nodes based upon varies parameters (e.g., queue size.) In addition to sharing the link, the mechanisms required to schedule and fill 'slots' will also effect the overall bandwidth presented to each node. This environment breaks the typical deployment of PEP where the bandwidth of the system from the perspective of the PEPs is assumed to be static and known apriori. A couple of techniques could be used to address this problem.

1. If the bandwidth of the system does not change frequently and a feedback loop from the modem to the PEP exists, then the PEP could be notified about the changes in capacity. These changes could either be in the form of queue availability at the constrained resource (flow control between the modem and the PEP) or current bandwidth of the constrained resource. It should be noted that if encryption devices are present between the PEP and the challenged resource, then this may not be possible.

2. Congestion control can be used by the PEP. For example, either TCP's Van Jacobson congestion control algorithm – which is a reactive congestion control algorithm or SCPS Vegas's congestion control algorithm – which is a proactive congestion control algorithm.
3. A combination of both a rate based system and a congestion control algorithm could be deployed. Rate control would provide a minimum guarantee and a maximum possible value and a congestion control algorithm such as Vegas could be used to find an appropriate operating point between these two values.

Interaction with Striping Data across Multiple Subnetworks

The concept of striping involves sending data across multiple subnetworks to increase the overall throughput of a network. When striping occurs, typically a node on each side of the multiple subnetworks ensures that all data from a single connection will only traverse a single subnetwork. This decreases the probably of extreme packet misordering which has a tremendous effect on overall throughput. In some environments, data from a single TCP connection may be striped over multiple subnetworks (e.g., multiple low bandwidth RF links) to improve the application performance. Striping a single connection, depending on the characteristics of the individual links, will result in data will arrive (sometimes massively) out of order. This may confuse and affect the retransmission policy of the PEP. If this occurs addition logic needs to be added to the PEP that holds back on retransmitting packets that may simply be arriving late. This additional logic may occur at the receiver requesting a retransmission or at the transmitter retransmitting data to fill the hole.



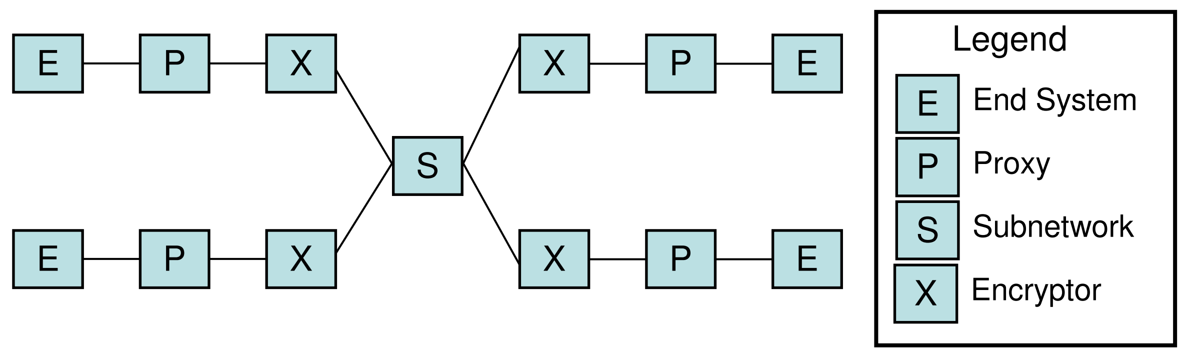
Sample Topology for Striping Data across Multiple Subnetworks

Interaction with Multiple Secure Enclaves

A single secure site is able to use an insecure commercial satellite resource for its secure data communication needs and is still able to use a gateway for performance improvements. A following step might allow multiple sites with different security classifications to share the common satellite resource. Thus each secure site would have a pair of gateways and encryptors at each site for secure data transmission. However since PEPs typically rely strictly on rate control as means to emit data through a network, there is a possibility, for congestion based loss to occur across the satellite network. Therefore the following options may occur:

1. Each pair of gateways is configured to rate-control traffic onto the channel at the full bandwidth of the satellite link. Network congestion will occur.
2. Each pair of gateways is configured *a priori* to rate-control traffic onto the channel at the prescribed percentage of the satellite's bandwidth. Network congestion will not occur. However, when only a single pair of gateways is actively communicating over the link, performance is non-optimal.

