Network Working Group
Request for Comments: 3135
Category: Informational

J. Border
Hughes Network Systems
M. Kojo
University of Helsinki
J. Griner
NASA Glenn Research Center
G. Montenegro
Sun Microsystems, Inc.
Z. Shelby
University of Oulu

June 2001

Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations

Status of this Memo

Copyright Notice

Abstract

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. In addition, motivations for their development and use are described along with some of the consequences of using them, especially in the context of the Internet.

Table of Contents

1. Introduction 3
2. Types of Performance Enhancing Proxies 4
2.1 Layering 4
2.1.1 Transport Layer PEPs 5
2.1.2 Application Layer PEPs 5
2.2 Distribution 6
2.3 Implementation Symmetry 6
2.4 Split Connections 7

2.5 Transparency 8
3. PEP Mechanisms 9
3.1 TCP ACK Handling 9
3.1.1 TCP ACK Spacing 9
3.1.2 Local TCP Acknowledgements 9
3.1.3 Local TCP Retransmissions 9
3.1.4 TCP ACK Filtering and Reconstruction 10
3.2 Tunneling 10
3.3 Compression 10
3.4 Handling Periods of Link Disconnection with TCP 11
3.5 Priority-based Multiplexing 12
3.6 Protocol Booster Mechanisms 13
4. Implications of Using PEPs 14
4.1 The End-to-end Argument 14
4.1.1 Security 14
4.1.1.1 Security Implications 15
4.1.1.2 Security Implication Mitigations 16
4.1.1.3 Security Research Related to PEPs 16
4.1.2 Fate Sharing 16
4.1.3 End-to-end Reliability 17
4.1.4 End-to-end Failure Diagnostics 19
4.2 Asymmetric Routing 19
4.3 Mobile Hosts 20
4.4 Scalability 20
4.5 Other Implications of Using PEPs 21
5. PEP Environment Examples 21
5.1 VSAT Environments 21
5.1.1 VSAT Network Characteristics 22
5.1.2 VSAT Network PEP Implementations 23
5.1.3 VSAT Network PEP Motivation 24
5.2 W-WAN Environments 25
5.2.1 W-WAN Network Characteristics 25
5.2.2 W-WAN PEP Implementations 26
5.2.2.1 Mowgli System 26
5.2.2.2 Wireless Application Protocol (WAP) 28
5.2.3 W-WAN PEP Motivation 29
5.3 W-LAN Environments 30
5.3.1 W-LAN Network Characteristics 30
5.3.2 W-LAN PEP Implementations: Snoop 31
5.3.3 W-LAN PEP Motivation 33
6. Security Considerations 34
7. IANA Considerations 34
8. Acknowledgements 34
9. References 35
10. Authors' Addresses 39
Appendix A - PEP Terminology Summary 41
Full Copyright Statement 45

1. Introduction

The Transmission Control Protocol [RFC0793] (TCP) is used as the transport layer protocol by many Internet and intranet applications. However, in certain environments, TCP and other higher layer protocol performance is limited by the link characteristics of the environment.

This document is a survey of Performance Enhancing Proxy (PEP) performance migitigation techniques. A PEP is used to improve the performance of the Internet protocols on network paths where native performance suffers due to characteristics of a link or subnetwork on the path. This document is informational and does not make recommendations about using PEPs or not using them. Distinct standards track recommendations for the performance mitigation of TCP over links with high error rates, links with low bandwidth, and so on, have been developed or are in development by the Performance Implications of Link Characteristics WG (PILC) [PILCWEB].

Link design choices may have a significant influence on the performance and efficiency of the Internet. However, not all link characteristics, for example, high latency, can be compensated for by choices in the link layer design. And, the cost of compensating for some link characteristics may be prohibitive for some technologies. The techniques surveyed here are applied to existing link technologies. When new link technologies are designed, they should be designed so that these techniques are not required, if at all possible.

This document does not advocate the use of PEPs in any general case. On the contrary, we believe that the end-to-end principle in designing Internet protocols should be retained as the prevailing approach and PEPs should be used only in specific environments and circumstances where end-to-end mechanisms providing similar performance enhancements are not available. In any environment where one might consider employing a PEP for improved performance, an end user (or, in some cases, the responsible network administrator) should be aware of the PEP and the choice of employing PEP functionality should be under the control of the end user, especially if employing the PEP would interfere with end-to-end usage of IP layer security mechanisms or otherwise have undesirable implications in some circumstances. This would allow the user to choose end-to-end IP at all times but, of course, without the performance enhancements that employing the PEP may yield.

This survey does not make recommendations, for or against, with respect to using PEPs. Standards track recommendations have been or are being developed within the IETF for individual link

characteristics, e.g., links with high error rates, links with low bandwidth, links with asymmetric bandwidth, etc., by the Performance Implications of Link Characteristics WG (PILC) [PILCWEB].

The remainder of this document is organized as follows. Section 2 provides an overview of different kinds of PEP implementations.

Section 3 discusses some of the mechanisms which PEPs may employ in order to improve performance. Section 4 discusses some of the implications with respect to using PEPs, especially in the context of the global Internet. Finally, Section 5 discusses some example environments where PEPs are used: satellite very small aperture terminal (VSAT) environments, mobile wireless WAN (W-WAN) environments and wireless LAN (W-LAN) environments. A summary of PEP terminology is included in an appendix (Appendix A).

2. Types of Performance Enhancing Proxies

There are many types of Performance Enhancing Proxies. Different types of PEPs are used in different environments to overcome different link characteristics which affect protocol performance. Note that enhancing performance is not necessarily limited in scope to throughput. Other performance related aspects, like usability of a link, may also be addressed. For example, [M-TCP] addresses the issue of keeping TCP connections alive during periods of disconnection in wireless networks.

The following sections describe some of the key characteristics which differentiate different types of PEPs.

2.1 Layering

In principle, a PEP implementation may function at any protocol layer but typically it functions at one or two layers only. In this document we focus on PEP implementations that function at the transport layer or at the application layer as such PEPs are most commonly used to enhance performance over links with problematic characteristics. A PEP implementation may also operate below the network layer, that is, at the link layer, but this document pays only little attention to such PEPs as link layer mechanisms can be and typically are implemented transparently to network and higher layers, requiring no modifications to protocol operation above the link layer. It should also be noted that some PEP implementations operate across several protocol layers by exploiting the protocol information and possibly modifying the protocol operation at more than one layer. For such a PEP it may be difficult to define at which layer(s) it exactly operates on.

2.1.1 Transport Layer PEPs

Transport layer PEPs operate at the transport level. They may be aware of the type of application being carried by the transport layer but, at most, only use this information to influence their behavior with respect to the transport protocol; they do not modify the application protocol in any way, but let the application protocol operate end-to-end. Most transport layer PEP implementations interact with TCP. Such an implementation is called a TCP Performance Enhancing Proxy (TCP PEP). For example, in an environment where ACKs may bunch together causing undesirable data segment bursts, a TCP PEP may be used to simply modify the ACK spacing in order to improve performance. On the other hand, in an environment with a large bandwidth*delay product, a TCP PEP may be used to alter the behavior of the TCP connection by generating local acknowledgments to TCP data segments in order to improve the connection's throughput.

The term TCP spoofing is sometimes used synonymously for TCP PEP functionality. However, the term TCP spoofing more accurately describes the characteristic of intercepting a TCP connection in the middle and terminating the connection as if the interceptor is the intended destination. While this is a characteristic of many TCP PEP implementations, it is not a characteristic of all TCP PEP implementations.

2.1.2 Application Layer PEPs

Application layer PEPs operate above the transport layer. Today, different kinds of application layer proxies are widely used in the Internet. Such proxies include Web caches and relay Mail Transfer Agents (MTA) and they typically try to improve performance or service availability and reliability in general and in a way which is applicable in any environment but they do not necessarily include any optimizations that are specific to certain link characteristics.

Application layer PEPs, on the other hand, can be implemented to improve application protocol as well as transport layer performance with respect to a particular application being used with a particular type of link. An application layer PEP may have the same functionality as the corresponding regular proxy for the same application (e.g., relay MTA or Web caching proxy) but extended with link-specific optimizations of the application protocol operation.

Some application protocols employ extraneous round trips, overly verbose headers and/or inefficient header encoding which may have a significant impact on performance, in particular, with long delay and slow links. This unnecessary overhead can be reduced, in general or

for a particular type of link, by using an application layer PEP in an intermediate node. Some examples of application layer PEPs which have been shown to improve performance on slow wireless WAN links are described in [LHKR96] and [CTC+97].

2.2 Distribution

A PEP implementation may be integrated, i.e., it comprises a single PEP component implemented within a single node, or distributed, i.e., it comprises two or more PEP components, typically implemented in multiple nodes. An integrated PEP implementation represents a single point at which performance enhancement is applied. For example, a single PEP component might be implemented to provide impedance matching at the point where wired and wireless links meet.

A distributed PEP implementation is generally used to surround a particular link for which performance enhancement is desired. For example, a PEP implementation for a satellite connection may be distributed between two PEPs located at each end of the satellite link.

2.3 Implementation Symmetry

A PEP implementation may be symmetric or asymmetric. Symmetric PEPs use identical behavior in both directions, i.e., the actions taken by the PEP occur independent from which interface a packet is received. Asymmetric PEPs operate differently in each direction. The direction can be defined in terms of the link (e.g., from a central site to a remote site) or in terms of protocol traffic (e.g., the direction of TCP data flow, often called the TCP data channel, or the direction of TCP ACK flow, often called the TCP ACK channel). An asymmetric PEP implementation is generally used at a point where the characteristics of the links on each side of the PEP differ or with asymmetric protocol traffic. For example, an asymmetric PEP might be placed at the intersection of wired and wireless networks or an asymmetric application layer PEP might be used for the request-reply type of HTTP traffic. A PEP implementation may also be both symmetric and asymmetric at the same time with regard to different mechanisms it employs. (PEP mechanisms are described in Section 3.)

