

Interactions of Realm Boundaries and End-to-End Network Applications

Mike Fisk[†]
mfisk@lanl.gov

Wu-chun Feng^{†§}
feng@lanl.gov

[†]Los Alamos National Laboratory *
Los Alamos, NM 87544

[§]Purdue University
West Lafayette, IN 47907

Abstract

One of the design principles of the Internet is that the network is made more flexible, and therefore useful, by placing functionality in end applications rather than in network infrastructure. Network gateways that violate this principle are considered harmful. This paper demonstrates that such upper-level gateways exist because of realm-specific performance, security, and protocol needs of certain portions of the Internet. Placing this functionality in end hosts is, conversely, harmful to the flexibility of using the Internet to link disparate networks. Requirements are developed for a protocol to allow end-hosts and gateways to negotiate the functionality of these gateways in terms of the needs of both end applications and network realms.

1 End-to-End Arguments

In [28], arguments are made for placing functionality close to the applications that use that functionality. Because the end applications have the most information regarding the given

problem space, they are in the best position to determine what functionality is required. Functionality provided by lower-level or intermediate systems may be redundant or contradictory with the needs of the application. With regard to networked or distributed applications, end-to-end design discourages the placement of functionality in network infrastructure and instead argues that functionality should be placed in the application whenever possible.

In practical terms, this design leads to the creation of software libraries of system functionality that applications may use. Some of these libraries may be implemented in an operating system kernel (as is usually done with TCP), while others may be implemented as user-space libraries (such as SSL). End-to-end arguments have played a key role in the design of the Internet and have arguably contributed to its success by keeping the network infrastructure relatively simple and allowing applications to evolve without the need for coordination with network infrastructure.

2 Boundary Gateways

As alluded to in its name, the end-to-end paradigm is concerned with two-party communications. With regard to network protocols, the network is treated as a passive, packet-delivery system that may use lower-layer head-

*This work was supported by the U.S. Dept of Energy's Next Generation Internet - Earth Systems Grid and Accelerated Strategic Computing Initiative - Distance and Distributed Computing and Communication programs through Los Alamos National Laboratory contract W-7405-ENG-36. This paper is Los Alamos Unclassified Report (LAUR) 00-3631.

ers in the packet for routing, but should never change the upper-layer portions of end-to-end packets.

Current trends in the Internet are leading to the creation of many mechanisms that do not fit within such a limited model of the network. This notion of the Internet as a homogenous, abstract cloud is not representative of its heterogeneous composition. Just as hosts on the Internet are a varied lot, the component networks on the Internet also vary in dimensions such as performance, link characteristics, and security. As the size and population of the Internet grows, further diversification can be expected.

Areas of similar network characteristics can be considered a *realm*. Any given network application may communicate across multiple realms. Further, the dynamic nature of Internet routing and mobile hosts means that realms may be added or removed from the path at any time. End applications currently have very few techniques for discovering the characteristics or policies of the realms they traverse. As a result, end applications fail to cater their traffic to most kinds of network conditions.

As a result, various types of upper-layer gateways have been created and placed at the boundaries between realms to manage the flow of traffic across those boundaries. This occurrence has had significant effects on end-to-end protocols. In an effort to generalize the functions and side-effects of these gateways, the following sections examine some of these upper-layer gateways that, in most cases, directly interfere with end-to-end protocol design and operation.

2.1 Network Address Translation

The Internet Protocol version 4 (IPv4)[24], provides a 32-bit address space for host addresses. Between the explosive growth in Internet sys-

tems and the mechanics of allocating addresses, it is now difficult for many institutions to get enough addresses for every system at that institution. Many institutions have deployed Network Address Translation (NAT) [8, 30] technology to allow hosts to share IPv4 addresses. Version 6 of the Internet Protocol (IPv6)[6] provides a 128-bit address space, but solutions such as NAT or proxying are needed to allow IPv6-only systems to be able to communicate with IPv4-only systems.

NAT is usually used as a gateway between the Internet and a private network of systems with non-global IP addresses. The NAT gateway translates between internal addresses and corresponding global addresses. Most commonly, TCP and UDP ports are also mapped so that multiple private addresses can be multiplexed into one global address. Thus, a site with numerous private addresses can survive with only a few global addresses.

