

The Internet Protocol Journal

June 1998

Volume 1, Number 1

*A Quarterly Technical Publication for
Internet and Intranet Professionals*

In This Issue

From the Editor	1
What Is a VPN?—Part I	2
SSL: Foundation for Web Security	20
Call for Papers	30
Book Reviews	31
Fragments	35

From The Editor

Welcome to the first edition of *The Internet Protocol Journal* (IPJ). This publication is designed to bring you in-depth technical articles on current and emerging Internet and intranet technologies. We will publish technology tutorials, as well as case studies on all aspects of internetworking.

Our first article is a detailed look at *Virtual Private Networks* (VPNs). Many organizations are turning to VPNs as a cost-effective way to implement enterprise networking, but the industry has not yet settled for a single approach, nor even a single definition of the VPN concept. The article by Paul Ferguson and Geoff Huston is in two parts. Part II will follow in our second issue, due out in September.

When the Internet Protocol suite (TCP/IP) was first designed, security was not a major consideration. Indeed, the primary goal in the early days of networking was sharing of information among academics and researchers. Today, TCP/IP is being used for mission-critical applications and for the emerging area of electronic commerce. As a result, security mechanisms are being added at all levels of the protocol stack. In this issue, we take a closer look at the *Secure Sockets Layer* (SSL), which is used for Web transactions. William Stallings explains how SSL works and how it is becoming the standard for Web security.

If you want to learn about computer networks, many options are available, including conferences, journals, standards documents, Web sites, glossaries and, of course, books. Our *Fragments* page gives you some pointers for further reading, and every issue will include at least one book review.

A detailed description of the scope of this journal can be found on page 30 in our *Call for Papers*. We want your input in this new publication. Please send comments, suggestions or questions to ipj@cisco.com. You may also use this address to request a complimentary copy of the next issue of IPJ. If you would like to write an article, send me e-mail and I will send you author guidelines.

—Ole J. Jacobsen, Editor and Publisher

ole@cisco.com

To reserve your complimentary
copy of the next issue of
The Internet Protocol Journal,
please complete and return the
attached postage-paid card.

What Is a VPN? — Part I

by Paul Ferguson, Cisco Systems
and Geoff Huston, Telstra

The term “VPN,” or *Virtual Private Network*, has become almost as recklessly used in the networking industry as has “QoS” (Quality of Service) to describe a broad set of problems and “solutions,” when the objectives themselves have not been properly articulated. This confusion has resulted in a situation where the popular trade press, industry pundits, and vendors and consumers of networking technologies alike generally use the term VPN as an offhand reference for a set of different technologies. This article provides a common-sense definition of a VPN, and an overview of different approaches to building one.

“The wonderful thing about virtual private networks is that its myriad definitions give every company a fair chance to claim that its existing product is actually a VPN. But no matter what definition you choose, the networking buzz-phrase doesn’t make sense. The idea is to create a private network via tunneling and/or encryption over the public Internet. Sure, it’s a lot cheaper than using your own frame relay connections, but it works about as well as sticking cotton in your ears in Times Square and pretending nobody else is around.”^[1]

A Common-Sense Definition

As *Wired Magazine* notes in the quotation, the myriad definitions of a VPN are less than helpful in this environment. Accordingly, it makes sense to begin this examination of VPNs to see if it is possible to provide a common-sense definition of a VPN. Perhaps the simplest method of attempting to arrive at a definition for VPNs is to look at each word in the acronym individually, and then tie each of them together in a simple, common-sense, and meaningful fashion.

Let’s start by examining the word “network.” This term is perhaps the least difficult one for us to define and understand, because the commonly accepted definition is fairly uncontroversial and generally accepted throughout the industry. A network consists of any number of devices that can communicate through some arbitrary method. Devices of this nature include computers, printers, routers, and so forth, and they may reside in geographically diverse locations. They may communicate in numerous ways because the electronic signaling specifications, and data-link, transport, and application-layer protocols are countless. For the purposes of simplicity, let’s say that a “network” is a collection of devices that can communicate in some fashion, and can successfully transmit and receive data among themselves.

The term “private” is fairly straightforward, and is intricately related to the concept of “virtualization” insofar as VPNs are concerned, as we’ll discuss in a moment. In the simplest of definitions, “private” means communications between two (or more) devices is, in some

fashion, secret—that the devices that are not participating in the “private” nature of communications are not privy to the communicated content, and that they are indeed completely unaware of the private relationship altogether. Accordingly, data privacy and security (data integrity) are also important aspects of a VPN that need to be considered when implementing any particular VPN.

Another means of expressing this definition of “private” is through its antonym, “public.” A “public” facility is one that is openly accessible, and is managed within the terms and constraints of a common public resource, often via a public administrative entity. By contrast, a private facility is one where access is restricted to a defined set of entities, and third parties cannot gain access. Typically, the private resource is managed by the entities who have exclusive right of access. Examples of this type of private network can be found in any organizational network that is not connected to the Internet, or to any other external organizational network, for that matter. These networks are private because there is no external connectivity, and thus no external network communications.

Another important aspect of privacy in a VPN is through its technical definition. For example, privacy in an addressing and routing system means that the addressing used within a VPN community of interest is separate and discrete from that of the underlying shared network, and from that of other VPN communities. The same holds true for the routing system used within the VPN and that of the underlying shared network. The routing and addressing scheme within a VPN should, in general, be self-contained, but this scenario degenerates into a philosophical discussion of the context of the term “VPN.” Also, it is worthwhile to examine the differences between the “peer” and “overlay” models of constructing VPNs—both of which are discussed in more detail later under the heading “Network-Layer VPNs.”

“Virtual” is a concept that is slightly more complicated. *The New Hacker’s Dictionary* (formerly known as the Jargon File)^[2] defines virtual as:

virtual /adj./ [via the technical term “virtual memory,” prob. from the term “virtual image” in optics] 1. Common alternative to {logical}; often used to refer to the artificial objects (like addressable virtual memory larger than physical memory) simulated by a computer system as a convenient way to manage access to shared resources. 2. Simulated; performing the functions of something that isn’t really there. An imaginative child’s doll may be a virtual playmate. Oppose {real}.

Insofar as VPNs are concerned, the second definition is perhaps the most appropriate comparison for virtual networks. The “virtualization” aspect is one that is similar to what we briefly described previously as private, but the scenario is slightly modified—the private communication is now conducted across a network infrastructure that

is shared by more than a single organization. Thus, the private resource is actually constructed by using the foundation of a logical partitioning of some underlying common, shared resource rather than by using a foundation of discrete and dedicated physical circuits and communications services. Accordingly, the private network has no corresponding private physical communications system. Instead, the private network is a virtual creation that has no physical counterpart.

The virtual communications between two (or more) devices is because the devices that are not participating in the virtual communications are not privy to the content of the data, and they are also altogether unaware of the private relationships between the virtual peers. The shared network infrastructure could, for example, be the global Internet and the number of organizations or other users not participating in the virtual network may literally number into the thousands or even millions.

A VPN can also said to be a discrete network^[3]:

(discrete \dis*crete", a. [L. discretus, p.p. of discernere. See Discreet.]
1. Separate; distinct; disjunct).

The discrete nature of VPNs allows both privacy and virtualization. Although VPNs are not completely separate, intrinsically, the distinction is that they operate in a discrete fashion across a shared infrastructure, providing exclusive communications environments that do not share any points of interconnection.

The combination of these terms produces VPN—a private network, where the privacy is introduced by some method of virtualization. A VPN could be built between two end systems or between two organizations, between several end systems within a single organization or between multiple organizations across the global Internet, between individual applications, or any combination.

It should be noted that there is really no such thing as a nonvirtual network, if the underlying common public transmission systems and other similar public infrastructure components are considered to be the base level of carriage of the network. What separates a VPN from a truly private network is whether the data transits a shared versus a nonshared infrastructure. For instance, an organization could lease private line circuits from various telecommunications providers and build a private network on the base of these private circuit leases, but the circuit-switched network owned and operated by the telecommunications companies are actually circuits connected to their *Digital Access and Crossconnect Systems* (DACSS) network and subsequently their fiber-optics infrastructure. This infrastructure is shared by any number of organizations through the use of multiplexing technologies. Unless an organization is actually deploying private fiber and layered transmission systems, any network is layered with “virtualized” connectivity services in this fashion.

A VPN doesn't necessarily mean communications isolation, but rather the controlled segmentation of communications for communities of interest across a shared infrastructure.

The common and somewhat formal characterization of the VPN, and perhaps the most straightforward and strict definition, follows:

A VPN is a communications environment in which access is controlled to permit peer connections only within a defined community of interest, and is constructed through some form of partitioning of a common underlying communications medium, where this underlying communications medium provides services to the network on a nonexclusive basis.

A simpler, more approximate, and much less formal description follows:

A VPN is private network constructed within a public network infrastructure, such as the global Internet.

It should also be noted that although VPNs may be constructed to address any number of specific business needs or technical requirements, a comprehensive VPN solution provides support for dial-in access, support for multiple remote sites connected by leased lines (or other dedicated means), the ability of the VPN service provider (SP) to "host" various services for the VPN customers (for example, Web hosting), and the ability to support not just intra-, but also inter-VPN connectivity, including connectivity to the global Internet.

VPN Motivations

There are several motivations for building VPNs, but a common thread is that they all share the requirement to "virtualize" some portion of an organization's communications—in other words, make some portion (or perhaps all) the communications essentially "invisible" to external observers, while taking advantage of the efficiencies of a common communications infrastructure.

The base motivation for VPNs lies in the economics of communications. Communications systems today typically exhibit the characteristic of a high fixed-cost component, and smaller variable-cost components that vary with the transport capacity, or bandwidth, of the system. Within this economic environment, it is generally financially attractive to bundle numerous discrete communications services onto a common, high-capacity communications platform, allowing the high fixed-cost components associated with the platform to be amortized over a larger number of clients. Accordingly, a collection of virtual networks implemented on a single common physical communications plant is cheaper to operate than the equivalent collection of smaller, physically discrete communications plants, each servicing a single network client.