Whether a PEP implementation is symmetric or asymmetric is independent of whether the PEP implementation is integrated or distributed. In other words, a distributed PEP implementation might operate symmetrically at each end of a link (i.e., the two PEPs function identically). On the other hand, a distributed PEP implementation might operate asymmetrically, with a different PEP implementation at each end of the link. Again, this usually is used with asymmetric links. For example, for a link with an asymmetric

amount of bandwidth available in each direction, the PEP on the end of the link forwarding traffic in the direction with a large amount of bandwidth might focus on locally acknowledging TCP traffic in order to use the available bandwidth. At the same time, the PEP on the end of the link forwarding traffic in the direction with very little bandwidth might focus on reducing the amount of TCP acknowledgement traffic being forwarded across the link (to keep the link from congesting).

2.4 Split Connections

A split connection TCP implementation terminates the TCP connection received from an end system and establishes a corresponding TCP connection to the other end system. In a distributed PEP implementation, this is typically done to allow the use of a third connection between two PEPs optimized for the link. This might be a TCP connection optimized for the link or it might be another protocol, for example, a proprietary protocol running on top of UDP. Also, the distributed implementation might use a separate connection between the proxies for each TCP connection or it might multiplex the data from multiple TCP connections across a single connection between the PEPs.

In an integrated PEP split connection TCP implementation, the PEP again terminates the connection from one end system and originates a separate connection to the other end system. [I-TCP] documents an example of a single PEP split connection implementation.

Many integrated PEPs use a split connection implementation in order to address a mismatch in TCP capabilities between two end systems. For example, the TCP window scaling option [RFC1323] can be used to extend the maximum amount of TCP data which can be "in flight" (i.e., sent and awaiting acknowledgement). This is useful for filling a link which has a high bandwidth*delay product. If one end system is capable of using scaled TCP windows but the other is not, the end system which is not capable can set up its connection with a PEP on its side of the high bandwidth*delay link. The split connection PEP then sets up a TCP connection with window scaling over the link to the other end system.

Split connection TCP implementations can effectively leverage TCP performance enhancements optimal for a particular link but which cannot necessarily be employed safely over the global Internet.

Note that using split connection PEPs does not necessarily exclude simultaneous use of IP for end-to-end connectivity. If a split connection is managed per application or per connection and is under the control of the end user, the user can decide whether a particular

TCP connection or application makes use of the split connection PEP or whether it operates end-to-end. When a PEP is employed on a last hop link, the end user control is relatively easy to implement.

In effect, application layer proxies for TCP-based applications are split connection TCP implementations with end systems using PEPs as a service related to a particular application. Therefore, all transport (TCP) layer enhancements that are available with split connection TCP implementations can also be employed with application layer PEPs in conjunction with application layer enhancements.

2.5 Transparency

Another key characteristic of a PEP is its degree of transparency. PEPs may operate totally transparently to the end systems, transport endpoints, and/or applications involved (in a connection), requiring no modifications to the end systems, transport endpoints, or applications.

On the other hand, a PEP implementation may require modifications to both ends in order to be used. In between, a PEP implementation may require modifications to only one of the ends involved. Either of these kind of PEP implementations is non-transparent, at least to the layer requiring modification.

It is sometimes useful to think of the degree of transparency of a PEP implementation at four levels, transparency with respect to the end systems (network-layer transparent PEP), transparency with respect to the transport endpoints (transport-layer transparent PEP), transparency with respect to the applications (application-layer transparent PEP) and transparency with respect to the users. For example, a user who subscribes to a satellite Internet access service may be aware that the satellite terminal is providing a performance enhancing service even though the TCP/IP stack and the applications in the user’s PC are not aware of the PEP which implements it.

Note that the issue of transparency is not the same as the issue of maintaining end-to-end semantics. For example, a PEP implementation which simply uses a TCP ACK spacing mechanism maintains the end-to-end semantics of the TCP connection while a split connection TCP PEP implementation may not. Yet, both can be implemented transparently to the transport endpoints at both ends. The implications of not maintaining the end-to-end semantics, in particular the end-to-end semantics of TCP connections, are discussed in Section 4.

3. PEP Mechanisms

An obvious key characteristic of a PEP implementation is the mechanism(s) it uses to improve performance. Some examples of PEP mechanisms are described in the following subsections. A PEP implementation might implement more than one of these mechanisms.

3.1 TCP ACK Handling

Many TCP PEP implementations are based on TCP ACK manipulation. The handling of TCP acknowledgments can differ significantly between different TCP PEP implementations. The following subsections describe various TCP ACK handling mechanisms. Many implementations combine some of these mechanisms and possibly employ some additional mechanisms as well.

3.1.1 TCP ACK Spacing

In environments where ACKs tend to bunch together, ACK spacing is used to smooth out the flow of TCP acknowledgments traversing a link. This improves performance by eliminating bursts of TCP data segments that the TCP sender would send due to back-to-back arriving TCP acknowledgments [BPK97].

3.1.2 Local TCP Acknowledgements

In some PEP implementations, TCP data segments received by the PEP are locally acknowledged by the PEP. This is very useful over network paths with a large bandwidth*delay product as it speeds up TCP slow start and allows the sending TCP to quickly open up its congestion window. Local (negative) acknowledgments are often also employed to trigger local (and faster) error recovery on links with significant error rates. (See Section 3.1.3.)

Local acknowledgments are automatically employed with split connection TCP implementations. When local acknowledgments are used, the burden falls upon the TCP PEP to recover any data which is dropped after the PEP acknowledges it.

3.1.3 Local TCP Retransmissions

A TCP PEP may locally retransmit data segments lost on the path between the TCP PEP and the receiving end system, thus aiming at faster recovery from lost data. In order to achieve this the TCP PEP may use acknowledgments arriving from the end system that receives the TCP data segments, along with appropriate timeouts, to determine

when to locally retransmit lost data. TCP PEPs sending local acknowledgments to the sending end system are required to employ local retransmissions towards the receiving end system.

Some PEP implementations perform local retransmissions even though they do not use local acknowledgments to alter TCP connection performance. Basic Snoop [SNOOP] is a well know example of such a PEP implementation. Snoop caches TCP data segments it receives and forwards and then monitors the end-to-end acknowledgments coming from the receiving TCP end system for duplicate acknowledgments (DUPACKs). When DUPACKs are received, Snoop locally retransmits the lost TCP data segments from its cache, suppressing the DUPACKs flowing to the sending TCP end system until acknowledgments for new data are received. The Snoop system also implements an option to employ local negative acknowledgments to trigger local TCP retransmissions. This can be achieved, for example, by applying TCP selective acknowledgments locally on the error-prone link. (See Section 5.3 for details.)

3.1.4 TCP ACK Filtering and Reconstruction

On paths with highly asymmetric bandwidth the TCP ACKs flowing in the low-speed direction may get congested if the asymmetry ratio is high enough. The ACK filtering and reconstruction mechanism addresses this by filtering the ACKs on one side of the link and reconstructing the deleted ACKs on the other side of the link. The mechanism and the issue of dealing with TCP ACK congestion with highly asymmetric links are discussed in detail in [RFC2760] and in [BPK97].

3.2 Tunneling

A Performance Enhancing Proxy may encapsulate messages to carry the messages across a particular link or to force messages to traverse a particular path. A PEP at the other end of the encapsulation tunnel removes the tunnel wrappers before final delivery to the receiving end system. A tunnel might be used by a distributed split connection TCP implementation as the means for carrying the connection between the distributed PEPs. A tunnel might also be used to support forcing TCP connections which use asymmetric routing to go through the end points of a distributed PEP implementation.

3.3 Compression

Many PEP implementations include support for one or more forms of compression. In some PEP implementations, compression may even be the only mechanism used for performance improvement. Compression reduces the number of bytes which need to be sent across a link. This is useful in general and can be very important for bandwidth

limited links. Benefits of using compression include improved link efficiency and higher effective link utilization, reduced latency and improved interactive response time, decreased overhead and reduced packet loss rate over lossy links.

Where appropriate, link layer compression is used. TCP and IP header compression are also frequently used with PEP implementations. [RFC1144] describes a widely deployed method for compressing TCP headers. Other header compression algorithms are described in [RFC2507], [RFC2508] and [RFC2509].

Payload compression is also desirable and is increasing in importance with today's increased emphasis on Internet security. Network (IP) layer (and above) security mechanisms convert IP payloads into random bit streams which defeat applicable link layer compression mechanisms by removing or hiding redundant "information." Therefore, compression of the payload needs to be applied before security mechanisms are applied. [RFC2393] defines a framework where common compression algorithms can be applied to arbitrary IP segment payloads. However, [RFC2393] compression is not always applicable. Many types of IP payloads (e.g., images, audio, video and "zipped" files being transferred) are already compressed. And, when security mechanisms such as TLS [RFC2246] are applied above the network (IP) layer, the data is already encrypted (and possibly also compressed), again removing or hiding any redundancy in the payload. The resulting additional transport or network layer compression will compact only headers, which are small, and possibly already covered by separate compression algorithms of their own.

With application layer PEPs one can employ application-specific compression. Typically an application-specific (or content-specific) compression mechanism is much more efficient than any generic compression mechanism. For example, a distributed Web PEP implementation may implement more efficient binary encoding of HTTP headers, or a PEP can employ lossy compression that reduces the image quality of online-images on Web pages according to end user instructions, thus reducing the number of bytes transferred over a slow link and consequently the response time perceived by the user [LHKR96].