Simple address translation breaks many upper layer protocols that transmit IP addresses (or ports in the case of port translation). For example, the FTP control connection communicates the IP address and TCP port to use for a data connection [26]. To work through a NAT gateway, the gateway must also translate the IP address and port within the FTP application protocol.

Since NAT requires modifications to the packet headers, packet security mechanisms such as IPsec Authenticated Headers [19] cannot be used. Encryption of the TCP headers and payload also breaks port translation and any applications that transmit their own IP addresses within the application layer.

While having obvious benefits, the use of NAT violates end-to-end principles by limiting end hosts to only using application protocols that are supported by network gateways beyond their control.

2.2 Performance Enhancing Proxies

Most Performance Enhancing Proxies[2] (PEPs) or Protocol Boosters [9] are proxies that, without the knowledge of end systems, modify or buffer packet streams in order to increase the performance of network protocols over network links with special characteristics such as satellite, cellular, and other wireless links. Some proxies modify packet streams to improve the behavior of transport protocols. Other proxies actually split the end-to-end connection into two separate connections that may even use entirely different protocols. Additional techniques include application-level caching and compression. The following sections describe some of the specific types of performance enhancing proxies and their interactions with end-to-end protocols.

2.2.1 Link Error Rates

Compared to copper and fiber-optic cables, most wireless networks are prone to packet loss or corruption. Internet protocols do not currently differentiate between these kinds of packet loss and losses due to congestion. TCP [25] and TCP-friendly [13] protocols will treat any packet loss as a sign of congestion and will halve their transmission rates. Between this throttling and the retransmissions, goodput will be very poor. In [1] goodput dropped 84% as the Poisson distributed bit error rate increased from 0 to 3.9×10^{-6} on a simulated 1.6Mbps link.

Consider the case of a gateway between a fiber-optic ‘backbone’ and a lossy wireless network such as a satellite link or cellular network. Assume that, like many wireless networks, this network will quickly detect errors or packet loss through mechanisms such as negative acknowledgements or duplicate ACK responses from the receiver.

To improve the performance of an end-to-end TCP connection, a new signalling mechanism could be created to allow the wireless network to notify the sender that the packet was lost due to reasons other than congestion. However, the sender will not retransmit the packet for at least the round-trip delay time of the Internet connection. With any significant network latency between the sender and the gateway, performance may still suffer.

Alternatively, the gateway can buffer TCP connections and handle retransmissions itself in a timely manner. The Snoop [1] library does this and also filters duplicate acknowledgement replies to the sender that would trigger retransmissions. For error rates above 5×10^{-7} , the traffic altered by Snoop performed between 1 and 20 times better than the unaltered, end-to-end traffic of a standard TCP Reno implementation.

To function, however, these Performance Enhancing Proxies must have access to the TCP headers. When IPsec encryption is used, the PEP will be of no benefit. Other end-to-end transport protocols will also not benefit from the proxy unless the PEP has implemented similar techniques for that specific protocol.

2.2.2 Delay-Bandwidth Product

With a windowing protocol such as TCP, full performance can only be achieved when the window is at least as large as the delay-bandwidth product of the network. With RFC 1323 [18], TCP can support window sizes up to 1 gigabyte (2^{30} octets), but not all hosts support this option. Further, clients that do support the option must have it manually configured. For many network connections, small window sizes are adequate, so most implementations default to relatively small values between 8 and 64 kilobytes. [29] suggests that the maximum possible window can always be used while [12] examines methods for dynami-

cally adjusting windows. However, virutally no systems actually use any of these mechanisms.

As a result, many operators of large delay-bandwidth networks place a Performance Enhancing Proxy at the entrance to that network. This gateway effectively splits TCP connections by acknowledging data entering the network and taking responsibility for reliably transmitting the data to the receiver. The sender receives the acknowledgements over the relatively low delay network and continues to send more data without filling its small window.

This kind of PEP does not require modifications to TCP packets, but must be able to examine TCP headers. Packet encryption techniques such as IPsec therefore negate the value of the proxy. Windowing protocols other than TCP are generally not supported.

2.3 Authenticating Proxies

It is common practice for a site to have a firewall that requires the authentication of incoming network connections. This authentication is frequently performed using an application layer proxy that forces in-band authentication. In many cases the proxy is *transparent*, meaning that there is no prior knowledge of the existence of a proxy. The end host initiates a connection to the destination host. The proxy pretends to be that destination host and, after authenticating, forwards data to the real destination.