Therefore, if aggregation of communications requirements leads to a more cost-effective communications infrastructure, why not pool all these services into a single public communications system? Why is there still the requirement to undertake some form of partitioning within this common system that results in these “virtual private” networks?

In response to this question, the second motivation for VPNs is that of communications privacy, where the characteristics and integrity of communications services within one closed environment is isolated from all other environments that share the common underlying plant. The level of privacy depends greatly on the risk assessment performed by the subscriber organization—if the requirement for privacy is low, then the simple abstraction of discretion and network obscurity may serve the purpose. However, if the requirement for privacy is high, then there is a corresponding requirement for strong security of access and potentially strong security applied to data passed over the common network.

History

This article cannot do justice to the concept of VPNs without some historical perspective, so we need to look at why VPNs are an evolving paradigm, and why they will continue to be an issue of confusion, contention, and disagreement. This examination is important because opinions on VPN solutions are quite varied, as well as how they should be approached.

Historically, one of the precursors to the VPN was the *Public Data Network* (PDN), and the current familiar instance of the PDN is the global Internet. The Internet creates a ubiquitous connectivity paradigm, where the network permits any connected network entity to exchange data with any other connected entity. The parallels with the global *Public Switched Telephone Network* (PSTN) are, of course, all too obvious—where a similar paradigm of ubiquitous public access is the predominate characteristic of the network.

The Public Data Network has no inherent policy of traffic segregation, and any modification to this network policy of permitting ubiquitous connectivity is the responsibility of the connecting entity to define and enforce. The network environment is constructed using a single addressing scheme and a common routing hierarchy, which allows the switching elements of the network to determine the location of all connected entities. All these connected entities also share access to a common infrastructure of circuits and switching.

However, the model of ubiquity in the “Internet PDN” does not match all potential requirements, especially the need for data privacy. For organizations that wish to use this public network for private purposes within a closed set of participants (for example, connecting a set of geographically separated offices), the Internet is not always a palatable possibility. Numerous factors are behind this mismatch, including issues of Quality of Service (QoS), availability and reliability, use of

public addressing schemes, use of public protocols, site security, and data privacy and integrity (the possibility of traffic interception). Additionally, a corporate network application may desire more stringent levels of performance management than are available within the public Internet, or indeed may wish to define a management regime that differs from that of the underlying Internet PDN.

Service-Level Agreements

It is worthwhile at this point to briefly examine the importance of *Service-Level Agreements* (SLAs) in regards to the deployment of VPNs. SLAs are negotiated contracts between VPN providers and their subscribers; they contain the service criteria to which the subscriber expects specific services to be delivered. The SLA is arguably the only binding tool at the subscriber's disposal with which to ensure that the VPN provider delivers the service(s) to the level and quality as agreed, and it is in the best interest of the subscribers to monitor the criteria outlined in the SLA for compliance. However, SLAs present some challenging technical issues for both the provider and the subscriber.

For the subscriber, the challenge is to devise and operate service measurement tools that can provide a reasonable indication as to what extent the SLA is being honored by the provider. Also, it should be noted that a subscriber may use an SLA to bind one or more providers to a contractual service level, but if the subscriber's VPN spans multiple providers' domains, the SLA must also encompass the issue of provider interconnection and the end-to-end service performance.

For the provider, the challenge lies in honoring multiple SLAs from a number of service providers. In the case of an Internet PDN provider, the common mode of best-effort service levels is not conducive to meeting SLAs, given the unpredictable nature of the host's resource allocation mechanisms. In such environments, the provider either has to ensure that the network is generously engineered in terms of the ratio of subscriber access capacity to internal switching capacity, or the provider can deploy service differentiation structures to ensure that minimum resource levels are allocated to each SLA subscriber. It must be noted that the former course of action does tend to reduce the benefit of aggregation of traffic, which in turn has an ultimate cost implication, while the latter course of action has implications in terms of operational management complexity and scalability of the network.

Alternatives to the VPN

The alternative to using the Internet as a VPN today is to lease circuits, or similar dedicated communications services, from the public network operators (the local telephone company in most cases), and create a completely private network. It is a layering convention that allows us to label this as "completely private," because these dedicated communications services are (at the lower layers of the protocol

stack) again instances of virtual private communications systems constructed atop a common transmission bearer system. Of course, this scenario is not without precedent, and it must be noted that most of the early efforts in data networking, and many of the current data networking architectures, do not assume a deployment model of ubiquitous public access.

It is interesting to note that this situation is odd, when you consider that the inherent value of an architecture where ubiquitous public access over a chaotic collection of closed private networks had been conclusively demonstrated in the telephony marketplace since the start of the 20th century. Although the data communications industry appears to be moving at a considerable technological pace, the level of experiential learning, and consequent level of true progress as distinct from simple motion, still leaves much to be desired!

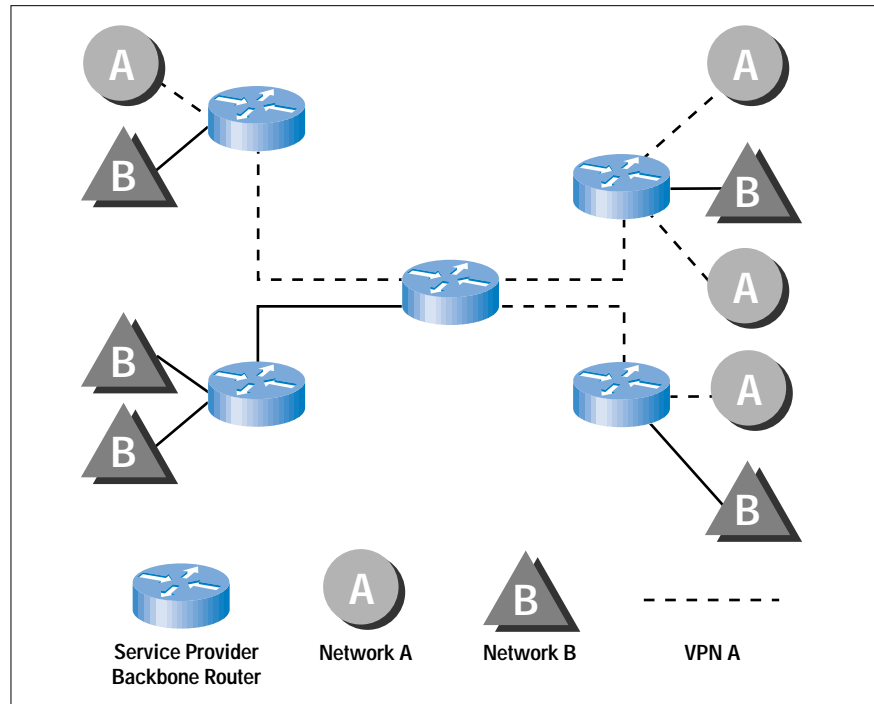
Instead of a public infrastructure deployment, the deployment model used has been that of a closed (or private) network environment where the infrastructure, addressing scheme, management, and services were dedicated to a closed set of subscribers. This model matched that of a closed corporate environment, where the network was dedicated to serve a single corporate entity as the sole client. This precursor to the VPN, which could be called the private data network, was physically constructed using dedicated local office wiring and dedicated leased circuits (or private virtual circuits from an underlying switching fabric such as X.25) to connect geographically diverse sites.

However, this alternative does have an associated cost, in that the client now has to manage the network and all its associated elements, invest capital in network switching infrastructure, hire trained staff, and assume complete responsibility for the provisioning and ongoing maintenance of the network service. Such a dedicated use of transport services, equipment, and staff is often difficult to justify for many small-to-medium sized organizations, and whereas the functionality of a private network system is required, the expressed desire is to reduce the cost of the service through the use of shared transport services, equipment, and management. Numerous scenarios can address this need, ranging from outsourcing the management of the switching elements of the network (managed network services), to outsourcing the capital equipment components (leased network services), to outsourcing the management, equipment, and transport elements to a service provider altogether.

An Example VPN

In the simple example illustrated in Figure 1, Network “A” sites have established a VPN (depicted by the dashed lines) across the service provider’s backbone network, where Network “B” is completely unaware of its existence. Both Networks “A” and “B” can harmoniously coexist on the same backbone infrastructure.

Figure 1:
A Virtual Private
Network of
"A" Sites



This type of VPN is, in fact, the most common type of VPN—one that has geographically diverse subnetworks that belong to a common administrative domain, interconnected by a shared infrastructure outside their administrative control (such as the global Internet or a single service provider backbone). The principal motivation in establishing a VPN of this type is that perhaps most of the communications between devices within the VPN community may be sensitive (again, a decision on the level of privacy required rests solely on a risk analysis performed by the administrators of the VPN), yet the total value of the communications system does not justify the investment in a fully private communications system that uses discrete transmission elements.

On a related note, the level of privacy that a VPN may enjoy depends greatly on the technology used to construct the VPN. For example, if the communications between each VPN subnetwork (or between each VPN host) is securely encrypted as it transits the common communications infrastructure, then it can be said that the privacy aspect of the VPN is relatively high.

In fact, the granularity of a VPN implementation can be broken down further to a single end-to-end, one-to-one connectivity scenario. Examples of these types of one-to-one VPNs are single dialup users who establish a VPN connection to a secure application, such as an online banking service, or a single user establishing a secure, encrypted session between a desktop and server application, such as a purchasing transaction conducted on the World Wide Web. This type of one-to-one VPN is becoming more and more prevalent as secure electronic commerce applications become more mature and are further deployed in the Internet. (See article starting on page 20.)

It is interesting to note that the concept of virtualization in networking has also been considered in regard to deploying both research and production services on a common infrastructure. The challenge in the research and education community is one in which there is a need to satisfy both network research and production requirements. VPNs have also been considered as a method to segregate traffic in a network such that research and production traffic behave as “ships in the night,” oblivious to one another’s existence, to the point that major events (for example, major failures, instability) within one community of interest are completely transparent to the other. This concept is further documented in MORPHnet^[4].