3.4 Handling Periods of Link Disconnection with TCP

Periods of link disconnection or link outages are very common with some wireless links. During these periods, a TCP sender does not receive the expected acknowledgments. Upon expiration of the retransmit timer, this causes TCP to close its congestion window with all of the related drawbacks. A TCP PEP may monitor the traffic coming from the TCP sender towards the TCP receiver behind the

disconnected link. The TCP PEP retains the last ACK, so that it can shut down the TCP sender's window by sending the last ACK with a window set to zero. Thus, the TCP sender will go into persist mode.

To make this work in both directions with an integrated TCP PEP implementation, the TCP receiver behind the disconnected link must be aware of the current state of the connection and, in the event of a disconnection, it must be capable of freezing all timers. [M-TCP] implements such operation. Another possibility is that the disconnected link is surrounded by a distributed PEP pair.

In split connection TCP implementations, a period of link disconnection can easily be hidden from the end host on the other side of the PEP thus precluding the TCP connection from breaking even if the period of link disconnection lasts a very long time; if the TCP PEP cannot forward data due to link disconnection, it stops receiving data. Normal TCP flow control then prevents the TCP sender from sending more than the TCP advertised window allowed by the PEP. Consequently, the PEP and its counterpart behind the disconnected link can employ a modified TCP version which retains the state and all unacknowledged data segments across the period of disconnection and then performs local recovery as the link is reconnected. The period of link disconnection may or may not be hidden from the application and user, depending upon what application the user is using the TCP connection for.

3.5 Priority-based Multiplexing

Implementing priority-based multiplexing of data over a slow and expensive link may significantly improve the performance and usability of the link for selected applications or connections.

A user behind a slow link would experience the link more feasible to use in case of simultaneous data transfers, if urgent data transfers (e.g., interactive connections) could have shorter response time (better performance) than less urgent background transfers. If the interactive connections transmit enough data to keep the slow link fully utilized, it might be necessary to fully suspend the background transfers for awhile to ensure timely delivery for the interactive connections.

In flight TCP segments of an end-to-end TCP connection (with low priority) cannot be delayed for a long time. Otherwise, the TCP timer at the sending end would expire, resulting in suboptimal performance. However, this kind of operation can be controlled in conjunction with a split connection TCP PEP by assigning different priorities for different connections (or applications). A split connection PEP implementation allows the PEP in an intermediate node

to delay the data delivery of a lower-priority TCP flow for an unlimited period of time by simply rescheduling the order in which it forwards data of different flows to the destination host behind the slow link. This does not have a negative impact on the delayed TCP flow as normal TCP flow control takes care of suspending the flow between the TCP sender and the PEP, when the PEP is not forwarding data for the flow, and resumes it once the PEP decides to continue forwarding data for the flow. This can further be assisted, if the protocol stacks on both sides of the slow link implement priority based scheduling of connections.

With such a PEP implementation, along with user-controlled priorities, the user can assign higher priority for selected interactive connection(s) and have much shorter response time for the selected connection(s), even if there are simultaneous low priority bulk data transfers which in regular end-to-end operation would otherwise eat the available bandwidth of the slow link almost completely. These low priority bulk data transfers would then proceed nicely during the idle periods of interactive connections, allowing the user to keep the slow and expensive link (e.g., wireless WAN) fully utilized.

Other priority-based mechanisms may be applied on shared wireless links with more than two terminals. With shared wireless mediums becoming a weak link in Internet QoS architectures, many may turn to PEPs to provide extra priority levels across a shared wireless medium [SHEL00]. These PEPs are distributed on all nodes of the shared wireless medium. For example, in an 802.11 WLAN this PEP is implemented in the access point (base station) and each mobile host. One PEP then uses distributed queuing techniques to coordinate traffic classes of all nodes. This is also sometimes called subnet bandwidth management. See [BBKT97] for an example of queuing techniques which can be used to achieve this. This technique can be implemented either above or below the IP layer. Priority treatment can typically be specified either by the user or by marking the (IPv4) ToS or (IPv6) Traffic Class IP header field.

3.6 Protocol Booster Mechanisms

Work in [FMSBMR98] shows a range of other possible PEP mechanisms called protocol boosters. Some of these mechanisms are specific to UDP flows. For example, a PEP may apply asymmetrical methods such as extra UDP error detection. Since the 16 bit UDP checksum is optional, it is typically not computed. However, for links with errors, the checksum could be beneficial. This checksum can be added to outgoing UDP packets by a PEP.

Symmetrical mechanisms have also been developed. A Forward Erasure Correction (FZC) mechanism can be used with real-time and multicast traffic. The encoding PEP adds a parity packet over a block of packets. Upon reception, the parity is removed and missing data is regenerated. A jitter control mechanism can be implemented at the expense of extra latency. A sending PEP can add a timestamp to outgoing packets. The receiving PEP then delays packets in order to reproduce the correct interval.

4. Implications of Using PEPs

The following sections describe some of the implications of using Performance Enhancing Proxies.

4.1 The End-to-end Argument

As indicated in [RFC1958], the end-to-end argument [SRC84] is one of the architectural principles of the Internet. The basic argument is that, as a first principle, certain required end-to-end functions can only be correctly performed by the end systems themselves. Most of the potential negative implications associated with using PEPs are related to the possibility of breaking the end-to-end semantics of connections. This is one of the main reasons why PEPs are not recommended for general use.

As indicated in Section 2.5, not all PEP implementations break the end-to-end semantics of connections. Correctly designed PEPs do not attempt to replace any application level end-to-end function, but only attempt to add performance optimizations to a subpath of the end-to-end path between the application endpoints. Doing this can be consistent with the end-to-end argument. However, a user or network administrator adding a PEP to his network configuration should be aware of the potential end-to-end implications related to the mechanisms being used by the particular PEP implementation.

4.1.1 Security

In most cases, security applied above the transport layer can be used with PEPs, especially transport layer PEPs. However, today, only a limited number of applications include support for the use of transport (or higher) layer security. Network (IP) layer security (IPsec) [RFC2401], on the other hand, can generally be used by any application, transparently to the application.

4.1.1.1 Security Implications

The most detrimental negative implication of breaking the end-to-end semantics of a connection is that it disables end-to-end use of IPsec. In general, a user or network administrator must choose between using PEPs and using IPsec. If IPsec is employed end-to-end, PEPs that are implemented on intermediate nodes in the network cannot examine the transport or application headers of IP packets because encryption of IP packets via IPsec's ESP header (in either transport or tunnel mode) renders the TCP header and payload unintelligible to the PEPs. Without being able to examine the transport or application headers, a PEP may not function optimally or at all.

If a PEP implementation is non-transparent to the users and the users trust the PEP in the middle, IPsec can be used separately between each end system and PEP. However, in most cases this is an undesirable or unacceptable alternative as the end systems cannot trust PEPs in general. In addition, this is not as secure as end-to-end security. (For example, the traffic is exposed in the PEP when it is decrypted to be processed.) And, it can lead to potentially misleading security level assumptions by the end systems. If the two end systems negotiate different levels of security with the PEP, the end system which negotiated the stronger level of security may not be aware that a lower level of security is being provided for part of the connection. The PEP could be implemented to prevent this from happening by being smart enough to force the same level of security to each end system but this increases the complexity of the PEP implementation (and still is not as secure as end-to-end security).

With a transparent PEP implementation, it is difficult for the end systems to trust the PEP because they may not be aware of its existence. Even if the user is aware of the PEP, setting up acceptable security associations with the PEP while maintaining the PEP's transparent nature is problematic (if not impossible).

Note that even when a PEP implementation does not break the end-to-end semantics of a connection, the PEP implementation may not be able to function in the presence of IPsec. For example, it is difficult to do ACK spacing if the PEP cannot reliably determine which IP packets contain ACKs of interest. In any case, the authors are currently not aware of any PEP implementations, transparent or non-transparent, which provide support for end-to-end IPsec, except in a case where the PEPs are implemented on the end hosts.

4.1.1.2 Security Implication Mitigations

There are some steps which can be taken to allow the use of IPsec and PEPs to coexist. If an end user can select the use of IPsec for some traffic and not for other traffic, PEP processing can be applied to the traffic sent without IPsec. Of course, the user must then do without security for this traffic or provide security for the traffic via other means (for example, by using transport layer security). However, even when this is possible, significant complexity may need to be added to the configuration of the end system.

Another alternative is to implement IPsec between the two PEPs of a distributed PEP implementation. This at least protects the traffic between the two PEPs. (The issue of trusting the PEPs does not change.) In the case where the PEP implementation is not transparent to the user, (assuming that the user trusts the PEPs,) the user can configure his end system to use the PEPs as the end points of an IPsec tunnel. And, an IPsec tunnel could even potentially be used between the end system and a PEP to protect traffic on this part of the path. But, all of this adds complexity. And, it still does not eliminate the risk of the traffic being exposed in the PEP itself as the traffic is received from one IPsec tunnel, processed and then forwarded (even if forwarded through another IPsec tunnel).

4.1.1.3 Security Research Related to PEPs

There is research underway investigating the possibility of changing the implementation of IPsec to be more friendly to the use of PEPs. One approach being actively looked at is the use of multi-layer IP security. [Zhang00] describes a method which allows TCP headers to be encrypted as one layer (with the PEPs in the path of the TCP connections included in the security associations used to encrypt the TCP headers) while the TCP payload is encrypted end-to-end as a separate layer. This still involves trusting the PEP, but to a much lesser extent. However, a drawback to this approach is that it adds a significant amount of complexity to the IP security implementation. Given the existing complexity of IPsec, this drawback is a serious impediment to the standardization of the multi-layer IP security idea and it is very unlikely that this approach will be adopted as a standard any time soon. Therefore, relying on this type of approach will likely involve the use of non-standard protocols (and the associated risk of doing so).