2.4 Validating Proxies

While authentication can be used to identify the source of the data as being a trusted user, the user's application or computer may be under the control of somebody with malicious intentions. For example, the user may unknowingly send data that includes viruses or the application's network socket may be hijacked by

another user of the system [4]. As a result, many firewalls include application layer proxies that validate protocols and/or application content. Like authenticating proxies, many of these proxies are also transparent. As a simple example, a validating proxy can filter out malicious e-mail attachments such as those used by the 'Love Letter' worm [5]. A more sophisticated proxy could detect maliciously high rates of certain packets such as pings used in some distributed denial of service attacks [10].

2.5 Protocol Suites

While standard Internet protocols are quite ubiquitous, there is still value in being able to have application connectivity with other suites of protocols. Perhaps the best example of current alternative protocols is the the Wireless Application Protocol (WAP) suite [14] for wireless handheld devices. This suite of protocols has been optimized for cellular networks and devices, but was also designed to translate easily into Internet protocols. The level of support from the cellular industry makes it appear that WAP will be a widely used protocol suite. Thus, there is a need for multi-layer gateways at the boundary points between these WAP networks and the Internet.

3 Regaining End-to-End Control

In most of the previously described cases, the functionality of these boundary gateways does little to provide functionality (such as reliability or encryption) to end-to-end applications. Rather, they provide functionality to meet the requirements of the network realm that the traffic is traversing.

The end-to-end paradigm discourages such un-requested functionality in the network. It is

clear that with many protocols built with the expectation of end-to-end continuity, intermediate systems that violate that expectation are likely to cause problems. As a result, the typical solution is to modify the end system to include support for the realm or the boundary.

In [2] it is argued that when Performance Enhancing Proxies (PEPs) are used, they should be under the control of the end user so that they can be disabled when they interfere with end-to-end protocols such as IPsec. The authors know of no such control mechanisms for PEPs, but there are solutions for some other boundary gateways and realm-specific problems.

3.1 End-to-end Solutions

There are clear end-to-end solutions for some realm-specific problems. For example, TCP has been modified to include end-to-end support for the large window sizes necessary on networks with large delay-bandwidth products. Better host or application security practices could also remove the need for authenticating and validating proxies.

Unfortunately, there are host management and deployment issues that result in pragmatic reasons why some of these end-to-end solutions cannot be depended upon in the near future. The sheer number of hosts on the network make it unlikely that any feature will be deployed widely in a reasonable amount of time. Almost eight years after the TCP window scaling option was created, a March 2000 sampling of 115143 web servers showed that less than 41% of them supported the option [31]. Worse yet, while features are commonly bundled in new software releases, operational practices require cognizant change by the operator.

3.2 IP Tunneling Solutions

When the sender of a packet is aware of the existence and requirements of a gateway, the sender can encapsulate the end-to-end packet with additional information for the gateway. The gateway can perform its functions on this additional information and then forward the original, end-to-end packet. Tunneling mode IPsec and Realm Specific IP use IP over IP encapsulation in this manner.

IPsec tunneling [19] can be used to replace application-layer, authenticating proxies. Each packet is wrapped in an additional layer of authenticating IP headers. The gateway checks these headers, strips them off, and forwards the inner packet on to the destination.

Realm Specific IP (RSIP) [3] is a proposed standard protocol that allows a system to allocate a global address from a local gateway system. The internal system then uses that global address for outside communications, but tunnels the global traffic inside local packets to the gateway. The gateway strips the outer headers and sends the tunneled traffic onto the Internet. In the reverse direction, the gateway maintains a table of allocated addresses and tunnels incoming traffic to the appropriate internal system.

Tunneling solutions only work if the sender can include all information that the gateway needs. For authentication and address translation, this has been demonstrated to be possible. For performance enhancing proxies, validating proxies, and protocol translation, however, it is still necessary to examine upper-layer protocols. Wrapping IP packets with an additional layer of IP headers is practical, but duplicating TCP headers or content into an outer layer is clearly inefficient.