It should also be noted that VPNs may be constructed to span more than one host communications network, so that the “state” of the VPN may be supported on one or more VPN provider networks. This scenario is perhaps at its most robust when all the providers explicitly support the resultant distributed VPN environment, but other solutions that do not necessarily involve knowledge of the overlay VPN are occasionally deployed with mixed results.

Types of VPNs

The confusion factor comes into play in the most basic discussions regarding VPNs, principally because there are actually several different types of VPNs, and depending on the functional requirements, several different methods of constructing each type of VPN are available. The process of selection should include consideration of what problem is being solved, risk analysis of the security provided by a particular implementation, issues of scale in growing the size of the VPN, and the complexity involved in implementation of the VPN, as well as ongoing maintenance and troubleshooting.

To simplify the description of the different types of VPNs, they are broken down in this article into categories that reside in the different layers of the TCP/IP protocol suite; Link Layer, Network Layer, Transport Layer, and Application Layer.

Network-Layer VPNs

The network layer in the TCP/IP protocol suite consists of the IP routing system—how reachability information is conveyed from one point in the network to another. There are a few methods to construct VPNs within the network layer—each is examined in the following paragraphs. A brief overview of non-IP VPNs is provided in Part II of this article.

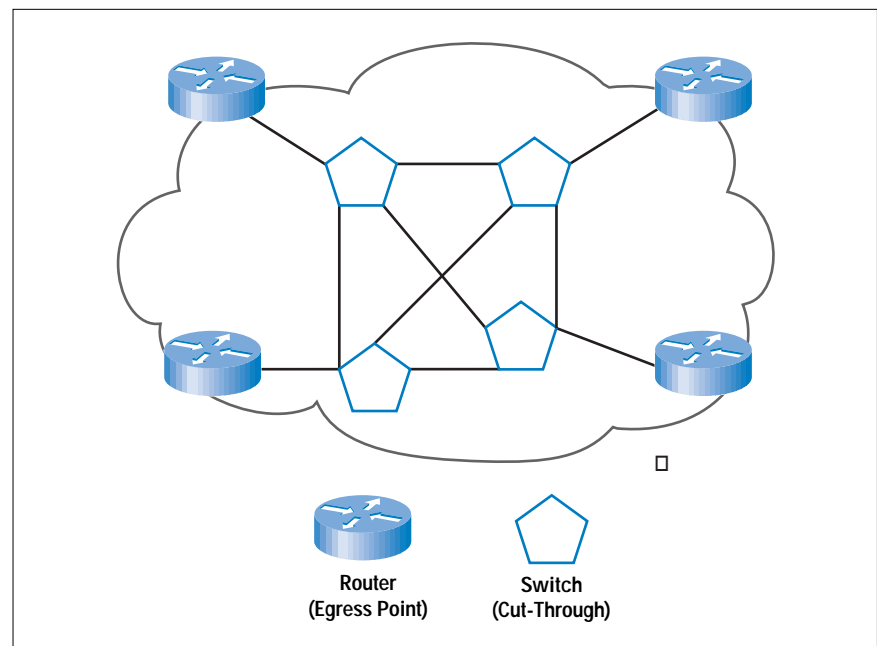
A brief overview of the differences in the “peer” and “overlay” VPN models is appropriate at this point. Simply put, the “peer” VPN model is one in which the network-layer forwarding path computation is done on a hop-by-hop basis, where each node in the intermediate data transit path is a peer with a next-hop node. Traditional routed networks are examples of peer models, where each router in the network

path is a peer with its next-hop adjacencies. Alternatively, the “overlay” VPN model is one in which the network-layer forwarding path is not done on a hop-by-hop basis, but rather, the intermediate link-layer network is used as a “cut-through” to another edge node on the other side of a large cloud. Examples of “overlay” VPN models include ATM, Frame Relay, and tunneling implementations.

Having drawn these simple distinctions between the peer and overlay models, it should be noted that the overlay model introduces some serious scaling concerns in cases where large numbers of egress peers are required because the number of adjacencies increases in direct proportion to the number of peers—the amount of computational and performance overhead required to maintain routing state, adjacency information, and other detailed packet forwarding and routing information for each peer becomes a liability in very large networks. If all the egress nodes in a cut-through network become peers in an effort to make all egress nodes one “Layer 3” hop away from one another, the scalability of the VPN overlay model is limited quite remarkably.

For example, as the simple diagram (Figure 2) illustrates, the routers that surround the interior switched infrastructure represent egress peers, because the switches in the core interior could be configured such that all egress nodes are one Layer 3 hop away from one another, creating what is commonly known as a “cut-through.” This scenario forms the foundation of an overlay VPN model.

Figure 2:
A Cut-Through VPN

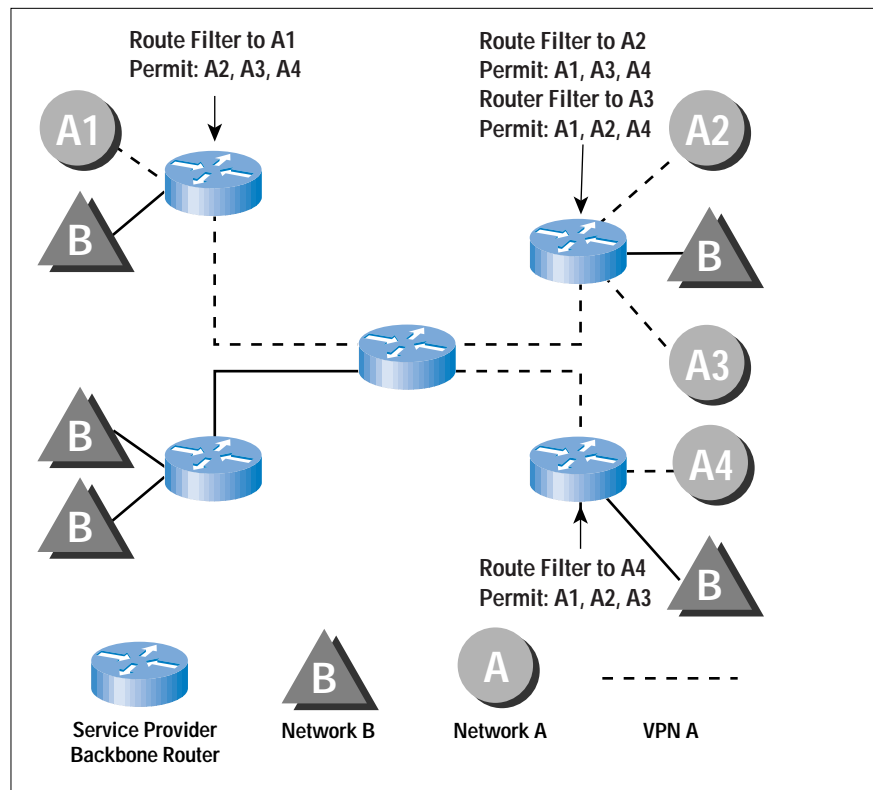


Alternatively, if the switches in the interior are replaced with routers, then the routers positioned at the edge of the cloud become peers with their next-hop router nodes, not other egress nodes. This scenario forms the foundation of the peer VPN model.

Controlled Route Leaking

“Controlled route leaking” (or *route filtering*) is a method that could also be called “privacy through obscurity” because it consists of nothing more than controlling route propagation to the point that only certain networks receive routes for other networks that are within their own community of interest. This model can be considered a “peer” model, because a router within a VPN site establishes a routing relationship with a router within the VPN provider’s network, instead of an edge-to-edge routing peering relationship with routers in other sites of that VPN. Although the common underlying Internet generally carries the routes for all networks connected to it, this architecture assumes that only a subset of such networks form a VPN. The routes associated with this set of networks are filtered such that they are not announced to any other set of connected networks, and all other non-VPN routes are not announced to the networks of the VPN. For example, in Figure 1, if the SP routers “leaked” routing information received from one site in Network “A” to only other sites in Network “A,” then sites not in Network “A” (for instance, sites in Network “B”) would have no explicit knowledge of any other networks which were attached to the service provider’s infrastructure (as shown in Figure 3). Given this lack of explicit knowledge of reachability to any location other than other members of the same VPN, privacy of services is implemented by the inability of any of the VPN hosts to respond to packets which contain source addresses from outside the VPN community of interest.

Figure 3:
Controlled Route
Leaking



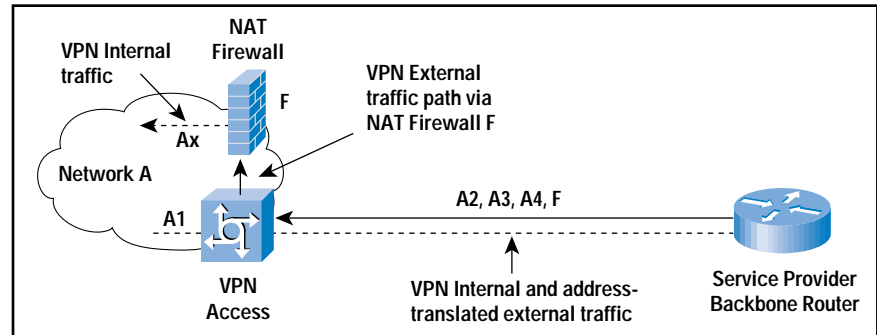
This use of partial routing information is prone to many forms of misconfiguration. One potential problem with route leaking is that it is extremely difficult, if not impossible, to prohibit the subscriber networks from pointing default to the upstream next-hop router for traffic destined for networks outside their community of interest. From within the VPN subscriber's context, this action may be reasonable, in that "default" for the VPN is reachability to all other members of the same VPN, and pointing a default route to the local egress path is, within a local context, a reasonable move. Thus, it is no surprise that this is a common occurrence in VPNs in which the customer configures and manages the customer premise equipment (CPE) routers. If the SP manages the configuration of the CPE routers, then this is rarely a problem. Otherwise, the SP might be wise to place traffic filters on first-hop routers to prohibit all traffic destined for networks outside the VPN community of interest.