Another option would allow the gateway to provide some indirect feedback to the other gateways; allowing some form of coordination. This approach is to augment the pure rate control mechanism with the Vegas congestion control algorithm. Essentially the gateways will use the Vegas algorithm for emitting data on the satellite network, but a rate control mechanism will also be used as a stop-gap to ensure a single set of gateways will not cause congestion loss.



Sample Topology for Multiple Secure Enclaves

Transport Layer - Send and Receive Window

Flow control is used to ensure that the sender does not overwhelm the receivers' buffers. In actuality, both the TCP sender and receiver have buffers. The sender uses the buffers as a retransmission buffer (i.e., to store data for possible retransmission) while the receiver uses the buffers as an out-of-sequence queue (i.e., to save packets received by the sender that are not in sequence to avoid the sender from possibly retransmitting them.) Therefore if the buffer sizes are different, only the minimum buffer size of the two is used for flow control. When looking at flow control within TCP, it is quite useful to examine the Bandwidth Delay Product (BDP). The BDP provides a relationship between the bandwidth of a network and the round trip delay though the network to establish the minimum TCP window size required under ideal conditions (i.e., no segment loss or network queuing causing the round trip time to increase). Therefore if the bandwidth of the network (i.e., the slowest link in the network) is 1 Mbps and the round trip time through the entire network is 600 milliseconds (which are typically for satellites operating in a geo-synchronous orbit) the minimum TCP buffer size needs to be 75,000 bytes. In general the transport layer window sizes for the WAN side connection should be set somewhere between one and two times the bandwidth delay product. A rule of thumb without accounting for other information, the send and receive windows should be set for twice the bandwidth delay product. If the window size is greater than 65536 bytes then window scaling must be enabled. The default transport layer window sizes for most operating systems range from 8K to 32K) and therefore they should be set at 32K for the LAN side connection.

Transport Layer - Data Rate (Rate control)

Rate control controls the maximum data rate that data will be emitted from the PEP. This allows the PEP to emit data at or just below the line rate of the challenged resource. In general, the value should be set to the minimum of the maximum rates on the WAN side. It should be noted that there are other factors that may further limit the clocking out of data at less than data rate. For example

For control via the transport layer send and receive windows may further limit the traffic.

A combination of a congestion control technique (e.g., VJ or Vegas) may further limit how data is emitted.

It is important to note that when downstream devices cause an increase of packet size, the PEP may overdrive the network and cause self congestion. With encryptors, the additional overhead associated with the IP encapsulation plus any encryption overhead needs to be accounted for. Additionally, some encryptors may prevent traffic analysis by padding all packets to their MTU size. The PEP must be made aware and must account for the additional overhead incurred by the encryptor.

Transport Layer – Snack Option

The Selective Negative Acknowledgment (SNACK) option allows the receiver to inform the SCPS-TP sender about one or more holes in the receiver’s out-of-sequence queue. Without a selective acknowledgment, TCP can use the ACK number to identify at most a single hole in the receiver’s buffer. Using its simple cumulative acknowledgment and the Fast Retransmit algorithm, TCP can recover efficiently from a single loss per window. However, because new data must be received for the receiver to advance the ACK number, TCP requires a minimum of one RTT to signal *each* additional hole in the out-of-sequence queue. The SNACK option, which is carried on an acknowledgment segment, identifies multiple holes in the sequence space buffered by the receiver. By providing more information about lost segments more quickly, the SNACK option can hasten recovery and prevent the sender from becoming window-limited. This allows the pipe to drain while waiting to learn about lost segments. The ability to transmit continuously in the presence of packet loss is especially important when loss is caused by corruption rather than congestion. In such a case, when it has been deemed that it is appropriate to disable congestion control as the response to loss, SNACK is of particular benefit in keeping the pipe full and allowing transmission to continue at full throttle while recovering from losses. In general, this option should be enabled, unless special network characteristics would warrant it non-use.