4.1.2 Fate Sharing

Another important aspect of the end-to-end argument is fate sharing. If a failure occurs in the network, the ability of the connection to survive the failure depends upon how much state is being maintained

on behalf of the connection in the network and whether the state is self-healing. If no connection specific state resides in the network or such state is self-healing as in case of regular end-to-end operation, then a failure in the network will break the connection only if there is no alternate path through the network between the end systems. And, if there is no path, both end systems can detect this. However, if the connection depends upon some state being stored in the network (e.g., in a PEP), then a failure in the network (e.g., the node containing a PEP crashes) causes this state to be lost, forcing the connection to terminate even if an alternate path through the network exists.

The importance of this aspect of the end-to-end argument with respect to PEPs is dependent upon both the PEP implementation and upon the types of applications being used. Sometimes coincidentally but more often by design, PEPs are used in environments where there is no alternate path between the end systems and, therefore, a failure of the intermediate node containing a PEP would result in the termination of the connection in any case. And, even when this is not the case, the risk of losing the connection in the case of regular end-to-end operation may exist as the connection could break for some other reason, for example, a long enough link outage of a last-hop wireless link to the end host. Therefore, users may choose to accept the risk of a PEP crashing in order to take advantage of the performance gains offered by the PEP implementation. The important thing is that accepting the risk should be under the control of the user (i.e., the user should always have the option to choose end-to-end operation) and, if the user chooses to use the PEP, the user should be aware of the implications that a PEP failure has with respect to the applications being used.

4.1.3 End-to-end Reliability

Another aspect of the end-to-end argument is that of acknowledging the receipt of data end-to-end in order to achieve reliable end-to-end delivery of data. An application aiming at reliable end-to-end delivery must implement an end-to-end check and recovery at the application level. According to the end-to-end argument, this is the only possibility to correctly implement reliable end-to-end operation. Otherwise the application violates the end-to-end argument. This also means that a correctly designed application can never fully rely on the transport layer (e.g., TCP) or any other communication subsystem to provide reliable end-to-end delivery.

First, a TCP connection may break down for some reason and result in lost data that must be recovered at the application level. Second, the checksum provided by TCP may be considered inadequate, resulting in undetected (by TCP) data corruption [Pax99] and requiring an

application level check for data corruption. Third, a TCP acknowledgement only indicates that data was delivered to the TCP implementation on the other end system. It does not guarantee that the data was delivered to the application layer on the other end system. Therefore, a well designed application must use an application layer acknowledgement to ensure end-to-end delivery of application layer data. Note that this does not diminish the value of a reliable transport protocol (i.e., TCP) as such a protocol allows efficient implementation of several essential functions (e.g., congestion control) for an application.

If a PEP implementation acknowledges application data prematurely (before the PEP receives an application ACK from the other endpoint), end-to-end reliability cannot be guaranteed. Typically, application layer PEPs do not acknowledge data prematurely, i.e., the PEP does not send an application ACK to the sender until it receives an application ACK from the receiver. And, transport layer PEP implementations, including TCP PEPs, generally do not interfere with end-to-end application layer acknowledgments as they let applications operate end-to-end. However, the user and/or network administrator employing the PEP must understand how it operates in order to understand the risks related to end-to-end reliability.

Some Internet applications do not necessarily operate end-to-end in their regular operation, thus abandoning any end-to-end reliability guarantee. For example, Internet email delivery often operates via relay Mail Transfer Agents, that is, relay Simple Mail Transfer Protocol (SMTP) servers. An originating MTA (SMTP server) sends the mail message to a relay MTA that receives the mail message, stores it in non-volatile storage (e.g., on disk) and then sends an application level acknowledgement. The relay MTA then takes "full responsibility" for delivering the mail message to the destination SMTP server (maybe via another relay MTA); it tries to forward the message for a relatively long time (typically around 5 days). This scheme does not give a 100% guarantee of email delivery, but reliability is considered "good enough".

An application layer PEP for this kind of an application may acknowledge application data (e.g., mail message) without essentially decreasing reliability, as long as the PEP operates according to the same procedure as the regular proxy (e.g., relay MTA). Again, as indicated above, the user and/or network administrator employing such a PEP needs to understand how it operates in order to understand the reliability risks associated with doing so.

4.1.4 End-to-end Failure Diagnostics

Another aspect of the end-to-end argument is the ability to support end-to-end failure diagnostics when problems are encountered. If a network problem occurs which breaks a connection, the end points of the connection will detect the failure via timeouts. However, the existence of a PEP in between the two end points could delay (sometimes significantly) the detection of the failure by one or both of the end points. (Of course, some PEPs are intentionally designed to hide these types of failures as described in Section 3.4.) The implications of delayed detection of a failed connection depend on the applications being used. Possibilities range from no impact at all (or just minor annoyance to the end user) all the way up to impacting mission critical business functions by delaying switchovers to alternate communications paths.

In addition, tools used to debug connection failures may be affected by the use of a PEP. For example, PING (described in [RFC792] and [RFC2151]) is often used to test for connectivity. But, because PING is based on ICMP instead of TCP (i.e., it is implemented using ICMP Echo and Reply commands at the network layer), it is possible that the configuration of the network might route PING traffic around the PEP. Thus, PING could indicate that an end-to-end path exists between two hosts when it does not actually exist for TCP traffic. Even when the PING traffic does go through the PEP, the diagnostics indications provided by the PING traffic are altered. For example, if the PING traffic goes transparently through the PEP, PING does not provide any indication that the PEP exists and since the PING traffic is not being subjected to the same processing as TCP traffic, it may not necessarily provide an accurate indication of the network delay being experienced by TCP traffic. On the other hand, if the PEP terminates the PING and responds to it on behalf of the end host, then the PING provides information only on the connectivity to the PEP. Traceroute (also described in [RFC2151]) is similarly affected by the presence of the PEP.

4.2 Asymmetric Routing

Deploying a PEP implementation usually requires that traffic to and from the end hosts is routed through the intermediate node(s) where PEPs reside. With some networks, this cannot be accomplished, or it might require that the intermediate node is located several hops away from the target link edge which in turn is impractical in many cases and may result in non-optimal routing.

Note that this restriction does not apply to all PEP implementations. For example, a PEP which is simply doing ACK spacing only needs to see one direction of the traffic flow (the direction in which the ACKs are flowing). ACK spacing can be done without seeing the actual flow of data.

4.3 Mobile Hosts

In environments where a PEP implementation is used to serve mobile hosts, additional problems may be encountered because PEP related state information may need to be transferred to a new PEP node during a handoff.

When a mobile host moves, it is subject to handovers. If the intermediate node and home for the serving PEP changes due to handover, any state information that the PEP maintains and is required for continuous operation must be transferred to the new intermediate node to ensure continued operation of the connection. This requires extra work and overhead and may not be possible to perform fast enough, especially if the host moves frequently over cell boundaries of a wireless network. If the mobile host moves to another IP network, routing to and from the mobile host may need to be changed to traverse a new PEP node.

Today, mobility implications with respect to using PEPs are more significant to W-LAN networks than to W-WAN networks. Currently, a W-WAN base station typically does not provide the mobile host with the connection point to the wireline Internet. (A W-WAN base station may not even have an IP stack.) Instead, the W-WAN network takes care of mobility with the connection point to the wireline Internet remaining unchanged while the mobile host moves. Thus, PEP state handover is not currently required in most W-WAN networks when the host moves. However, this is generally not true in W-LAN networks and, even in the case of W-WAN networks, the user and/or network administrator using a PEP needs to be cognizant of how the W-WAN base stations and the PEP work in case W-WAN PEP state handoff becomes necessary in the future.

4.4 Scalability

Because a PEP typically processes packet information above the IP layer, a PEP requires more processing power per packet than a router. Therefore, PEPs will always be (at least) one step behind routers in terms of the total throughput they can support. (Processing above the IP layer is also more difficult to implement in hardware.) In addition, since most PEP implementations require per connection state, PEP memory requirements are generally significantly higher

than with a router. Therefore, a PEP implementation may have a limit on the number of connections which it can support whereas a router has no such limitation.

Increased processing power and memory requirements introduce scalability issues with respect to the use of PEPs. Placement of a PEP on a high speed link or a link which supports a large number of connections may require network topology changes beyond just inserting the PEP into the path of the traffic. For example, if a PEP can only handle half of the traffic on a link, multiple PEPs may need to be used in parallel, adding complexity to the network configuration to divide the traffic between the PEPs.

4.5 Other Implications of Using PEPs

This document describes some significant implications with respect to using Performance Enhancing Proxies. However, the list of implications provided in this document is not necessarily exhaustive. Some examples of other potential implications related to using PEPs include the use of PEPs in multi-homing environments and the use of PEPs with respect to Quality of Service (QoS) transparency. For example, there may be potential interaction with the priority-based multiplexing mechanism described in Section 3.5 and the use of differentiated services [RFC2475]. Therefore, users and network administrators who wish to deploy a PEP should look not only at the implications described in this document but also at the overall impact (positive and negative) that the PEP will have on their applications and network infrastructure, both initially and in the future when new applications are added and/or changes in the network infrastructure are required.

5. PEP Environment Examples

The following sections describe examples of environments where PEP is currently used to improve performance. The examples are provided to illustrate the use of the various PEP types and PEP mechanisms described earlier in the document and to help illustrate the motivation for their development and use.

5.1 VSAT Environments

Today, VSAT networks are implemented with geosynchronous satellites. VSAT data networks are typically implemented using a star topology. A large hub earth station is located at the center of the star with VSATs used at the remote sites of the network. Data is sent from the hub to the remote sites via an outroute. Data is sent from the remote sites to the hub via one or more inroutes. VSATs represent an environment with highly asymmetric links, with an outroute typically

much larger than an inroute. (Multiple inroutes can be used with each outroute but any particular VSAT only has access to a single inroute at a time, making the link asymmetric.)