It should also be noted that this environment implicitly assumes a common routing core. A common routing core, in turn, implies that each VPN must use addresses that do not clash with those of any other VPN on the same common infrastructure, and cannot announce arbitrary private addresses into the VPN. Another, perhaps less obvious, side effect of this form of VPN structure is that it is not possible for two VPNs to have a single point of interconnection, nor is it possible for a VPN to operate a single point of interconnection to the public Internet in such an environment. (This single point would be a so-called "gateway," where all external traffic is passed through a control point that can enforce some form of access policy and record a log of external transactions.) The common routing core uses a single routing paradigm, based solely on destination address.

It should also be noted that this requirement highlights one of the dichotomies of VPN architectures. VPNs must assume that they operate in a mutually hostile environment, where any vulnerability that exposes the private environment to access by external third parties may be exploited in a hostile fashion. However, VPNs rarely are truly isolated communications environments, and typically all VPNs do have some form of external interface that allows controlled reachability to other VPNs and to the broader public data network. The trade-off between secure privacy and the need for external access is a constant feature of VPNs.

Implementation of inter-VPN connectivity requires the network to route externally originated packets to the VPN interconnection point, and if they are admitted into the VPN at the interconnection point, the same packet may be passed back across the network to the ultimate VPN destination address. Without the use of *Network Address Translation* (NAT) technologies at the interconnection point of ingress into the VPN, this kind of communications structure is insupportable within this architecture (Figure 4).

Figure 4:
Segregating VPN
traffic via address
translation



In general, the technique of supporting private communities of interest simply by route filtering can at best be described as a primitive method of VPN construction, which is prone to administrative errors, and admits an undue level of insecurity and network inflexibility. Even with comprehensive traffic and route filtering, the resulting environment is not totally robust. The operational overhead required to support complementary sets of traditional routing and traffic filters is a relevant consideration, and this approach does not appear to possess the scaling properties desirable to allow the number of VPNs to grow beyond the bounds of a few hundred, using today's routing technologies.

Having said that, however, a much more scalable approach is to use *Border Gateway Protocol (BGP) communities*^[5] as a method to control route propagation. The use of BGP communities scales much better than alternative methods with respect to controlling route propagation and is less prone to human misconfiguration. Briefly, the use of the BGP communities attribute allows a VPN provider to “mark” BGP *Network-Layer Reachability Information (NLRI)* with a community attribute, such that configuration control allows route information to propagate in accordance with a community profile.

Because traffic from different communities of interest must traverse a common shared infrastructure, there is no significant data privacy in the portion of the network where traffic from multiple communities of interest share the infrastructure. Therefore, it can be said that although connected subnetworks—or rather, subscribers to the VPN service—may not be able to detect the fact that there are other subscribers to the service, multiple interwoven streams of subscriber data traffic pass unprotected in the core of the service provider's network.

Tunneling

Sending specific portions of network traffic across a tunnel is another method of constructing VPNs. Some tunneling methods are more effective than others. The most common tunneling mechanisms are *Generic Routing Encapsulation (GRE)*^[6] tunneling between a source and destination router, router-to-router or host-to-host tunneling protocols such as *Layer 2 Tunneling Protocol (L2TP)*^[7] and *Point-to-Point Tunneling Protocol (PPTP)*^[8], and *Distance Vector Multicast Routing Protocol (DVMRP)*^[9] tunnels.

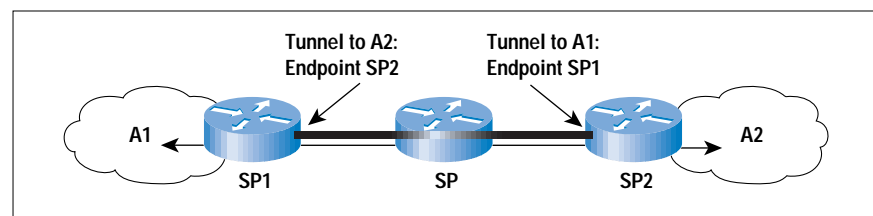
Tunneling can be considered an overlay model, but the seriousness of the scaling impact depends on whether the tunnels are point-to-point or point-to-multipoint. Point-to-point tunnels have fewer scaling problems than do point-to-multipoint tunnels, except in situations where a single node begins to build multiple point-to-point tunnels with multiple endpoints. Although a linear scaling problem is introduced at this point, the manageability of point-to-point tunnels lies solely in the administrative overhead and the number of the tunnels themselves. On the other hand, point-to-multipoint tunnels use “cut-through” mechanisms to make greater numbers of endpoints one hop away from one another and subsequently introduce a much more serious scaling problem.

Although the *Multicast Backbone* (Mbone) itself could literally be considered a global VPN, and although DVMRP tunnels are still widely used by organizations to connect to the Mbone, it really is not germane to the central topic of VPNs, because the focus of this article is on unicast traffic.

Traditional Modes of Tunneling

GRE tunnels, as mentioned previously, are generally configured between a source (*ingress*) router and a destination (*egress*) router, such that packets designated to be forwarded across the tunnel (already formatted with an encapsulation of the data with the “normal” protocol-defined packet header) are further encapsulated with a new header (the GRE header), and placed into the tunnel with a destination address of the tunnel endpoint (the new next-hop). When the packet reaches the tunnel endpoint, the GRE header is stripped away, and the packet continues to be forwarded to the destination, as designated in the original IP packet header (Figure 5).

Figure 5:
Tunneling across a
Service Provider



GRE tunnels are generally point-to-point—that is, there is a single source address for the tunnel and usually only a single destination tunnel endpoint. However, some vendor implementations allow the configuration of point-to-multipoint tunnels—that is, a single source address and multiple destinations. Although this implementation is generally used in conjunction with *Next Hop Resolution Protocol* (NHRP)^[10], the effectiveness and utility of NHRP is questionable and should be tested prior to deployment. It is also noteworthy that NHRP is known to produce steady-state forwarding loops when used to establish shortcuts between routers. In the scenario discussed previously, NHRP is used for establishing shortcuts between routers.

Tunnels, however, do have numerous compelling attractions when used to construct VPNs. The architectural concept is to create VPNs as a collection of tunnels across a common host network. Each point of attachment to the common network is configured as a physical link that uses addressing and routing from the common host network, and one or more associated tunnels. Each tunnel endpoint logically links this point of attachment to other remote points from the same VPN. The technique of tunneling uses a tunnel egress address defined within the address space of the common host network, whereas the packets carried within the tunnel use the address space of the VPN, which in turn constrains the tunnel endpoints to be collocated to those points in the network where the VPN and the host network interconnect.

Pros and Cons

The advantage of this approach is that the routing for the VPN is isolated from the routing of the common host network. The VPNs can reuse the same private address space within multiple VPNs without any cross impact, providing considerable independence of the VPN from the host network. This requirement is key for many VPNs in that private VPNs typically may not use globally unique or coordinated address space, and there is often the consequential requirement to support multiple VPNs which independently use the same address block. Such a configuration is not supportable within a controlled route leakage VPN architecture. The tunnel can also encapsulate numerous different protocol families, so that it is possible for a tunnel-based VPN to mimic much of the functionality of dedicated private networks. Again, the need to support multiple protocols in a format which preserves the functionality of the protocol is a critical requirement for many VPN support architectures. This requirement is one in which an IP common network with controlled route leakage cannot provide such services, whereas a tunneling architecture can segment the VPN-private protocol from the common host network. The other significant advantage of the tunneled VPN is the segregation of the common host routing environment with that of the VPN. To the VPN, the common host network assumes the properties of numerous point-to-point circuits, and the VPN can use a routing protocol across the virtual network which matches the administrative requirements of the VPN. Equally, the common host network can use a routing design which matches the administrative requirements of the host network (or collection of host networks), and is not constrained by the routing protocols used by the VPN client networks.

Although it could be said that these advantages indicate that GRE tunneling is the panacea for VPN design, using GRE tunnels as a mechanism for VPNs does have several drawbacks, mostly with regard to administrative overhead, scaling to large numbers of tunnels, and QoS and performance.

Since GRE tunnels must be manually configured, there is a direct relationship to the number of tunnels that must be configured and the amount of administrative overhead required to configure and maintain them—each time the tunnel endpoints must change, and they must be manually reconfigured. Also, although the amount of processing required to encapsulate a packet for GRE handling may appear to be small, there is a direct relationship to the number of configured tunnels and the total amount of processing overhead required for GRE encapsulation. Of course, tunnels can be structured to be triggered automatically, but such an approach has numerous drawbacks that dictate careful consideration of related routing and performance issues. The worst end state of such automatic tunnel generation is that of a configuration loop where the tunnel passes traffic over itself. It is important, once again, to reiterate the impact of a large number of routing peering adjacencies that result from a complete mesh of tunnels; this scenario can result in a negative effect on routing efficiency.

An additional concern with GRE tunneling is the ability of traffic classification mechanisms to identify traffic with a fine enough level of granularity, and not become a hindrance to forwarding performance. If the traffic classification process used to identify packets (that are to be forwarded across the tunnel) interferes with the router's ability to maintain acceptable packet-per-second forwarding rates, then this becomes a performance liability.

Privacy of the network remains an area of concern because the tunnel is still vulnerable—privacy is not absolute. Packets that use GRE formatting can be injected into the VPN from third-party sources. To ensure a greater degree of integrity of privacy of the VPN, it is necessary to deploy ingress filters that are aligned to the configured tunnel structure.

It is also necessary to ensure that the CPE routers are managed by the VPN service provider, because the configuration of the tunnel endpoints is a critical component of the overall architecture of integrity of privacy. However, most VPN service providers are reluctant to add CPE equipment to their asset inventory and undertake remote management of such CPE equipment, due to the high operational overheads and poor capital efficiencies which are typical of CPE deployment. Arguably, one might suggest that having a dedicated CPE router defeats one of the basic premises of constructing a VPN—the use of shared infrastructure as a way to reduce the overall network cost.

It should be noted that VPNs can be constructed using tunnels without the explicit knowledge of the host network provider, and the VPN can span numerous host networks without any related underlying agreements between the network operators to mutually support the overlay VPN. Such an architecture is little different from provider-operated VPN architecture; the major difference lies in the issue of traffic and

performance engineering, and the administrative boundary of the management of the VPN overlay. Independently configured VPN tunnels can result in injection of routes back into the VPN in a remote location, a scenario that can cause traffic to traverse the same link twice, once in an unencapsulated format and again within a tunnel. This situation can then lead to adverse performance impacts.