3.3 Negative Side-effects of IP Tunneling

Transport layer protocols are designed to work on real network links. Hiding those links by tunneling between distant end-points can hinder the ability of those transports to optimize themselves for that link. Consider a TCP connection operating over an IPsec tunnel between two gateways. It may be impossible for an intermediate router to apply normal traffic control and queuing policies to that traffic. For example, techniques are being developed for the explicit notification of loss, corruption, or congestion [27] of TCP packets. Over an IPsec tunnel, it may be impossible for the router to use those tools since the transport layer has been abstracted. In addition, local Performance Enhancing Proxies may also not be able to benefit those connections.

The end goal is end-to-end continuity for applications, not for IP or transport protocols. IP over IP tunneling effectively promotes IP to a transport protocol, but it lacks much of the flow and congestion control infrastructure necessary for such a task.

3.4 Application Tunneling Solutions

IP tunneling solutions can be described as a push-down approach. Information for gateways is pushed down into lower layer headers. The result is multiple IP layers in the packet. There exists an obvious corollary to this design. Gateway information can be pushed up into an upper-layer protocol. The result in this model is multiple application layers.

The SOCKS [20] protocol does this by tunneling TCP and UDP connection setup, and sometimes data, within the SOCKS application protocol. Since SOCKS encapsulates distilled TCP, UDP, and IP information rather than a pre-formed IP packet, it does not provide the

same level of end-to-end continuity that tunneling provides and that is required for protocols like IPsec.

4 Multi-party Dependencies

The core premise of the end-to-end design paradigm is that functionality should be under direct control of the application rather than be a required part of an opaque, underlying component. The primary reasons behind this design are as follows:

Effective Implementation. The application possesses the most knowledge about the requirements of the current situation. It can therefore make the best tradeoffs between conflicting factors.

Agility. The network as a whole is more agile if new functionality can be deployed solely on the end systems without having dependencies on functionality in the network itself.

There are two conflicting concerns at play:

First, end applications need the flexibility to tailor network behavior to the needs of the application. This flexibility is hindered by rigidity in the network.

Second, the Internet is a dynamic and heterogeneous network. End applications are frequently not prepared to operate well with all possible networks and configurations. Gateways in the network can remedy some of this rigidity in applications by adding functionality to the network.

4.1 Dependence Inversion

The traditional solutions described earlier all involve placing knowledge in the end application about the requirements of the network. For completeness, the end applications must be prepared to handle the requirements of any realm that might exist in the Internet.

In this case we have created the same kind of interdependency that the end-to-end paradigm seeks to avoid. Instead of the applications depending on the network, the network now depends on the applications. Given the ratio of end systems to gateways, it is extremely difficult for a gateway to depend on specific functionality to be deployed in all end systems. The reason that these upper-layer gateways are used in the first place is that they offer solutions that do not depend on modifying the end applications.

To avoid creating this interdependency, we must evaluate the placement of functionality not in terms of a two-party, end-to-end connection, but in terms of a multi-party, end-to-end composition. In this model of a cooperative network, the concerns of realms being transited are given equal, first-class, stature with application needs.

4.2 General Solutions

Adding functionality to the network can reduce the flexibility that applications have in tailoring network behavior. The resolution of this conflict can best be made by allowing for functionality to be added to the network, but letting end applications tailor that functionality to their needs.

Realm Specific IP, IPsec tunnels, and SOCKS are point solutions for specific instances of this problem. There is much friction involved with designing and deploying a new control protocol,

so it is greatly desirable that a single protocol be generic to all types of network functionality. The following sections develop a description of those generic requirements.

5 Layering Issues

As exhibited by RSIP and IPsec tunneling, IP tunneling is a construct of general utility. However, it preserves the notion that everything above the IP layer should be opaque to network gateways. As shown earlier, this abstraction has negative side-effects. To avoid these negative side-effects, the tunneling protocol should send data over a transport protocol equivalent to that requested by the application. For example, if the end-to-end connection is TCP, the lower layers should be TCP/IP as in Figure 2.