VSAT networks are generally used to implement private networks (i.e., intranets) for enterprises (e.g., corporations) with geographically dispersed sites. VSAT networks are rarely, if ever, used to implement Internet connectivity except at the edge of the Internet (i.e., as the last hop). Connection to the Internet for the VSAT network is usually implemented at the VSAT network hub site using appropriate firewall and (when necessary) NAT [RFC2663] devices.

5.1.1 VSAT Network Characteristics

With respect to TCP performance, VSAT networks exhibit the following subset of the satellite characteristics documented in [RFC2488]:

Long feedback loops

Propagation delay from a sender to a receiver in a geosynchronous satellite network can range from 240 to 280 milliseconds, depending on where the sending and receiving sites are in the satellite footprint. This makes the round trip time just due to propagation delay at least 480 milliseconds. Queueing delay and delay due to shared channel access methods can sometimes increase the total delay up to on the order of a few seconds.

Large bandwidth*delay products

VSAT networks can support capacity ranging from a few kilobits per second up to multiple megabits per second. When combined with the relatively long round trip time, TCP needs to keep a large number of packets "in flight" in order to fully utilize the satellite link.

Asymmetric capacity

As indicated above, the outroute of a VSAT network is usually significantly larger than an inroute. Even though multiple inroutes can be used within a network, a given VSAT can only access one inroute at a time. Therefore, the incoming (outroute) and outgoing (inroute) capacity for a VSAT is often very asymmetric. As outroute capacity has increased in recent years, ratios of 400 to 1 or greater are becoming more and more common. With a TCP maximum segment size of 1460 bytes and delayed acknowledgments [RFC1122] in use, the ratio of IP packet bytes for data to IP packet bytes for ACKs is only (3000 to 40) 75 to 1.

Thus, inroute capacity for carrying ACKs can have a significant impact on TCP performance. (The issue of asymmetric link impact on TCP performance is described in more detail in [BPK97].)

With respect to the other satellite characteristics listed in [RFC2488], VSAT networks typically do not suffer from intermittent connectivity or variable round trip times. Also, VSAT networks generally include a significant amount of error correction coding. This makes the bit error rate very low during clear sky conditions, approaching the bit error rate of a typical terrestrial network. In severe weather, the bit error rate may increase significantly but such conditions are rare (when looked at from an overall network availability point of view) and VSAT networks are generally engineered to work during these conditions but not to optimize performance during these conditions.

5.1.2 VSAT Network PEP Implementations

Performance Enhancing Proxies implemented for VSAT networks generally focus on improving throughput (for applications such as FTP and HTTP web page retrievals). To a lesser degree, PEP implementations also work to improve interactive response time for small transactions.

There is not a dominant PEP implementation used with VSAT networks. Each VSAT network vendor tends to implement their own version of PEP functionality, integrated with the other features of their VSAT product. [HNS] and [SPACENET] describe VSAT products with integrated PEP capabilities. There are also third party PEP implementations designed to be used with VSAT networks. These products run on nodes external to the VSAT network at the hub and remote sites. NettGain [FLASH] and Venturi [FOURELLE] are examples of such products. VSAT network PEP implementations generally share the following characteristics:

- They focus on improving TCP performance;
- They use an asymmetric distributed implementation;
- They use a split connection approach with local acknowledgments and local retransmissions;
- They support some form of compression to reduce the amount of bandwidth required (with emphasis on saving inroute bandwidth).

The key differentiators between VSAT network PEP implementations are:

- The maximum throughput they attempt to support (mainly a function of the amount of buffer space they use);

- The protocol used over the satellite link. Some implementations use a modified version of TCP while others use a proprietary protocol running on top of UDP;

- The type of compression used. Third party VSAT network PEP implementations generally focus on application (e.g., HTTP) specific compression algorithms while PEP implementations integrated into the VSAT network generally focus on link specific compression.

PEP implementations integrated into a VSAT product are generally transparent to the end systems. Third party PEP implementations used with VSAT networks usually require configuration changes in the remote site end systems to route TCP packets to the remote site proxies but do not require changes to the hub site end systems. In some cases, the PEP implementation is actually integrated transparently into the end system node itself, using a "bump in the stack" approach. In all cases, the use of a PEP is non-transparent to the user, i.e., the user is aware when a PEP implementation is being used to boost performance.

5.1.3 VSAT Network PEP Motivation

VSAT networks, since the early stages of their deployment, have supported the use of local termination of a protocol (e.g., SDLC and X.25) on each side of the satellite link to hide the satellite link from the applications using the protocol. Therefore, when LAN capabilities were added to VSAT networks, VSAT customers expected and, in fact, demanded, the use of similar techniques for improving the performance of IP based traffic, in particular TCP traffic.

As indicated in Section 5.1, VSAT networks are primarily used to implement intranets with Internet connectivity limited to and closely controlled at the hub site of the VSAT network. Therefore, VSAT customers are not as affected (or at least perceive that they are not as affected) by the Internet related implications of using PEPs as are other technologies. Instead, what is more important to VSAT customers is the optimization of the network. And, VSAT customers, in general, prefer that the optimization of the network be done by the network itself rather than by implementing changes (such as enabling the TCP scaled window option) to their own equipment. VSAT customers prefer to optimize their end system configuration for local communications related to their local mission critical functions and let the VSAT network hide the presence of the satellite link as much as possible. VSAT network vendors have also been able to use PEP functionality to provide value added "services" to their customers such as extending the useful of life of older equipment which includes older, "non-modern" TCP stacks.

Of course, as the line between intranets and the Internet continues to fade, the implications of using PEPs start to become more significant for VSAT networks. For example, twelve years ago security was not a major concern because the equipment cost related to being able to intercept VSAT traffic was relatively high. Now, as technology has advanced, the cost is much less prohibitive. Therefore, because the use of PEP functionality in VSAT networks prevents the use of IPsec, customers must rely on the use of higher layer security mechanisms such as TLS or on proprietary security mechanisms implemented in the VSAT networks themselves (since currently many applications are incapable of making (or simply don't make) use of the standardized higher layer security mechanisms). This, in turn, affects the cost of the VSAT network as well as affects the ability of the customers to make use of Internet based capabilities.

5.2 W-WAN Environments

In mobile wireless WAN (W-WAN) environments the wireless link is typically used as the last-hop link to the end user. W-WANs include such networks as GSM [GSM], GPRS [GPRS],[BW97], CDPD [CDPD], IS-95 [CDMA], RichoNet, and PHS. Many of these networks, but not all, have been designed to provide mobile telephone voice service in the first place but include data services as well or they evolve from a mobile telephone network.

5.2.1 W-WAN Network Characteristics

W-WAN links typically exhibit some combination of the following link characteristics:

- low bandwidth (with some links the available bandwidth might be as low as a few hundred bits/sec)

- high latency (minimum round-trip delay close to one second is not exceptional)

- high BER resulting in frame or packet losses, or long variable delays due to local link-layer error recovery

- some W-WAN links have a lot of internal buffer space which tend to accumulate data, thus resulting in increased round-trip delay due to long (and variable) queuing delays

- on some W-WAN links the users may share common channels for their data packet delivery which, in turn, may cause unexpected delays to the packet delivery of a user due to simultaneous use of the same channel resources by the other users

- unexpected link disconnections (or intermittent link outages) may occur frequently and the period of disconnection may last a very long time

- (re)setting the link-connection up may take a long time (several tens of seconds or even minutes)

- the W-WAN network typically takes care of terminal mobility: the connection point to the Internet is retained while the user moves with the mobile host

- the use of most W-WAN links is expensive. Many of the service providers apply time-based charging.

5.2.2 W-WAN PEP Implementations

Performance Enhancing Proxies implemented for W-WAN environments generally focus on improving the interactive response time but at the same time aim at improving throughput, mainly by reducing the transfer volume over the inherently slow link in various ways. To achieve this, typically enhancements are applied at almost all protocol layers.

5.2.2.1 Mowgli System

The Mowgli system [KRA94] is one of the early approaches to address the challenges induced by the problematic characteristics of low bandwidth W-WAN links.

The indirect approach used in Mowgli is not limited to a single layer as in many other split connection approaches, but it involves all protocol layers. The basic architecture is based on split TCP (UDP is also supported) together with full support for application layer proxies with a distributed PEP approach. An application layer proxy pair may be added between a client and server, the agent (local proxy) on a mobile host and the proxy on an intermediate node that provides the mobile host with the connection to the wireline Internet. Such a pair may be either explicit or fully transparent to the applications, but it is, at all times, under end-user control thus allowing the user to select the traffic that traverses through the PEP implementation and choose end-to-end IP for other traffic.

In order to allow running legacy applications unmodified and without recompilation, the socket layer implementation on the mobile host is slightly modified to connect the applications, which are configured to traverse through the PEP, to a local agent while retaining the original TCP/IP socket semantics. Two types of application layer agent-proxy pairs can be configured for mobile host application use.

A generic pair can be used with any application and it simply provides split transport service with some optional generic enhancements like compression. An application-specific pair can be retailed for any application or a group of applications that are able to take leverage on the same kind of enhancements. A good example of enhancements achieved with an application-specific proxy pair is the Mowgli WWW system that improves significantly the user perceived response time of Web browsing mainly by reducing the transfer volume and the number of round trips over the wireless link [LAKLR95], [LHKR96].