It is also true that the overlay VPN model has no control over which path is taken in the common host network, nor the stability of that path. This scenario can then lead to adverse performance impacts on the VPN. Aside from the technology aspects of this approach, the major issue is one of whether the VPN management is outsourced to the network provider, or undertaken within administrative functions of the VPN. One of the more serious considerations in building a VPN on tunneling is that there is virtually no way to determine the cost of the route across a tunnel, because the true path is masked by the cut-through nature of the tunnel. This situation could ultimately result in highly suboptimal routing, meaning that a packet could take a path determined by the cut-through mechanism that is excessively suboptimal, while native per-hop routing protocols might find a much more efficient method to forward the packets to their destinations.

Conclusion

So far in our discussion of VPNs, we have introduced a working definition of the term “Virtual Private Network” and discussed the motivations behind the adoption of such networks. We have outlined a framework for describing the various forms of VPNs, and then examined numerous network-layer VPN structures, in particular, that of controlled route leakage and tunneling techniques.

In Part II we will continue this examination of network-layer VPNs, including virtual private dial networks and network-layer encryption. In addition, we will examine link-layer VPNs that use ATM and Frame Relay substrates, and also look at switching and encryption techniques, and issues concerning QoS and non-IP VPNs.

References

- [1] Steinberg, Steve G., “Hype List—Deflating this month’s overblown memes.” *Wired Magazine*, 6.02, February 1998, p. 80. Ironically, number 1 on the Hype List is virtual private networks, with a life expectancy of 18 months.
- [2] Raymond, Eric S., compiler. *The New Hacker’s Dictionary, Third Edition*. MIT Press, ISBN 0-262-68092-0, 1996. The Jargon File online: <http://www.ccil.org/jargon/>
- [3] *Webster’s Revised Unabridged Dictionary* (1913). Hypertext Webster Gateway: http://work.ucsd.edu:5141/cgi-bin/http_webster

- [4] Aiken, R., R. Carlson, I. Foster, T. Kuhfuss, R. Stevens, and L. Winkler. "Architecture of the Multi-Modal Organizational Research and Production Heterogeneous Network (MORPHnet)," Argonne National Laboratory, ECT and MCS Divisions, January 1997.
<http://www.anl.gov/ECT/Public/research/morphnt2.htm>
- [5] Chandra, R., P. Traina, and T. Li. RFC 1997, "BGP Communities Attribute." August 1996; E. Chen and T. Bates. RFC 1998, "An Application of the BGP Community Attribute in Multi-home Routing." August 1996.
- [6] Hanks, S., T. Li, D. Farinacci, and P. Traina. RFC 1701, "Generic Routing Encapsulation." October 1994; S. Hanks, T. Li, D. Farinacci, and P. Traina. RFC 1702, "Generic Routing Encapsulation over IPv4 networks." October 1994.
- [7] Valencia, A., K. Hamzeh, A. Rubens, T. Kolar, M. Littlewood, W. M. Townsley, J. Taarud, G. S. Pall, B. Palter, and W. Verthein. "Layer Two Tunneling Protocol 'L2TP.'" `draft-ietf-pppext-l2tp-10.txt`, March 1998.
- [8] Hamzeh, K., G. Singh Pall, W. Verthein, J. Taarud, and W. A. Little. "Point-to-Point Tunneling Protocol—PPTP." `draft-ietf-pppext-pptp-02.txt`, July 1997.
See also: <http://www.microsoft.com/backoffice/communications/morepptp.htm>
- [9] Waitzman, D., C. Partridge, and S. Deering. RFC 1075, "Distance Vector Multicast Routing Protocol." November 1988. For historical purposes, see also <ftp://ftp.isi.edu/mbone/faq.txt>
- [10] Luciani, J., D. Katz, D. Piscitello, B. Cole, and N. Doraswamy. "NBMA Next Hop Resolution Protocol (NHRP)," `draft-ietf-rolc-nhrp-15.txt`, February 1998.

PAUL FERGUSON is a consulting engineer at Cisco Systems and an active participant in the Internet Engineering Task Force (IETF). His principal areas of expertise include large-scale network architecture and design, global routing, Quality of Service (QoS) issues, and Internet Service Providers. Prior to his current position at Cisco Systems, he worked in network engineering, analytical, and consulting capacities for Sprint, Computer Sciences Corporation (CSC), and NASA. He is coauthor of *Quality of Service: Delivering QoS on the Internet and in Corporate Networks*, published by John Wiley & Sons, ISBN 0-471-24358-2, a collaboration with Geoff Huston. E-mail: ferguson@cisco.com

GEOFF HUSTON holds a B.Sc and a M.Sc from the Australian National University. He has been closely involved with the development of the Internet for the past decade, particularly within Australia, where he was responsible for the the initial build of the Internet within the Australian academic and research sector. Huston is currently the Chief Technologist in the Internet area for Telstra. He is also an active member of the IETF, and was an inaugural member of the Internet Society Board of Trustees. He is coauthor of *Quality of Service: Delivering QoS on the Internet and in Corporate Networks*, published by John Wiley & Sons, ISBN 0-471-24358-2, a collaboration with Paul Ferguson. E-mail: gih@telstra.net

SSL: Foundation for Web Security

by William Stallings

Virtually all businesses, most government agencies, and many individuals now have Web sites. The number of individuals and companies with Internet access is expanding rapidly, and all of them have graphical Web browsers. As a result, businesses are enthusiastic about setting up facilities on the Web for electronic commerce. But the reality is that the Internet and the Web are extremely vulnerable to compromises of various sorts. As businesses utilize the Internet for more than information dissemination, they will need to use trusted security mechanisms.

An increasingly popular general-purpose solution is to implement security as a protocol that sits between the underlying transport protocol (TCP) and the application. The foremost example of this approach is the *Secure Sockets Layer* (SSL) and the follow-on Internet standard of SSL known as *Transport Layer Security* (TLS). At this level, there are two implementation choices. For full generality, SSL (or TLS) could be provided as part of the underlying protocol suite and therefore be transparent to applications. Alternatively, SSL can be embedded in specific packages. For example, Netscape and Microsoft Explorer browsers come equipped with SSL, and most Web servers have implemented the protocol. Although it is possible to use SSL for applications other than Web transactions, its use at present is typically as part of Web browsers and servers and hence limited to Web traffic. Most of this article deals with the technical details of SSL; the status of TLS is described at the end.

If you have viewed an HTML source document, you have seen that the links are referenced with `HREF=<URL>` within an anchor (A) tag. In most cases, the reference is to another document through the use of the *Hyper Text Transfer Protocol*, or HTTP. For this, the browser initiates one or more sessions to the destination port of TCP/80 (the well-known port for HTTP) on the server. In some cases, a plug-in can be called, and data specific to that plug-in can be transferred to or from the browser. For that, the browser would initiate a session to the well-known TCP port of the plug-in. SSL is called when the reference starts like the following: `HREF="https://. .` By calling "https" within the browser, it is mandating that the data be transferred through the use of SSL. By clicking on this hot link, the browser initiates a session to the server on port TCP/443. SSL attempts to negotiate a secure link and transfers the data across it. If the negotiation fails, no data is transferred. The browser usually indicates that a secure connection has been requested. Netscape Navigator version 3 indicates this with a blue border around the page and a highlighted key in the lower left corner. Netscape Communicator version 4 displays this with a closed padlock in a lower status window. Microsoft's Internet Explorer indicates it

with a padlock in a lower information window. Display of these signs indicates that the information within the browser window has been delivered through the security of SSL.

SSL was originated by Netscape. Version 3 of the protocol was designed with public review and input from industry and was published as an Internet Draft document. Subsequently, when a consensus was reached to submit the protocol for Internet standardization, the TLS working group was formed within the *Internet Engineering Task Force* (IETF) to develop a common standard. The current work on TLS is aimed at producing an initial version as an Internet Standard. This first version of TLS can be viewed as essentially an SSLv3.1, and is very close to SSLv3. TLS includes a mechanism by which a TLS entity can back down to the SSLv3.0 protocol; in that sense, TLS is backward compatible with SSL.

SSL Architecture

SSL is designed to make use of TCP to provide a reliable end-to-end secure service. SSL is not a single protocol but rather two layers of protocols.

The SSL Record Protocol provides basic security services to various higher-layer protocols. In particular, the HTTP, which provides the transfer service for Web client/server interaction, can operate on top of SSL. Three higher-layer protocols are defined as part of SSL: the *Handshake Protocol*, the *Change CipherSpec Protocol*, and the *Alert Protocol*. These SSL-specific protocols are used in the management of SSL exchanges.

Two important SSL concepts are the SSL session and the SSL connection, which are defined in the specification as follows:

- **Connection:** A logical client/server link that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. The connections are transient. Every connection is associated with one session.
- **Session:** An association between a client and a server. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic security parameters, which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.

Between any pair of parties (applications such as HTTP on client and server), there may be multiple secure connections. In theory, there may also be multiple simultaneous sessions between parties, but this feature is not used in practice.

Several states are associated with each session. When a session is established, there is a current operating state for both read and write (that is, receive and send). In addition, during the Handshake Protocol, pending read and write states are created. Upon successful conclusion of the Handshake Protocol, the pending states become the current states. A session state is defined by the following parameters (definitions taken from the SSL specification):

- Session identifier: An arbitrary byte sequence chosen by the server to identify an active or resumable session state.
- Peer certificate: An X509.v3 certificate of the peer. This element of the state may be null.
- Compression method: The algorithm used to compress data prior to encryption.
- CipherSpec: Specifies the bulk data encryption algorithm (such as DES) and a hash algorithm (such as MD5 or SHA-1). It also defines cryptographic attributes such as the hash size.
- Master secret: 48-byte secret shared between the client and server.
- Is resumable: A flag indicating whether the session can be used to initiate new connections.