The following section describes the differences between SNACK and SACK. SACK is used by the receiver to tell the sender when there is an out of sequence queue (i.e., missing packets.) A SACK block indicated which packets have been received properly. Also explicitly stated in the original RFC that the sender may not use this information to assume which packets have not been received properly, most modern implementation make that assumption. Rather, the RFC states the sender may use this information to determine which packets have left the network and adjust the congestion window appropriately. Given the new interpretation of SACK, SACK and SNACK actually perform similar functions. SNACK however is not bit efficient and is able to more easily indicate multiple holes in the out of sequence queue.

Transport Layer – Retransmission TimeOut (RTO) Parameters

TCP provides a reliable transport layer. One of the ways it provides reliability is for each end to acknowledge the data it receives from the other end. But data segments and acknowledgments can get lost. TCP handles this by setting a timeout when it sends data, and if the data isn't acknowledged when the timeout expires, it retransmits the data. A critical element of any implementation is the timeout and retransmission strategy. How is the timeout interval determined, and how frequently does a retransmission occur. There exist three RTO parameters which control how frequently retransmission may occur. First there is the Initial Retransmission TimeOut (IRTO). Next there is the Minimum Retransmission TimeOut (MIN_RTO), which provides a minimum value or floor on how soon the retransmission timeout may occur. Finally, there is the Maximum Retransmission Timeout (MAX_RTO) which provides a maximum value or ceiling between successive timeouts.

These values may be changes based on knowledge of the challenged link. For example, due to certain characteristics of the challenged link (scheduling of the resource), the minimum value of a round trip time (e.g., ping) may be 5 seconds. Therefore you may want to set the Initial RTO to be 10 seconds. The Minimum RTO value may be set to 6 seconds, since the Round Trip Time must be greater than 5 seconds. The Maximum RTO may also be changed based on knowledge of the resource as well.

Transport Layer – Changing Delayed Ack Frequency of the Receiver

In environments involving high asymmetry where the bandwidth of the forward channel (data flow) is much greater than the bandwidth for the reverse channel (TCP ack flow), performance will suffer because the sending rate will be limited. In highly asymmetric environments, communication may be ack channel limited. This means the transmitter may not emit packets until acknowledgments have been received. Reducing the frequency the receiver generated acknowledgement may reduce the congestion on the ack channel, thus allowing packets to be emitted quicker. This typically does not occur until the bandwidth ratio becomes greater than 50:1.

Transport Layer – Time Stamps Option

Timestamps, which are defined in RFC 1323 increase the number of round trip time samples that are used in the calculation for a Retransmission TimeOut (RTO) timer value. The timestamp option lets the sender place a timestamp value in every segment. The receiver reflects this value in the acknowledgment, allowing the sender to calculate an RTT for each received ACK. Non-timestamp enabled implementations only measure one RTT per window, which is acceptable for windows containing a few segments. Larger window sizes, however, require better RTT calculations. The specification of this option is negotiated during connection establishment phase. For timestamps the sender places a 32-bit value in the first field, and the receiver echoes this back in the reply field. TCP headers containing this option will increase by 12 octets – typically from the 20 octets to 32 octets.

Timestamps are typically enabled unless the additional 12 octets of overhead are determined to be too great for the benefits.

Transport Layer – Compression Option

Standard Van-Jacobson header compression is extremely efficient in reducing the size of a TCP/IP header. Part of its efficiency comes from delta-encoding, whereby values in the fields of the N+1st TCP header are sent not as absolutes, but are sent in the compressed header as their difference from their values in the Nth header. A drawback of this approach is that the loss of a single packet generally incurs a retransmission timeout in order to resynchonize transmitter and receiver. Some studies have been done where, when a packet with a delta-encoded compressed header cannot be decompressed at the receiver, the receiver guesses that a packet has been lost, assumes values for the fields in the lost packet based on the history of the data stream, and tries to uncompress the received packet, using these guesses. This works well when the data stream is sufficiently predictable that adequate guesses can be made about missed packets (specifically the size of the data portion of the packet).