Mowgli provides also an option to replace the TCP/IP core protocols on the last-hop link with a custom protocol that is tuned for low-bandwidth W-WAN links [KRLKA97]. This protocol was designed to provide the same transport service with similar semantics as regular TCP and UDP provide, but use a different protocol implementation that can freely apply any appropriate protocol mechanisms without being constrained by the current TCP/IP packet format or protocol operation. As this protocol is required to operate over a single logical link only, it could partially combine the protocol control information and protocol operation of the link, network, and transport layers. In addition, the protocol can operate on top of various link services, for example on top of different raw link services, on top of PPP, on top of IP, or even on top of a single TCP connection using it as a link service and implementing "TCP multiplexing" over it. In all other cases, except when the protocol is configured to operate on top of raw (wireless) link service, IP may co-exist with the custom protocol allowing simultaneous end-to-end IP delivery for the traffic not traversing through the PEP implementation.

Furthermore, the custom protocol can be run in different operation modes which turn on or off certain protocol functions depending on the underlying link service. For example, if the underlying link service provides reliable data delivery, the checksum and the window-based error recovery can be turned off, thus reducing the protocol overhead; only a very simple recovery mechanism is needed to allow recovery from an unexpected link disconnection. Therefore, the protocol design was able to use extremely efficient header encoding (only 1-3 bytes per packet in a typical case), reduce the number of round trips significantly, and various features that are useful with low-bandwidth W-WAN links were easy to add. Such features include suspending the protocol operation over the periods of link disconnection or link outage together with fast start once the link becomes operational again, priority-based multiplexing of user data over the W-WAN link thus offering link capacity to interactive

applications in a timely manner even in presence of bandwidth-intensive background transfers, and link-level flow control to prevent data from accumulating into the W-WAN link internal buffers.

If desired, regular TCP/IP transport, possibly with corresponding protocol modifications in TCP (and UDP) that would tune it more suitable for W-WAN links, can be employed on the last-hop link.

5.2.2.2 Wireless Application Protocol (WAP)

The Mowgli system was designed to support mobile hosts that are attached to the Internet over constrained links, but did not address the specific challenges with low-end mobile devices. Many mobile wireless devices are power, memory, and processing constrained, and the communication links to these devices have lower bandwidth and less stable connections. These limitations led designers to develop the Wireless Application Protocol (WAP) that specifies an application framework and network protocols intended to work across differing narrowband wireless network technologies bringing Internet content and advanced data services to low-end digital cellular phones and other mobile wireless terminals, such as pagers and PDAs.

The WAP model consists of a WAP client (mobile terminal), a WAP proxy, and an origin server. It requires a WAP proxy between the WAP client and the server on the Internet. WAP uses a layered, scalable architecture [WAPARCH], specifying the following five protocol layers to be used between the terminal and the proxy: Application Layer (WAE) [WAPWAE], Session Layer (WSP) [WAPWSP], Transaction Layer (WTP) [WAPWTP], Security Layer (WTLS) [WAPWTLS], and Transport Layer (WDP) [WAPWDP]. Standard Internet protocols are used between the proxy and the origin server. If the origin server includes WAP proxy functionality, it is called a WAP Server.

In a typical scenario, a WAP client sends an encoded WAP request to a WAP proxy. The WAP proxy translates the WAP request into a WWW (HTTP) request, performing the required protocol conversions, and submits this request to a standard web server on the Internet. After the web server responds to the WAP proxy, the response is encoded into a more compact binary format to decrease the size of the data over the air. This encoded response is forwarded to the WAP client [WAPPROXY].

WAP operates over a variety of bearer datagram services. When communicating over these bearer services, the WAP transport layer (WDP) is always used between the WAP client and WAP proxy and it provides port addressed datagram service to the higher WAP layers. If the bearer service supports IP (e.g., GSM-CSD, GSM-GPRS, IS-136, CDPD), UDP is used as the datagram protocol. However, if the bearer

service does not support IP (e.g., GSM-SMS, GSM-USSD, GSM Cell Broadcast, CDMS-SMS, TETRA-SDS), WDP implements the required datagram protocol as an adaptation layer between the bearer network and the protocol stack.

The use of the other layers depends on the port number. WAP has registered a set of well-known ports with IANA. The port number selected by the application for communication between a WAP client and proxy defines the other layers to be used at each end. The security layer, WTLS, provides privacy, data integrity and authentication. Its functionality is similar to TLS 1.0 [RFC2246] extended with datagram support, optimized handshake and dynamic key refreshing. If the origin server includes WAP proxy functionality, it might be used to facilitate the end-to-end security solutions, otherwise it provides security between the mobile terminal and the proxy.

The transaction layer, WTP, is message based without connection establishment and tear down. It supports three types of transaction classes: an unconfirmed request (unidirectional), a reliable (confirmed) request (unidirectional), and a reliable (confirmed) request-reply transaction. Data is carried in the first packet and 3-way handshake is eliminated to reduce latencies. In addition acknowledgments, retransmission, and flow control are provided. It allows more than one outstanding transaction at a time. It handles the bearer dependence of a transfer, e.g., selects timeout values and packet sizes according to the bearer. Unfortunately, WTP uses fixed retransmission timers and does not include congestion control, which is a potential problem area as the use of WAP increases [RFC3002].

The session layer, WSP, supports binary encoded HTTP 1.1 with some extensions such as long living session with suspend/resume facility and state handling, header caching, and push facility. On top of the architecture is the application environment (WAE).

5.2.3 W-WAN PEP Motivation

As indicated in Section 5.2.1, W-WAN networks typically offer very low bandwidth connections with high latency and relatively frequent periods of link disconnection and they usually are expensive to use. Therefore, the transfer volume and extra round-trips, such as those associated with TCP connection setup and teardown, must be reduced and the slow W-WAN link should be efficiently shielded from excess traffic and global (wired) Internet congestion to make Internet access usable and economical. Furthermore, interactive traffic must be transmitted in a timely manner even if there are other simultaneous bandwidth intensive (background) transfers and during the periods with connectivity the link must be kept fully utilized

due to expensive use. In addition, the (long) periods of link disconnection must not abort active (bulk data) transfers, if an end-user so desires.

As (all) applications cannot be made mobility/W-WAN aware in short time frame or maybe ever, support for mobile W-WAN use should be implemented in a way which allows most applications, at least those running on fixed Internet hosts, to continue their operation unmodified.

5.3 W-LAN Environments

Wireless LANs (W-LAN) are typically organized in a cellular topology where an access point with a W-LAN transceiver controls a single cell. A cell is defined in terms of the coverage area of the base station. The access points are directly connected to the wired network. The access point in each of the cells is responsible for forwarding packets to and from the hosts located in the cell. Often the hosts with W-LAN transceivers are mobile. When such a mobile host moves from one cell to another cell, the responsibility for forwarding packets between the wired network and the mobile host must be transferred to the access point of the new cell. This is known as a handoff. Many W-LAN systems also support an operation mode enabling ad-hoc networking. In this mode access points are not necessarily needed, but hosts with W-LAN transceiver can communicate directly with the other hosts within the transceiver’s transmission range.

5.3.1 W-LAN Network Characteristics

Current wireless LANs typically provide link bandwidth from 1 Mbps to 11 Mbps. In the future, wide deployment of higher bandwidths up to 54 Mbps or even higher can be expected. The round-trip delay with wireless LANs is on the order of a few milliseconds or tens of milliseconds. Examples of W-LANs include IEEE 802.11, HomeRF, and Hiperlan. Wireless personal area networks (WPAN) such as Bluethooth can use the same PEP techniques.

Wireless LANs are error-prone due to bit errors, collisions and link outages. In addition, consecutive packet losses may also occur during handoffs. Most W-LAN MAC protocols perform low level retransmissions. This feature shields upper layers from most losses. However, unavoidable losses, retransmission latency and link outages still affect upper layers. TCP performance over W-LANs or a network path involving a W-LAN link is likely to suffer from these effects.

As TCP wrongly interprets these packet losses to be network congestion, the TCP sender reduces its congestion window and is often forced to timeout in order to recover from the consecutive losses. The result is often unacceptably poor end-to-end performance.

5.3.2 W-LAN PEP Implementations: Snoop

Berkeley's Snoop protocol [SNOOP] is a TCP-specific approach in which a TCP-aware module, a Snoop agent, is deployed at the W-LAN base station that acts as the last-hop router to the mobile host. Snoop aims at retaining the TCP end-to-end semantics. The Snoop agent monitors every packet that passes through the base station in either direction and maintains soft state for each TCP connection. The Snoop agent is an asymmetric PEP implementation as it operates differently on TCP data and ACK channels as well as on the uplink (from the mobile host) and downlink (to the mobile host) TCP segments.

For a data transfer to a mobile host, the Snoop agent caches unacknowledged TCP data segments which it forwards to the TCP receiver and monitors the corresponding ACKs. It does two things:

1. Retransmits any lost data segments locally by using local timers and TCP duplicate ACKs to identify packet loss, instead of waiting for the TCP sender to do so end-to-end.

2. Suppresses the duplicate ACKs on their way from the mobile host back to the sender, thus avoiding fast retransmit and congestion avoidance at the latter.

Suppressing the duplicate ACKs is required to avoid unnecessary fast retransmits by the TCP sender as the Snoop agent retransmits a packet locally. Consider a system that employs the Snoop agent and a TCP sender S that sends packets to receiver R via a base station BS. Assume that S sends packets A, B, C, D, E (in that order) which are forwarded by BS to the wireless receiver R. Assume the first transmission of packet B is lost due to errors on the wireless link. In this case, R receives packets A, C, D, E and B (in that order). Receipt of packets C, D and E trigger duplicate ACKs. When S receives three duplicate ACKs, it triggers fast retransmit (which results in a retransmission, as well as reduction of the congestion window). The Snoop agent also retransmits B locally, when it receives three duplicate ACKs. The fast retransmit at S occurs despite the local retransmit on the wireless link, degrading throughput. Snoop deals with this problem by dropping TCP duplicate ACKs appropriately at BS.