A connection state is defined by the following parameters:

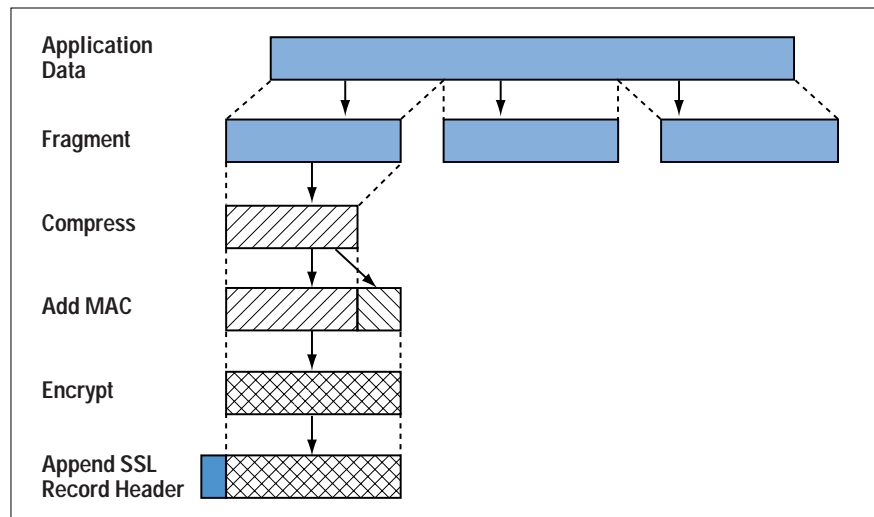
- Server and client random: Byte sequences that are chosen by the server and client for each connection.
- Server write MAC secret: The secret key used in MAC operations on data sent by the server.
- Client write MAC secret: The secret key used in MAC operations on data sent by the client.
- Server write key: The conventional encryption key for data encrypted by the server and decrypted by the client.
- Client write key: The conventional encryption key for data encrypted by the client and decrypted by the server.
- Initialization vectors: When a block cipher in CBC mode is used, an initialization vector (IV) is maintained for each key. This field is first initialized by the SSL Handshake Protocol. Thereafter the final ciphertext block from each record is preserved for use as the IV for the next record.
- Sequence numbers: Each party maintains separate sequence numbers for transmitted and received messages for each connection. When a party sends or receives a change CipherSpec message, the appropriate sequence number is set to zero.

SSL Record Protocol

The SSL Record Protocol provides two services for SSL connections: confidentiality, by encrypting application data; and message integrity, by using a *message authentication code* (MAC). The Record Protocol is a base protocol that can be utilized by some of the upper-layer protocols of SSL. One of these is the handshake protocol which, as described later, is used to exchange the encryption and authentication keys. It is vital that this key exchange be invisible to anyone who may be watching this session.

Figure 1 indicates the overall operation of the SSL Record Protocol. The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment. Received data is decrypted, verified, decompressed, and reassembled and then delivered to the calling application, such as the browser.

Figure 1:
SSL Record Protocol
Operation



The first step is fragmentation. Each upper-layer message is fragmented into blocks of 2^{14} bytes (16,384 bytes) or less. Next, compression is optionally applied. In SLLv3 (as well as the current version of TLS), no compression algorithm is specified, so the default compression algorithm is null. However, specific implementations may include a compression algorithm.

The next step in processing is to compute a message authentication code over the compressed data. For this purpose, a shared secret key is used. In essence, the hash code (for example, MD5) is calculated over a combination of the message, a secret key, and some padding. The receiver performs the same calculation and compares the incoming MAC value with the value it computes. If the two values match, the receiver is assured that the message has not been altered in transit. An attacker would not be able to alter both the message and the MAC, because the attacker does not know the secret key needed to generate the MAC.

Next, the compressed message plus the MAC are encrypted using symmetric encryption. A variety of encryption algorithms may be used, including the Data Encryption Standard (DES) and triple DES.

The final step of SSL Record Protocol processing is to prepend a header, consisting of the following fields:

- Content Type (8 bits): The higher-layer protocol used to process the enclosed fragment.
- Major Version (8 bits): Indicates major version of SSL in use. For SSLv3, the value is 3.
- Minor Version (8 bits): Indicates minor version in use. For SSLv3, the value is 0.
- Compressed Length (16 bits): The length in bytes of the plain-text fragment (or compressed fragment if compression is used).

The content types that have been defined are `change_cipher_spec`, `alert`, `handshake`, and `application_data`. The first three are the SSL-specific protocols, mentioned previously. The application-data type refers to the payload from any application that would normally use TCP but is now using SSL, which in turn uses TCP. In particular, the HTTP protocol that is used for Web transactions falls into the application-data category. A message from HTTP is passed down to SSL, which then wraps this message into an SSL record.

Change CipherSpec Protocol

The Change CipherSpec Protocol is one of the three SSL-specific protocols that use the SSL Record Protocol, and it is the simplest. This protocol consists of a single message, which consists of a single byte with the value 1. The sole purpose of this message is to cause the pending state to be copied into the current state, which updates the CipherSuite to be used on this connection. This signal is used as a coordination signal. The client must send it to the server and the server must send it to the client. After each side has received it, all of the following messages are sent using the agreed-upon ciphers and keys.

Alert Protocol

The Alert Protocol is used to convey SSL-related alerts to the peer entity. As with other applications that use SSL, alert messages are compressed and encrypted, as specified by the current state.

Each message in this protocol consists of two bytes. The first byte takes the value “warning” (1) or “fatal” (2) to convey the severity of the message. If the level is fatal, SSL immediately terminates the connection. Other connections on the same session may continue, but no new connections on this session may be established. The second byte contains a code that indicates the specific alert. An

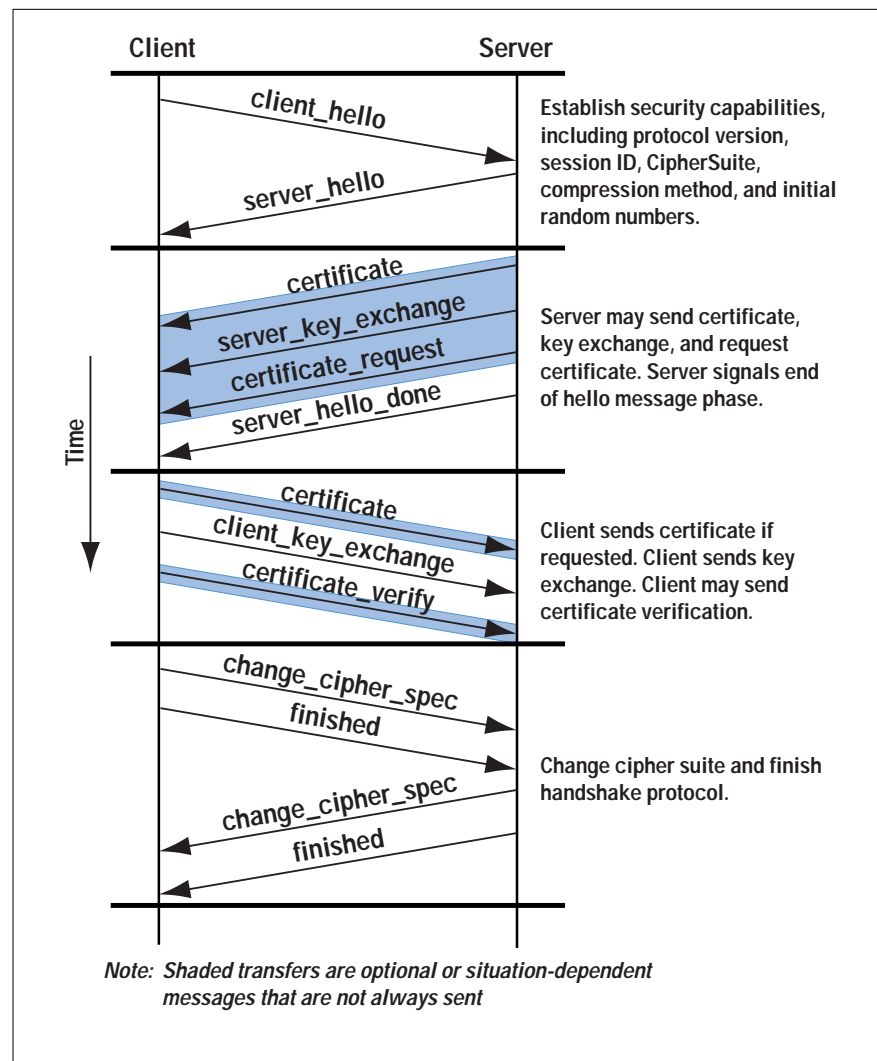
example of a fatal message is `illegal_parameter` (a field in a handshake message was out of range or inconsistent with other fields). An example of a warning message is `close_notify` (notifies the recipient that the sender will not send any more messages on this connection; each party is required to send a `close_notify` alert before closing the write side of a connection).

Handshake Protocol

The most complex part of SSL is the Handshake Protocol. This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record. The Handshake Protocol is used before any application data is transmitted. The Handshake Protocol consists of a series of messages exchanged by the client and the server.

Figure 2 shows the initial exchange needed to establish a logical connection between the client and the server. The exchange can be viewed as having four phases.

Figure 2:
Handshake Protocol
Action



Phase 1 is used to initiate a logical connection and to establish the security capabilities that will be associated with it. The exchange is initiated by the client, which sends a `client_hello` message with the following parameters:

- **Version:** The highest SSL version understood by the client.
- **Random:** A client-generated random structure, consisting of a 32-bit timestamp and 28 bytes generated by a secure random number generator. These values serve as nonces and are used during key exchange to prevent replay attacks.
- **Session ID:** A variable-length session identifier. A nonzero value indicates that the client wishes to update the parameters of an existing connection or create a new connection on this session. A zero value indicates that the client wishes to establish a new connection on a new session.
- **CipherSuite:** A list that contains the combinations of cryptographic algorithms supported by the client, in decreasing order of preference. Each element of the list (each CipherSuite) defines both a key exchange algorithm and a CipherSpec; these are discussed subsequently.
- **Compression Method:** A list of the compression methods the client supports.