The SCPS transport layer implements a different form of header compression that is loss-tolerant. That is, the receiver can decompress any correctly received packet without having received the previous packet in the sequence. We call this compression scheme Loss-Tolerant TCP header compression. Though not as efficient as Van Jacobson header compression, it does not use delta-encoding and hence is robust against the loss of a single packet. (Of course, the lost packet will still have to be retransmitted. The issue here is that packets received out-of-sequence after the lost packet can be decompressed and stored in an out-of-sequence buffer). The SCPS compression option is not a link layer compression technique, thus the IP packet is left untouched, thus a compress packer may be routed through the network. This should be used on low bandwidth links.

Techniques to Augment PEP Services

Data Compression. Depending on the nature of the traffic traversing the challenged environments, data compression can be used to significantly reduce the traffic traversing the link. This essentially provides additional capacity for other data to consume the constrained resource. If the PEP is using some form of rate control to emit data across the network, then the rate control mechanism must be aware of the compression functionality and the number of octets actually traversing the link.

HTTP Caching. Techniques to prevent the consumption of the challenged resource are also very useful. An HTTP cache, a squid; for example, on the client side can store data responses for future requests. On subsequent requests the http cache can service and respond without consuming the constrained resources. In addition to HTTP caching, other applications, FTP and DNS for example can also take advantage of caching. These functions are typically performed before the data reaches the PEP device.

Documentation of TCP Performance Issues and PEPs within the IETF

Within the Internet community, the Internet Engineering Task Force (IETF) is the main body that develops standards and documents various networking aspects concerning the Internet's protocols. Within the IETF, various working groups are formed to address a variety of problems and issues. These working groups as well as individual document their work in an iterative process known as 'Internet-Drafts' or IDs for short. IDs have a lifetime of 6 months then they expire. When IDs become sufficiently mature, the may be elevated to 'Request for Comment' or RFC status. RFC do not expire, but may be superceded by newer RFCs.

Within the IETF, the following RFCs have been written about TCP performance over satellite links and PEPs.

RFC 2448 - Enhancing TCP Over Satellite Channels using Standard Mechanisms"
Dated January 1999
URL ftp://ftp.isi.edu/in-notes/rfc2488.txt

This RFC documents some of the TCP parameter that can be tuned to increase the performance of TCP application traversing satellite and other challenged links. It should be noted these 'tunings' require administrative or root access to possibly both endpoints involved in the data transfer. Unfortunately, this assumption is only valid in limited scenarios.

RFC 2769 - Ongoing TCP Research Related to Satellites
Dated February 2000
URL ftp://ftp.isi.edu/in-notes/rfc2760.txt

This RFC documents the status of a variety on work which may allow TCP enhancements that may allow TCP to better utilize the available bandwidth provided by networks containing satellite links. At the time of publishing, mechanisms outlined have not been judged to be mature enough to be recommended by the IETF. The goal of this document is to educate researchers as to the current work and progress being done in TCP research related to satellite networks.

RFC 3135 - Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations
Dated June 2001
URL ftp://ftp.isi.edu/in-notes/rfc3135.txt

This document is a survey of Performance Enhancing Proxies (PEPs) often employed to improve degraded TCP performance caused by characteristics of specific link environments, for example, in satellite, wireless WAN, and wireless LAN environments. Different types of Performance Enhancing Proxies are described as well as the mechanisms used to improve performance. Emphasis is put on proxies operating with TCP.

RFC 3155 - End-to-end Performance Implications of Links with Errors
Date August 2001
URL ftp://ftp.isi.edu/in-notes/rfc3155.txt

This document discusses the specific TCP mechanisms that are problematic in environments with high uncorrected error rates, and discusses what can be done to mitigate the problems without introducing intermediate devices into the connection.