For a data transfer from a mobile host, the Snoop agent detects the packet losses on the wireless link by monitoring the data segments it forwards. It then employs either Negative Acknowledgements (NAK) locally or Explicit Loss Notifications (ELN) to inform the mobile sender that the packet loss was not related to congestion, thus allowing the sender to retransmit without triggering normal congestion control procedures. To implement this, changes at the mobile host are required.

When a Snoop agent uses NAKs to inform the TCP sender of the packet losses on the wireless link, one possibility to implement them is using the Selective Acknowledgment (SACK) option of TCP [RFC2018]. This requires enabling SACK processing at the mobile host. The Snoop agent sends a TCP SACK, when it detects a hole in the transmission sequence from the mobile host or when it has not received any new packets from the mobile host for a certain time period. This approach relies on the advisory nature of the SACKs: the mobile sender is advised to retransmit the missing segments indicated by SACK, but it must not assume successful end-to-end delivery of the segments acknowledged with SACK as these segments might get lost later in the path to the receiver. Instead, the sender must wait for a cumulative ACK to arrive.

When the ELN mechanism is used to inform the mobile sender of the packet losses, Snoop uses one of the 'unreserved' bits in the TCP header for ELN [SNOOPELN]. The Snoop agent keeps track of the holes that correspond to segments lost over the wireless link. When a (duplicate) ACK corresponding to a hole in the sequence space arrives from the TCP receiver, the Snoop agent sets the ELN bit on the ACK to indicate that the loss is unrelated to congestion and then forwards the ACK to the TCP sender. When the sender receives a certain number of (duplicate) ACKs with ELN (a configurable variable at the mobile host, e.g., two), it retransmit the missing segment without performing any congestion control measures.

The ELN mechanism using one of the six bits reserved for future use in the TCP header is dangerous as it exercises checks that might not be correctly implemented in TCP stacks, and may expose bugs.

A scheme such as Snoop is needed only if the possibility of a fast retransmit due to wireless errors is non-negligible. In particular, if the wireless link uses link-layer recovery for lost data, then this scheme is not beneficial. Also, if the TCP window tends to stay smaller than four segments, for example, due to congestion related losses on the wired network, the probability that the Snoop agent will have an opportunity to locally retransmit a lost packet is small. This is because at least three duplicate ACKs are needed to trigger the local retransmission, but due to small window the Snoop

agent may not be able to forward three new packets after the lost packet and thus induce the required three duplicate ACKs. Conversely, when the TCP window is large enough, Snoop can provide significant performance improvement (compared with standard TCP).

In order to alleviate the problem with small TCP windows, Snoop proposes a solution in which a TCP sender is allowed to transmit a new data segment for each duplicate ACK it receives as long as the number of duplicate ACKs is less than the threshold for TCP fast retransmission (three duplicate ACKs). If the new segment reaches the receiver, it will generate another duplicate ACK which, in turn, allows the sender to transmit yet another data segment. This continues until enough duplicate ACKs have accumulated to trigger TCP fast retransmission. This proposal is the same as the "Limited Transfer" proposal [RFC3042] that has recently been forwarded to the standards track. However, to be able to benefit from this solution, it needs to be deployed on TCP senders and therefore it is not ready for use in a short time frame.

Snoop requires the intermediate node (base station) to examine and operate on the traffic between the mobile host and the other end host on the wired Internet. Hence, Snoop does not work if the IP traffic is encrypted. Possible solutions involve:

- making the Snoop agent a party to the security association between the client and the server;

- IPsec tunneling mode, terminated at the Snooping base station.

However, these techniques require that users trust base stations.

Snoop also requires that both the data and the corresponding ACKs traverse the same base station. Furthermore, the Snoop agent may duplicate efforts by the link layer as it retransmits the TCP data segments "at the transport layer" across the wireless link. (Snoop has been described by its designers as a TCP-aware link layer. This is the right approach: the link and network layers can be much more aware of each other than strict layering suggests.)

5.3.3 W-LAN PEP Motivation

Wireless LANs suffer from an error prone wireless channel. Errors can typically be considered bursty and channel conditions may change rapidly from mobility and environmental changes. Packets are dropped from bit errors or during handovers. Periods of link outage can also be experienced. Although the typical MAC performs retransmissions, dropped packets, outages and retransmission latency still can have serious performance implications for IP performance, especially TCP.

PEPs can be used to alleviate problems caused by packet losses, protect TCP from link outages, and to add priority multiplexing. Techniques such as Snoop are integrally implemented in access points, while priority and compression schemes are distributed across the W-LAN.

6. Security Considerations

The use of Performance Enhancing Proxies introduces several issues which impact security. First, (as described in detail in Section 4.1.1,) using PEPs and using IPsec is generally mutually exclusive. Unless the PEP is also both capable and trusted to be the endpoint of an IPsec tunnel (and the use of an IPsec tunnel is deemed good enough security for the applicable threat model), a user or network administrator must choose between improved performance and network layer security. In some cases, transport (or higher) layer security can be used in conjunction with a PEP to mitigate the impact of not having network layer security. But, support by applications for the use of transport (or higher) layer security is far from ubiquitous.

Additionally, the PEP itself needs to be protected from attack. First, even when IPsec tunnels are used with the PEP, the PEP represents a point in the network where traffic is exposed. And, the placement of a PEP in the network makes it an ideal platform from which to launch a denial of service or man in the middle attack. (Also, taking the PEP out of action is a potential denial of service attack itself.) Therefore, the PEP must be protected (e.g., by a firewall) or must protect itself from improper access by an attacker just like any other device which resides in a network.

7. IANA Considerations

This document is an informational overview document and, as such, does not introduce new nor modify existing name or number spaces managed by IANA.

8. Acknowledgements

This document grew out of the Internet-Draft "TCP Performance Enhancing Proxy Terminology", RFC 2757 "Long Thin Networks", and work done in the IETF TCPSAT working group. The authors are indebted to the active members of the PILC working group. In particular, Joe Touch and Mark Allman gave us invaluable feedback on various aspects of the document and Magdolna Gerendai provided us with essential help on the WAP example.

9. References

[BBKT97] P. Bhagwat, P. Bhattacharya, A. Krishma, S.K. Tripathi, "Using channel state dependent packet scheduling to improve TCP throughput over wireless LANs," ACM Wireless Networks, March 1997, pp. 91 - 102. Available at: http://www.acm.org/pubs /articles/journals/wireless/1997-3-1/p91-bhagwat/p91-bhagwat.pdf

[BPK97] H. Balakrishnan, V.N. Padmanabhan, R.H. Katz, "The Effects of Asymmetry on TCP Performance," Proc. ACM/IEEE Mobicom, Budapest, Hungary, September 1997.

[BW97] G. Brasche, B. Walke, "Concepts, Services, and Protocols of the New GSM Phase 2+ general Packet Radio Service," IEEE Communications Magazine, Vol. 35, No. 8, August 1997.

[CDMA] Electronic Industry Alliance (EIA)/Telecommunications Industry Association (TIA), IS-95: Mobile Station-Base Station Compatibility Standard for Dual-Mode Wideband Spread Spectrum Cellular System, 1993.

[CDPD] Wireless Data Forum, CDPD System Specification, Release 1.1, 1995.

[CTC+97] H. Chang, C. Tait, N. Cohen, M. Shapiro, S. Mastrianni, R. Floyd, B. Housel, D. Lindquist, "Web Browsing in a Wireless Environment: Disconnected and Asynchronous Operation in ARTour Web Express," Proc. MobiCom'97, Budapest, Hungary, September 1997.

[FMSBMR98] D.C. Feldmeier, A.J. McAuley, J.M. Smith, D.S. Bakin, W.S. Marcus, T.M. Raleigh, "Protocol Boosters," IEEE Journal on Selected Areas of Communication, Vol. 16, No. 3, April 1998.

[FLASH] Flash Networks Ltd., performance boosting products technology vendor based in Holmdel, New Jersey. Website at http://www.flashnetworks.com.

[FOURELLE] Fourelle Systems, performance boosting products technology vendor based in Santa Clara, California. Website at http://www.fourelle.com.

[GPRS] ETSI, "General Packet Radio Service (GPRS): Service Description, Stage 2," GSM03.60, v.6.1.1, August 1998.

[GSM] M. Rahnema, "Overview of the GSM system and protocol architecture," IEEE Communications Magazine, Vol. 31, No. 4, pp. 92-100, April 1993.

[HNS] Hughes Network Systems, Inc., VSAT technology vendor based in Germantown, Maryland. Website at http://www.hns.com.

[I-TCP] A. Bakre, B.R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," Proc. 15th International Conference on Distributed Computing Systems (ICDCS), May 1995.

[KRA94] M. Kojo, K. Raatikainen, T. Alanko, "Connecting Mobile Workstations to the Internet over a Digital Cellular Telephone Network," Proc. Workshop on Mobile and Wireless Information Systems (MOBIDATA), Rutgers University, NJ, November 1994. Revised version published in Mobile Computing, pp. 253-270, Kluwer, 1996.

[KRLKA97] M. Kojo, K. Raatikainen, M. Liljeberg, J. Kiiskinen, T. Alanko, "An Efficient Transport Service for Slow Wireless Telephone Links," IEEE Journal on Selected Areas of Communication, Vol. 15, No. 7, September 1997.

[LAKLR95] M. Liljeberg, T. Alanko, M. Kojo, H. Laamanen, K. Raatikainen, "Optimizing World-Wide Web for Weakly-Connected Mobile Workstations: An Indirect Approach," Proc. of the 2nd Int. Workshop on Services in Distributed and Networked Environments, Whistler, Canada, pp. 132-139, June 1995.