After sending the `client_hello` message, the client waits for the `server_hello` message, which contains the same parameters as the `client_hello` message. For the `server_hello` message, the following conventions apply. The Version field contains the lower of the version suggested by the client and the highest version supported by the server. The Random field is generated by the server and is independent of the client's Random field. If the SessionID field of the client was nonzero, the same value is used by the server; otherwise the server's SessionID field contains the value for a new session. The CipherSuite field contains the single CipherSuite selected by the server from those proposed by the client. The Compression field contains the compression method selected by the server from those proposed by the client.

The first element of the CipherSuite parameter is the key exchange method (that is, the means by which the cryptographic keys for conventional encryption and MAC are exchanged). The following key exchange methods are supported:

- **RSA:** The secret key is encrypted with the receiver's RSA public key. A public-key certificate for the receiver's key must be made available.
- **Fixed Diffie-Hellman:** This a Diffie-Hellman key exchange in which the server's certificate contains the Diffie-Hellman public parameters

signed by the *certificate authority* (CA). That is, the public-key certificate contains the Diffie-Hellman public-key parameters. The client provides its Diffie-Hellman public key parameters either in a certificate, if client authentication is required, or in a key exchange message. This method results in a fixed secret key between two peers, based on the Diffie-Hellman calculation using the fixed public keys.

- **Ephemeral Diffie-Hellman:** This technique is used to create ephemeral (temporary, one-time) secret keys. In this case, the Diffie-Hellman public keys are exchanged, and signed using the sender's private RSA or DSS key. The receiver can use the corresponding public key to verify the signature. Certificates are used to authenticate the public keys. This option appears to be the most secure of the three Diffie-Hellman options because it results in a temporary, authenticated key.
- **Anonymous Diffie-Hellman:** The base Diffie-Hellman algorithm is used, with no authentication. That is, each side sends its public Diffie-Hellman parameters to the other, with no authentication. This approach is vulnerable to man-in-the-middle attacks, in which the attacker conducts anonymous Diffie-Hellman exchanges with both parties.

Following the definition of a key exchange method is the Cipher-Spec, which indicates the encryption and hash algorithms and other related parameters.

The server begins Phase 2 by sending its certificate, if it needs to be authenticated; the message contains one or a chain of X.509 certificates. The certificate message is required for any agreed-on key exchange method except anonymous Diffie-Hellman. Note that if fixed Diffie-Hellman is used, this certificate message functions as the server's key exchange message because it contains the server's public Diffie-Hellman parameters.

Next, a `server_key_exchange` message may be sent, if it is required. It is not required in two instances: (1) The server has sent a certificate with fixed Diffie-Hellman parameters; or (2) RSA key exchange is to be used.

Next, a nonanonymous server (server not using anonymous Diffie-Hellman) can request a certificate from the client. The `certificate_request` message includes two parameters: `certificate_type` and `certificate_authorities`. The certificate type indicates the type of public-key algorithm. The second parameter in the `certificate_request` message is a list of the distinguished names of acceptable certificate authorities.

The final message in Phase 2, and one that is always required, is the `server_done` message, which is sent by the server to indicate the end of the server hello and associated messages. After sending this message, the server waits for a client response. This message has no parameters.

Upon receipt of the `server_done` message, the client should verify that the server provided a valid certificate, if required, and check that the server hello parameters are acceptable. If all is satisfactory, the client sends one or more messages back to the server in Phase 3. If the server has requested a certificate, the client begins this phase by sending a certificate message. If no suitable certificate is available, the client sends a `no_certificate` alert instead.

Next is the `client_key_exchange` message, which must be sent in this phase. The content of the message depends on the type of key exchange.

Finally, in this phase, the client may send a `certificate_verify` message to provide explicit verification of a client certificate. This message is only sent following any client certificate that has signing capability (that is, all certificates except those containing fixed Diffie-Hellman parameters).

Phase 4 completes the setting up of a secure connection. The client sends a `change_cipher_spec` message and copies the pending CipherSpec into the current CipherSpec. Note that this message is not considered part of the Handshake Protocol but is sent using the Change CipherSpec Protocol. The client then immediately sends the finished message under the new algorithms, keys, and secrets. The finished message verifies that the key exchange and authentication processes were successful.

In response to these two messages, the server sends its own `change_cipher_spec` message, transfers the pending to the current CipherSpec, and sends its finished message. At this point the handshake is complete and the client and server may begin to exchange application layer data.

After the records have been transferred, the TCP session is closed. However, since there is no direct link between TCP and SSL, the state of SSL may be maintained. For further communications between the client and the server, many of the negotiated parameters are retained. This may occur if, in the case of Web traffic, the user clicks on another link that also specifies HTTPs on the same server. If the clients or servers wish to resume the transfer of records, they don't have to again negotiate encryption algorithms or totally new keys. The SSL specifications suggest that the state information be cached for no longer than 24 hours. If no sessions are resumed within that time, all information is deleted and any new sessions have to go through the handshake again. The specifications also recommend that neither the client nor the server have to retain this information, and shouldn't if either of them suspects that the encryption keys have been compromised. If either the client or the server does not agree to resume the session, for any reason, then both will have to go through the full handshake.

Transport Layer Security

TLS is an IETF standardization initiative whose goal is to produce an Internet standard version of SSL. In fact, the charter for the TLS working group states:

“The TLS working group is a focused effort on providing security features at the transport layer, rather than general purpose security and key management mechanisms. The standard track protocol specification will provide methods for implementing privacy, authentication, and integrity above the transport layer.”

This means that TLS can be used to provide security services to any application that uses TCP or the *User Datagram Protocol* (UDP). However, the driving force behind this work is to develop a standardized version of SSL. Microsoft has indicated that TLS will go into the next major version of its browser and Web server products, and Netscape has made a similar commitment. With this kind of support, it is likely that TLS will move quickly along the Internet Standards track.

The current draft version of TLS is very similar to SSLv3. TLS uses slightly different cryptographic algorithms for such things as the MAC function generation of secret keys. TLS also includes more alert codes.

SSL is already widely deployed and, under the name TLS, is moving toward Internet standardization. It is the solution of choice for Web transaction security.

References

- [1] <http://www.phaos.com/sslresource.html>
(has links to vendors, SSL specifications, and FAQs)
- [2] <http://www.netscape.com/newsref/std/SSL.html>
(PostScript versions of the spec are available there)
- [3] <http://www.ietf.org/html.charters/tls-charter.html>
(contains latest RFCs and Internet Drafts for TLS)
- [4] <http://www.imc.org/ietf-tls/mail-archive/>
(mailing list archive)
- [5] <ftp://ftp.ietf.org/internet-drafts/draft-ietf-tls-protocol-05.txt>
- [6] <ftp://ftp.ietf.org/internet-drafts/draft-ietf-tls-https-01.txt>
- [7] <http://www.consensus.com/security/ssl-talk-faq.html>

WILLIAM STALLINGS is a consultant, lecturer, and author of over a dozen books on data communications and computer networking. He has a PhD in computer science from M.I.T. This article is based on material in the author's latest book, *Cryptography and Network Security, Second Edition* (Prentice-Hall, 1998). His home in cyberspace is <http://www.shore.net/~ws> and he can be reached at ws@shore.net

Call for Papers

The Internet Protocol Journal (IPJ) is published quarterly by Cisco Systems. The journal is not intended to promote any specific products or services, but rather is intended to serve as an informational and educational resource for engineering professionals involved in the design, development, and operation of public and private internets and intranets. The journal will carry tutorial articles (“What is...?”), as well as implementation/operation articles (“How to...”). It will provide readers with technology and standardization updates for all levels of the protocol stack and serve as a forum for discussion of all aspects of internetworking.

Topics include, but are not limited to:

- Access and infrastructure technologies such as: ISDN, Gigabit Ethernet, SONET, ATM, xDSL, cable fiber optics, satellite, wireless, and dial systems
- Transport and interconnection functions such as: switching, routing, tunneling, protocol transition, multicast, and performance
- Network management, administration, and security issues, including: authentication, privacy, encryption, monitoring, firewalls, troubleshooting, and mapping
- Value-added systems and services such as: Virtual Private Networks, resource location, caching, client/server systems, distributed systems, network computing, and quality of service
- Application and end-user issues such as: e-mail, Web authoring, server technologies and systems, electronic commerce, and application management
- Legal, policy, and regulatory topics such as: copyright, content control, content liability, settlement charges, “modem tax,” and trademark disputes in the context of internetworking

In addition to feature-length articles, IPJ will contain standardization updates, overviews of leading and bleeding-edge technologies, book reviews, announcements, opinion columns, and letters to the Editor.

Cisco will pay a stipend of US\$1000 for published, feature-length articles. Author guidelines are available from Ole Jacobsen, the Editor and Publisher of IPJ, reachable via e-mail at ole@cisco.com

Book Reviews

Groupware *Groupware: Collaborative Strategies for Corporate LANs and Intranets*, by David Coleman, ISBN 0-13-727728-8, Prentice-Hall PTR, 1997, <http://www.prenhall.com>.

Some areas of science provide very poor training for dealing with primarily human processes. One might think that packet switching would be an exception because it lives on the stochastic nature of bursty communications. Because our knowledge of human and group activity is, at best, characterized by statistical assessments, those working in networking should do well in understanding and dealing with the unpredictable and human nature of communication, especially when it involves using networks.

So much for theory. In general, the world of lower-level networking has done little for the upper strata of computer-mediated human communication, except to provide a platform for the work of others. An apparent exception in the world of Internet technology is e-mail, yet it actually serves more as proof of the problem than as an exception. The basic facilities in Internet e-mail are the same today as they were 25 years ago. As nice as they are, the word “basic” is essential when characterizing them. Almost none of the Internet’s standardized e-mail facilities are really targeted at providing automated or structural support for the work of a group.

Groupware Defined

The collection of products and services designed to help people collaborate via computer, by direct interaction, or by information dissemination is called “groupware.” Coleman’s book is a revision of *Groupware: Technology and Applications*. Written only 15 months earlier, the world changed more than enough in that time to require the revision. The first book had relatively little to say about the Internet, whereas this new book tries mightily to factor it into the equation. The result is a bit erratic, but the digressions serve to highlight how rapidly things are changing, rather than to suggest looking elsewhere for a better source on the topic.