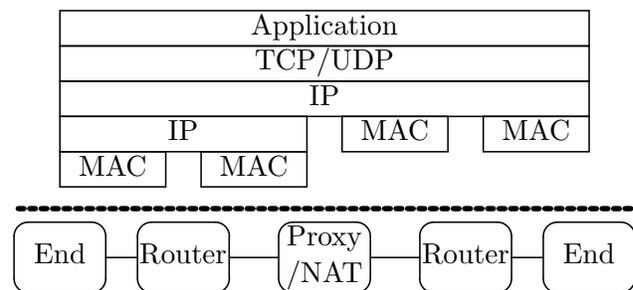


Figure 1: Layering of IP Tunneling

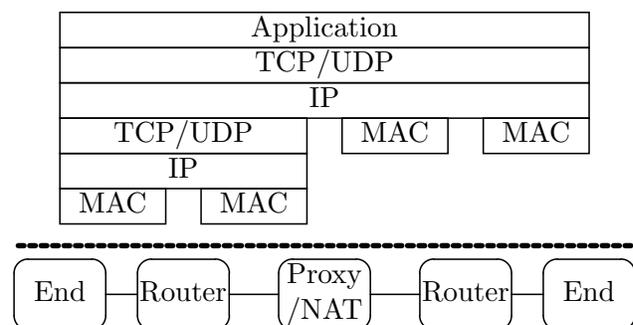


Figure 2: Layering of IP over TCP/IP

Operating a TCP/IP connection within a TCP/IP tunnel is redundant and competing congestion control mechanisms may have unintended effects on each other. Further, the lower layers establish a connection and appli-

cation data can be sent directly over that connection without intermediate TCP/IP segmentation. This arrangement of running application data directly over a single TCP/IP layer is shown in Figure 3. Packet-wise this construction is deceptively identical to a current upper-layer gateway that destroys end-to-end continuity.

There is, however, a semantic difference with all of these methods that the end-to-end application is cognizant of the upper-layer gateways and explicitly establishes the lower-layer connection to the gateway. There is effectively a session layer protocol operating between the application layer and the transport layer. This session layer is responsible for setting up the tunnel in the manner necessary to get connectivity for the end application. The HTTP protocol [11] includes a crude version of this kind of functionality for using HTTP proxies.

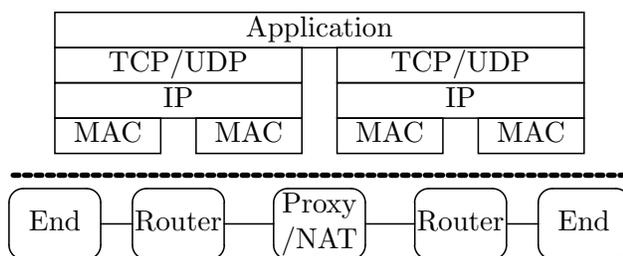


Figure 3: Layering of a Consecutive Composite

5.1 Consecutive Composition of Lower Layers

It is hard to argue that existing applications are strictly layered on top of TCP/IP. As mentioned earlier, FTP includes IP addresses and TCP ports in application data. Access control decisions are frequently based on port numbers, IP addresses, or IPsec information. In all of these cases, information about a lower layer is propagated up to higher layers.

There is a symbiotic relationship between end-to-end designs and fuzzy layering of the IP and TCP or UDP layers. With the assumption that

these layers are end-to-end, end-to-end application data can safely use information from lower layers. However, when it is not true that IP and TCP are end-to-end, applications that use lower layer information can cause problems for themselves.

However, there is one layer boundary that is strictly honored. Information from within the Medium Access Control (MAC) layer is not used for normal applications. Most applications have no knowledge of what kind of local network link is being used, much less what MAC addresses are being used.¹ This strict insulation is necessary since the MAC headers on a packet must change many times as a packet proceeds across a typical connection.

The strict separation at this layer gives evidence that it is not unreasonable to compose an application connection out of multiple, consecutive transport connections. Just as the MAC header is limited in scope to two adjacent routers, the IP and TCP/UDP layers may also be limited in scope to adjacent applications or upper-layer gateways.

6 A Session Signalling Protocol

In the process of describing the composition of multiple, consecutive transport protocols, the existence of a session layer protocol was mentioned. In summary, it is necessary to have a generic, extensible protocol that is applicable to all types of upper-layer gateways. The basic functionality should be provided at a higher level than the transport protocols (TCP, UDP, etc), but as a layer of functionality that appli-

¹Upper-layer protocols do exchange information about the MAC layer's path MTU (Maximum Transmission Unit) size. The MTU, however, is a part of the defined interface to the MAC layer rather than a private component of that layer. Layer 2 packets do not transmit MTU information.

cations can depend on. This functionality is consistent with a session-layer protocol. Below are the basic features of what we refer to as a *Session Signalling Protocol*.