[LHKR96] M. Liljeberg, H. Helin, M. Kojo, K. Raatikainen, "Mowgli WWW Software: Improved Usability of WWW in Mobile WAN Environments," Proc. IEEE Global Internet 1996 Conference, London, UK, November 1996.

[M-TCP] K. Brown, S. Singh, "M-TCP: TCP for Mobile Cellular Networks," ACM Computer Communications Review Volume 27(5), 1997. Available at ftp://ftp.ece.orst.edu/pub/singh/papers/mtcp.ps.gz.

[Pax99] V. Paxson, "End-to-End Internet Packet Dynamics," IEEE/ACM Transactions on Networking, Vol. 7, No. 3, 1999, pp. 277-292.

[PILCWEB] http://pilc.grc.nasa.gov.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1122] Braden, R., "Requirements for Internet Hosts -- Communications Layers", STD 3, RFC 1122, October 1989.

[RFC1144] Jacobson, V., "Compressing TCP/IP Headers for Low-Speed Serial Links", RFC 1144, February 1990.

[RFC1323] Jacobson, V., Braden, R. and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.

[RFC1958] Carpenter, B., "Architectural Principles of the Internet", RFC 1958, June 1996.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S. and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, October 1996.

[RFC2151] Kessler, G. and S. Shepard, "A Primer On Internet and TCP/IP Tools and Utilities", FYI 30, RFC 2151, June 1997.

[RFC2246] Dierk, T. and E. Allen, "TLS Protocol Version 1," RFC 2246, January 1999.

[RFC2393] Shacham, A., Monsour, R., Pereira, R. and M. Thomas, "IP Payload Compression Protocol (IPcomp)", RFC 2393, December 1998.

[RFC2401] Kent, S., and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[RFC2488] Allman, M., Glover, D. and L. Sanchez, "Enhancing TCP Over Satellite Channels using Standard Mechanisms", BCP 28, RFC 2488, January 1999.

[RFC2507] Degermark, M., Nordgren, B. and S. Pink, "IP Header Compression", RFC 2507, February 1999.

[RFC2508] Casner, S. and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", RFC 2508, February 1999.

[RFC2509] Engan, M., Casner, S. and C. Bormann, "IP Header Compression over PPP", RFC 2509, February 1999.

[RFC2663] Srisuresh, P. and Y. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[RFC2760] Allman, M., Dawkins, S., Glover, D., Griner, J., Henderson, T., Heidemann, J., Kruse, H., Ostermann, S., Scott, K., Semke, J., Touch, J. and D. Tran, "Ongoing TCP Research Related to Satellites", RFC 2760, February 2000.

[RFC3002] Mitzel, D., "Overview of 2000 IAB Wireless Internetworking Workshop", RFC 3002, December 2000.

[RFC3042] Allman, M., Balakrishnan, H. and S. Floyd, "Enhancing TCP's Loss Recovery Using Limited Transmit", RFC 3042, January 2001.

[SHEL00] Z. Shelby, T. Saarinen, P. Mahonen, D. Melpignano, A. Marshall, L. Munoz, "Wireless IPv6 Networks - WINE," IST Mobile Summit, Ireland, October 2000.

[SNOOP] H. Balakrishnan, S. Seshan, E. Amir, R. Katz, "Improving TCP/IP Performance over Wireless Networks," Proc. 1st ACM Conference on Mobile Communications and Networking (Mobicom), Berkeley, California, November 1995.

[SNOOPELN] H. Balakrishnan, R. Katz, "Explicit Loss Notification and Wireless Web Performance," Proc. IEEE Globecom 1998, Internet Mini-Conference, Sydney, Australia, November 1998.

[SPACENET] Spacenet, VSAT technology vendor based in Mclean, Virginia. Website at http://www.spacenet.com.

[SRC84] J.H. Saltzer, D.P. Reed, D.D. Clark, "End-To-End Arguments in System Design," ACM TOCS, Vol. 2, No. 4, pp. 277-288, November 1984.

[WAPARCH] Wireless Application Protocol Architecture Specification, April 1998, http://www.wapforum.org.

[WAPPROXY] Wireless Application Protocol Push Proxy Gateway Service Specification, August 1999, http://www.wapforum.org.

[WAPWAE] Wireless Application Protocol Wireless Application Environment Overview, March 2000, http://www.wapforum.org.

[WAPWDP] Wireless Application Protocol Wireless Datagram Protocol Specification, February 2000, http://www.wapforum.org.

[WAPWSP] Wireless Application Protocol Wireless Session Protocol Specification, May 2000, http://www.wapforum.org.

[WAPWTLS] Wireless Application Protocol Wireless Transport Layer Security Specification, February 2000, http://www.wapforum.org.

[WAPWTP] Wireless Application Protocol Wireless Transaction Protocol Specification, February 2000, http://www.wapforum.org.

[Zhang00] Y. Zhang, B. Singh, "A Multi-Layer IPsec Protocol," Proc. proceedings of 9th USENIX Security Symposium, Denver, Colorado, August 2000. Available at http://www.wins.hrl.com/people/ygz/papers/usenix00.html.

10. Authors' Addresses

Questions about this document may be directed to:

John Border
Hughes Network Systems
11717 Exploration Lane
Germantown, Maryland 20876

Phone: +1-301-548-6819
Fax: +1-301-548-1196
EMail: border@hns.com

Markku Kojo
Department of Computer Science
University of Helsinki
P.O. Box 26 (Teollisuuskatu 23)
FIN-00014 HELSINKI
Finland

Phone: +358-9-1914-4179
Fax: +358-9-1914-4441
EMail: kojo@cs.helsinki.fi

Jim Griner
NASA Glenn Research Center
MS: 54-5
21000 Brookpark Orad
Cleveland, Ohio 44135-3191

Phone: +1-216-433-5787
Fax: +1-216-433-8705
EMail: jgriner@grc.nasa.gov

Gabriel Montenegro
Sun Microsystems Laboratories, Europe
29, chemin du Vieux Chene
38240 Meylan, FRANCE

Phone: +33 476 18 80 45
EMail: gab@sun.com

Zach Shelby
University of Oulu
Center for Wireless Communications
PO Box 4500
FIN-90014
Finland

Phone: +358-40-779-6297
EMail: zach.shelby@ee.oulu.fi

Appendix A - PEP Terminology Summary

This appendix provides a summary of terminology frequently used during discussion of Performance Enhancing Proxies. (In some cases, these terms have different meanings from their non-PEP related usage.)

ACK filtering

Removing acknowledgments to prevent congestion of a low speed link, usually used with paths which include a highly asymmetric link. Sometimes also called ACK reduction. See Section 3.1.4.

ACK spacing

Delayed forwarding of acknowledgments in order to space them appropriately, for example, to help minimize the burstiness of TCP data. See Section 3.1.1.

application layer PEP

A Performance Enhancing Proxy operating above the transport layer. May be aimed at improving application or transport protocol performance (or both). Described in detail in Section 2.1.2.

asymmetric link

A link which has different rates for the forward channel (used for data segments) and the back (or return) channel (used for ACKs).

available bandwidth

The total capacity of a link available to carry information at any given time. May be lower than the raw bandwidth due to competing traffic.

bandwidth utilization

The actual amount of information delivered over a link in a given period, usually expressed as a percent of the raw bandwidth of the link.

gateway

Has several meanings with respect to PEPs, depending on context:

- An access point to a particular link;

- A device capable of initiating and terminating connections on

  behalf of a user or end system (e.g., a firewall or proxy).

Not necessarily, but could be, a router.

in flight (data)

Data sent but not yet acknowledged. More precisely, data sent for which the sender has not yet received the acknowledgement.

link layer PEP

A Performance Enhancing Proxy operating below the network layer.

local acknowledgement

The generation of acknowledgments by an entity in the path between two end systems in order to allow the sending system to transmit more data without waiting for end-to-end acknowledgments. Described (in the context of TCP) in Section 3.1.2.

performance enhancing proxy

An entity in the network acting on behalf of an end system or user (with or without the knowledge of the end system or user) in order to enhance protocol performance. Section 2 describes various types of performance enhancing proxies. Section 3 describes the mechanisms performance enhancing proxies use to improve performance.

raw bandwidth

The total capacity of an unloaded link available to carry information.

Snoop

A TCP-aware link layer developed for wireless packet radio and cellular networks. It works by caching segments at a wireless base station. If the base station sees duplicate acknowledgments for a segment that it has cached, it retransmits the missing segment while suppressing the duplicate acknowledgement stream being forwarded back to the sender until the wireless receiver starts to acknowledge new data. Described in detail in Section 5.3.2 and [SNOOP].

split connection

A connection that has been terminated before reaching the intended destination end system in order to initiate another connection towards the end system. This allows the use of different connection characteristics for different parts of the path of the originally intended connection. See Section 2.4.

TCP PEP

A Performance Enhancing Proxy operating at the transport layer with TCP. Aimed at improving TCP performance.

TCP splitting

Using one or more split TCP connections to improve TCP performance.

TCP spoofing

Sometimes used as a synonym for TCP PEP. More accurately, TCP spoofing refers to using transparent (to the TCP stacks in the end systems) mechanisms to improve TCP performance. See Section 2.1.1.

transparent

In the context of a PEP, transparent refers to not requiring changes to be made to the end systems, transport endpoints and/or applications involved in a connection. See Section 2.5 for a more detailed explanation.

transport layer PEP

A Performance Enhancing Proxy operating at the transport layer. Described in detail in Section 2.1.1.

tunneling

In the context of PEPs, tunneling refers to the process of wrapping a packet for transmission over a particular link between two PEPs. See Section 3.2.

WAP

The Wireless Application Protocol specifies an application framework and network protocols intended to work across differing narrow-band wireless network technologies. See Section 5.2.2.2.

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.