The new book has an entirely different subtitle, giving a reasonable sense that the content targets more an understanding of system organization and function than detailed technical explanation. That’s just fine, because the book really is not particularly technical. It covers the requirements and functions for supporting activity by groups.

Downsizing and working remotely are two very strong driving forces for increased use of groupware. This book is essentially an introduction to concepts, functionality, and use of systems that attempt to help staff members work together. Oddly, that does not only mean working together when physically separated, because there is discussion of meeting room assistance, such as with automated sense-of-the-group tallying devices.

Organization

The first two chapters introduce the topic, emphasizing that human and group process concerns dominate the field and are intimately tied to the aggressive efforts that organizations are making to run more productively and, frequently, with fewer people. The third chapter discusses functionality in terms of the World Wide Web. The book reflects the current enthusiasm for the Web, sometimes to the detriment of the appropriate use of messaging technology, although messaging is more prevalent among groupware than other kinds of commercial Internet systems.

The realm of groupware does not have a firm taxonomy. My own synthesis includes: Message (text and document) Exchange, Forms Exchange, Calendaring & Scheduling, Workflow, Presentations and Interactive Meetings, and Document Development and Sharing. The next six chapters cover the functional pieces of this groupware realm.

The next five chapters cover the major vendors of integrated groupware products: Lotus Notes, Novel GroupWise, TeamWARE, Hewlett-Packard, and Oracle Interoffice. HP's chapter discusses "strategy," suggesting the lack of a well-integrated product suite, but one more survey of the terrain is nonetheless useful. And that, perhaps, is the major reason for reading this book: It constantly emphasizes the human and process-oriented aspect of organizational behavior and the need to attend carefully both to the needs of the humans and the nature of the processes. It is easy to understand that an improper travel authorization, will bring an organization to its knees. It is easy to forget that the system is used by humans who well might not want the added complexity or rigidity of the system and who, therefore, must be part of the design and adoption effort. In my opinion, the book takes a rather more negative view about groupware acceptability than is necessary, but then I like such technology, and the average worker in the average organization does not.

The last six chapters of this book intermix case studies and Hahn, of Collabra and Netscape, points the reader to Chapter 17, "Groupware & Reengineering: The Human Side of Change." Although one of the better considerations of these issues in the book, it is far from the only one.

A Useful Survey

If you have little familiarity with these "upper level application" areas of networking, the functionality, products, or use, then this book is a good one to read. You will not learn much about the underlying technology, nor will you be able to qualify as a "certified groupware support engineer," but you will obtain an extremely useful survey of the field, and you will obtain it from the perspective of human and organization use. As the Internet moves into the mass market, that perspective is a good one.

—Dave Crocker
Brandenburg Consulting
dcrocker@brandenburg.com

High-Speed Networks *High-Speed Networks: TCP/IP and ATM Design Principles*,
by William Stallings, ISBN 0-13-525965-7 Prentice-Hall, 1997,
<http://www.shore.net/~ws/HsNet.html>

High-speed networks now dominate both the WAN and LAN markets. In the WAN market, data networks have evolved from packet-switching networks to ATM networks operating at 155 Mbps or more. In the LAN market, the staple 10-Mbps Ethernet is being replaced with 100-Mbps Fast Ethernet, Gigabit Ethernet, and even Asynchronous Transfer Mode (ATM) LANs. This book provides a survey of high-speed networks and the design issues related to them. Much of the book is devoted to the study of various techniques aimed at reducing network congestion.

Organization

The book is divided into seven sections. The first section deals with the fundamentals: TCP/IP principles; packet switching and Frame Relay networks; and internetworking principles. The second section provides an overview of ATM and Fast and Gigabit Ethernet. These two sections can easily be torn out of the book and serve as an excellent primer on today's modern networks. I am going to recommend to my employer that they be made mandatory reading.

In the third section of the book, Stallings focuses on one treatment of queueing theory, namely, how it is applied to modeling network behavior. Stallings has an undeniable gift for taking large complicated subjects and teaching the fundamentals, and then some, without belittling the subject at hand or the reader. This book is witness to this gift, and this chapter but one fine example. But once the reader has an understanding of queueing theory, Stallings throws a wrench in the gears. The chapter on self-similarity explains why traditional queueing models are inadequate when trying to predict the performance of Ethernet traffic and other self-similar streams. While this section is by far the most theoretical, it is at the same time necessary for the reader's understanding of network performance, and while many readers may not care to devote the time necessary to gain a complete understanding of self-similarity, astute students are urged to invest in more than a simple gloss-over of this section.

Having understood the basics of self-similarity, I hoped the fifth section of the book, on network traffic management, would be addressed with greater emphasis on delivering quality of service and the problems related to self-similarity. Instead, the material is based on traditional queueing models.

The fourth section, flow control, is divided into two categories. The first, link control mechanisms, focuses on some of the performance issues related to the use of *Automatic Repeat Request* (ARQ) link control protocols. The second category, transport control mechanisms,

concentrates on the TCP flow control mechanism. I expected to find references to bugs in some TCP implementations exposed by high-volume WWW servers, but didn't. Stallings goes on to present an overview of some of the performance issues of TCP over ATM. As institutions begin upgrading their networks, this issue is sure to receive a great deal of interest. The section concludes with a look at the *Real-Time Transport Protocol*, another area sure to spark attention as the need to move large multimedia data across WANs, in real time, becomes more relevant.

The sixth section of the book covers Internet routing protocols and opens with a primer on graph theory. Four routing protocols (RIP, OSPF, BGP, and IDRP) are covered. The section concludes with a discussion of multicasting as an introduction to RSVP. This section sparked my curiosity enough to call for a visit to the WWW site for RSVP development.

Stallings shies away from directly addressing application-driven improvements aimed at increasing network performance. In today's Web/CGI-driven world, I would expect this to be a topic of interest to many. Perhaps this is a subject for another book. But the topic is not entirely avoided. The last section of the book focuses on various lossless and lossy compression techniques. The quirkiness of material covered makes this section a darling.

Recommended

This book rates an A+. Unlike most books about computers being published today, this book is neither superficial nor is it insulting to the reader. It is intended for both professional and academic audiences. Stallings' desire to truly educate is apparent. This is not a book about promoting the hype, this is a book about serious learning.

—Neophytos Iacovou,
University of Minnesota
Academic & Distributed Computing Services
iacovou@boombox.micro.umn.edu

Fragments

The Fragments page is intended to provide you with updates and pointers to information related to Internet technology developments.

The Future of the Domain Name System (DNS)

For more than a year, a debate has taken place regarding the future of the DNS. In particular, the issue of competitive name registries, possible addition of new *global Top Level Domains* (gTLDs) and the future of the *Internet Assigned Numbers Authority* (IANA) have been discussed. Information regarding the initial proposal can be found at: <http://www.gtld-mou.org/>. The US Government has issued a so-called *Green Paper* entitled “Technical Management of Internet Names and Addresses.” The Green Paper and comments received on this document can be found at:

<http://www.ntia.doc.gov/ntiahome/domainname/>

IETF and Related Links

The *Internet Engineering Task Force* (IETF) is responsible for the development of standards for Internet technology. Membership to the IETF is open and you can participate in person or subscribe to the IETF mailing list. The IETF meets three times per year. For a list of future meetings and other IETF information see: <http://www.ietf.org>. On this website you will also find a number of links to organizations which are related to the IETF in one way or another:

- *The Internet Society* (ISOC) and its annual INET conference.
- *The Internet Architecture Board* (IAB)
- *The Internet Assigned Numbers Authority* (IANA)
- *The Internet Research Task Force* (IRTF)

SIGCOMM

If you want to learn about the latest developments on the research side of networking you should check out SIGCOMM, the Association for Computing Machinery’s Special Interest Group on Communications. You can find out more about the group and their annual conference at: <http://www.acm.org/sigcomm/sigcomm98>

Send Us Your Comments!

We look forward to hearing your comments and suggestions regarding anything you read in this publication. Send e-mail to: ipj@cisco.com.

This publication is distributed on an “as-is” basis, without warranty of any kind either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, or noninfringement. This publication could contain technical inaccuracies or typographical errors. Later issues may modify or update information provided in this issue. Neither the publisher nor any contributor shall have any liability to any person for any loss or damage caused directly or indirectly by the information contained herein.

The Internet Protocol Journal

Ole J. Jacobsen, Editor and Publisher

Editorial Advisory Board

Dr. Vint Cerf, Sr. VP, Internet Architecture and Engineering
MCI Communications, USA

David Farber
The Alfred Fitler Moore Professor of Telecommunication Systems
University of Pennsylvania, USA

Edward R. Kozel, Sr. VP, Corporate Development
Cisco Systems, Inc., USA

Peter Löthberg, Network Architect
Stupi AB, Sweden

Dr. Jun Murai, Professor, WIDE Project
Keio University, Japan

Dr. Deepinder Sidhu, Professor, Computer Science &
Electrical Engineering, University of Maryland, Baltimore County
Director, Maryland Center for Telecommunications Research, USA

Pindar Wong, Chairman and President,
VeriFi Limited, Hong Kong

*The Internet Protocol Journal is
published quarterly by the Cisco News
Publications Group, Cisco Systems, Inc.
www.cisco.com*

*Tel: +1 408 526-4000
E-mail: ipj@cisco.com*

*Cisco, Cisco Systems, and the Cisco
Systems logo are registered
trademarks of Cisco Systems, Inc. in
the USA and certain other countries.
All other trademarks mentioned in this
document are the property of their
respective owners.*

*Copyright © 1998 Cisco Systems Inc.
All rights reserved. Printed in the USA.*



The Internet Protocol Journal, Cisco Systems
170 West Tasman Drive, M/S SJ-J4
San Jose, CA 95134-1706
USA

ADDRESS SERVICE REQUESTED

Bulk Rate Mail U.S. Postage PAID Cisco Systems, Inc.
--