6.1 Gateway Location and Negotiation

SOCKS, IPsec tunneling, and Realm Specific IP (RSIP) are all existing mechanisms for interacting with network gateways. Of the three, however, only RSIP addresses how end applications discover these gateways. In the case of RSIP, the Service Location Protocol (SLP) [16] is used. SLP, however, does not scale to finding intermediate gateways on wide-area networks under multiple domains of control.

To proceed with the model of composing consecutive transport and IP instances, it is necessary to have an automatic way to discover the gateways between two end hosts. All parties involved must then negotiate the necessary transport connections that will make-up the composition. To support alternate protocol suites such as WAP, the negotiation should focus on characteristics of the connection rather than specific protocols. As described below, there are multiple security aspects that must be included in this negotiation.

During the lifetime of a session, routing changes or movement of mobile agents may change the path through the network. The protocol must therefore determine when gateways are added or removed from the path. These changes to the gateway environment may also result in changes to the existing transport connections.

6.2 Encryption

Gateways that perform auditing or validation of application-layer data will most likely not allow end-to-end encryption. Performance En-

hancing Proxies may request unencrypted data as well, but should be willing to accept data that is encrypted from end-to-end. If the application is willing to operate without end-to-end encryption, it may still require that all traffic between validating gateways and/or end-hosts be encrypted.

End-to-end encryption of application data can be handled by a higher layer, but may possibly be conveniently provided by the session layer implementation on the end systems.

6.3 Authentication

The originator and authenticity of all data (in both directions) must be verifiable. Not all gateways may require this authentication, but the functionality must be present to support authenticating proxies. To support end-user authentication, the end-system implementation must be able to authenticate the user, either directly or through the calling application.

Applications may also wish to know the identity of a gateway before agreeing to pass unencrypted data through it. The application would presumably have some out-of-band mechanism for assigning a level of trust to the identified gateway.

6.4 Multiple Connections

There are certain applications that, like FTP, set-up secondary network connections. The session layer must allow these applications to refer to and identify these connections in an end-to-end manner. Further the session layer can signal to authenticating proxies that the secondary connection is part of the same session.

7 Additional Uses

In addition to allowing the different types of boundary gateways that were identified earlier, there are other uses for composing transport layer protocols with a Session Signalling Protocol. These other applications are those that are pursuing some of the same tunneling and proxying techniques to solve other problems such as IP mobility.

The Mobile IP protocol [23] takes application traffic using a canonical address and tunnels that traffic to a gateway on the home network. Different tunneling paths may have drastically different congestion and flow control characteristics, but with Mobile IP, the transport connection remains established. For example, if a mobile system moves from a high-bandwidth LAN to a low-bandwidth wireless link, the transport protocol may immediately flood the network with traffic. Most TCP implementations will only discover the loss in available bandwidth after congestion occurs.

In addition to solving the IP tunneling side effects mentioned in section 3.3, a session signalling protocol would cause mobile end systems to establish new transport protocols when the system changes local IP addresses. These transport protocols would use their normal mechanisms for discovering link characteristics in a friendly manner. Removing these transport connections from IP tunnels also allows them to benefit from any Performance Enhancing Proxies that are present.

8 Conclusions

The authors believe that the genericity of the concept of composing consecutive transport connections, together with a session layer protocol for building those compositions, would bring the Internet closer to the goals expressed

in the end-to-end arguments. It is true that networks absent of upper-layer gateways are more pure in end-to-end terms. However, it is not clear that it is practical to implement the functionality of those gateways on end systems. The fact that there are realms in the Internet, and the networks connecting to it, that are owned, operated, and legislated by diverse groups with differing environments has created inter-realm boundaries. Those boundaries affect end-to-end protocols and must be addressed. The concepts outlined in this paper form an infrastructure for making that multi-party negotiation possible.

The creation and deployment of a Session Signalling Protocol is not trivial. However, it could replace the implementation and deployment of several point solutions such as Realm Specific IP, SOCKS, Mobile IP, and other solutions not yet created. The authors intend to implement a Session Signalling Protocol and experiment with its use in the Globus middleware infrastructure [15] for grid computing. In the process, existing protocols will be leveraged whenever possible. Specifically, many of the features of such a protocol are already provided by protocols such as SOCKS, IPsec, Transport Layer Security (TLS) [7], Group Secure Association Key Management Protocol [17] and authentication frameworks such as the Simple Authentication and Security Layer [22] and the Generic Security Service Application Program Interface [21].

References

- [1] Hari Balakrishnan, Srinivasan Seshan, and Randy H. Katz. Improving reliable transport and handoff performance in cellular wireless networks. *Wireless Networks*, 1(4):469–481, 1995.
- [2] J. Border, M. Kojo, Jim Griner, and G. Montenegro. IETF draft-ietf-pilc-pep-

- 02.txt: Performance Enhancing Proxies, March 2000.
- [3] M. Borella and J. Lo. IETF draft-ietf-nat-rsip-framework-04.txt: Realm specific IP: Framework, March 2000.
- [4] CERT. CERT advisory CA-95:01: IP spoofing attacks and hijacked terminal connections, January 1995.
- [5] CERT. CERT advisory CA-2000-04: Love letter worm, May 2000.
- [6] S. Deering and R. Hinden. RFC 2460: Internet Protocol, Version 6 (IPv6) specification, December 1998.
- [7] O. T. Dierks and C. Allen. RFC 2246: The TLS protocol version 1, January 1999. Status: PROPOSED STANDARD.
- [8] K. Egevang and P. Francis. RFC 1631: The IP Network Address Translator (NAT), May 1994.
- [9] D.C. Feldmeier, A.J. McAuley, J.M. Smith, D.S. Bakin, W.S. Marcus, and T.M. Raleigh. Protocol boosters. *IEEE Journal on Selected Areas of Communications*, April 1998.
- [10] P. Ferguson and D. Senie. RFC 2267: Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing, January 1998. Status: INFORMATIONAL.
- [11] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. RFC 2068: Hypertext Transfer Protocol — HTTP/1.1, January 1997. Status: PROPOSED STANDARD.
- [12] Mike Fisk and Wu chun Feng. Dynamic adjustment of tcp window sizes. Technical Report LAUR 00-3321, Los Alamos National Laboratory, July 2000.
- [13] Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widmer. Equation-based congestion control for unicast applications. In *SIGCOMM*, August 2000.
- [14] Wireless Application Protocol Forum. *Wireless Application Protocol Architecture Specification*, April 1998. <http://www.wapforum.org/>.
- [15] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, Summer 1997.
- [16] E. Guttman, C. Perkins, J. Veizades, and M. Day. RFC 2608: Service Location Protocol, version 2, June 1999.
- [17] H. Harney, A. Colegrove, E. Harder, U. Meth, and R. Fleischer. IETF draft-harney-sparta-gsakmp-sec-01.txt: Group Secure Association Key Management Protocol, May 2000.
- [18] V. Jacobson, R. Braden, and D. Borman. RFC 1323: TCP extensions for high performance, May 1992.
- [19] S. Kent and R. Atkinson. RFC 2401: Security architecture for the Internet Protocol, November 1998.
- [20] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. RFC 1928: SOCKS protocol version 5, April 1996. Status: PROPOSED STANDARD.
- [21] J. Linn. RFC 2078: Generic Security Service Application Program Interface, version 2, January 1997.
- [22] J. Myers. RFC 2222: Simple Authentication and Security Layer (SASL), October 1997.
- [23] C. Perkins. RFC 2002: IP mobility support, October 1996.
- [24] J. Postel. RFC 791: Internet Protocol, September 1981.
- [25] J. Postel. RFC 793: Transmission Control Protocol, September 1981.
- [26] J. Postel and J. K. Reynolds. RFC 959: File Transfer Protocol, October 1985.

- [27] K. Ramakrishnan and S. Floyd. RFC 2481: A proposal to add Explicit Congestion Notification (ECN) to IP, January 1999.
- [28] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *TOCS*, 2(4):277–288, November 1984. Revised version of a paper from the Second International Conference on Distributed Computing Systems, Paris, France, April 8-10, 1981, pp. 509-512.
- [29] J. Semke, J. Mahdavi, and M. Mathis. Automatic TCP buffer tuning. In *Proceedings of ACM SIGCOMM '98*, pages 315–323. ACM Press, 1998.
- [30] P. Srisuresh and M. Holdrege. RFC 2663: IP Network Address Translator (NAT) terminology and considerations, August 1999.
- [31] Richard Wendland. A question about the deployment of SACK and NewReno TCP, March 2000. E-mail to IETF end2end-interest@isi.edu list.