The Internet Protocol Journal

September 2000 Volume 3, Number 3

A Quarterly Technical Publication for Internet and Intranet Professionals

In This Issue

You can download IPJ back issues and find subscription information at:
www.cisco.com/ipj

From The Editor

In our last issue, Geoff Huston described the basic design and operation of the *Transmission Control Protocol* (TCP). He outlined how numerous enhancements to TCP implementations have been developed over time to improve its performance, particularly in the face of congested networks. The Internet is a rapidly changing environment in which both the applications and the underlying transmission systems are undergoing an evolution, if not a revolution. Some of these changes, such as the introduction of wireless devices, affect the way TCP works, because the protocol makes many implicit assumptions about the network over which it operates. In this issue, Geoff looks at the future for TCP and describes techniques for adopting TCP to today's Internet.

Security continues to be a major concern for everyone involved in the design and operation of networks. Widely publicized "hacker attacks," "denial-of-service attacks," and outright online fraud has brought the topic into sharp focus in the last few years. Because security was not part of the original design of the Internet, numerous solutions at every level of the protocol stack have been proposed and implemented over the last three decades. Today's network manager is, therefore, faced with a *system* of security components that must be carefully configured and monitored in order to provide sufficient security without preventing users from getting their work done. In our second article, Chris Lonvick explores a model for evaluating and securing a network.

The online subscription system for this journal is now up and running at **www.cisco.com/ipj** In addition to offering a subscription form, the system allows you to select delivery options, update your mailing and e-mail address, and much more. Please visit our Web site and give it a try. If you encounter any difficulties, please send your comments to **ipj@cisco.com.**

—Ole J. Jacobsen, Editor and Publisher
ole@cisco.com

The Future for TCP

by Geoff Huston, Telstra

The previous article, "TCP Performance," examined the operation of the *Transmission Control Protocol* (TCP) protocol[1]. The article examined the role of TCP in providing a reliable end-to-end data transfer function, and described how TCP incorporates numerous control functions that are intended to make efficient use of the underlying IP network through a host-based congestion control function. Congestion control is an important component of TCP implementations, and today TCP congestion control plays an important role in the overall stability of the Internet.

Today's Internet spans a very broad base of uses, and ensuring that TCP provides a highly robust, efficient, and reliable service platform for such a diversity of use is a continuing task. The Web has introduced a component of short duration reliable transfers into the public Internet traffic profile. These short sessions are often referred to as "TCP mice" because of the short duration and large number of such TCP sessions. Complementing these short sessions is the increasing size of large transfers as *File Transfer Protocol* (FTP) data sets become larger in response to increasing capacity within the public Internet network[4]. In addition, there is an increasing diversity of media used within the Internet, both in terms of higher-speed systems and in the use of wireless systems for Internet access. In this article we will extend our examination of TCP by looking at how TCP is being used and adapted to match this changing environment.

A Review of TCP Performance

Within any packet-switched network, when demand exceeds available capacity, the packet switch will use a queue to hold the excess packets. When this queue fills, the packet switch must drop packets. Any reliable data protocol that operates across such a network must recognize this possibility and take corrective action. TCP is no exception to this constraint. TCP uses data sequence numbering to identify packets, and explicit acknowledgements (ACKs) to allow the sender and receiver to be aware of reliable packet transfer. This form of reliable protocol design is termed "end-to-end" control, because interior switches do not attempt to correct packet drops. Instead, this function is performed through the TCP protocol exchange between sender and receiver. TCP uses cumulative ACKs rather than per-packet ACKs, where an ACK referencing a particular point within the data stream implicitly acknowledges all data with a sequence value less than the ACKed sequence.

TCP also uses ACKs to clock the data flow. ACKs arriving back at the sender arrive at intervals approximately equal to the intervals at which the data packets arrived at the sender. If TCP uses these ACKs to trigger sending further data packets into the network, then the packets will be entered into the network at the same rate as they are arriving at their destination. This mode of operation is termed "ACK clocking."

TCP recovers from packet loss using two mechanisms. The most basic operation is the use of packet timeouts by the sender. If an ACK for a packet fails to arrive within the timeout value, the sender will retransmit the oldest unacknowledged packet. In such a case, TCP assumes that the loss was caused by a network congestion condition, and the sender will enter "Slow Start" mode. This condition causes significant delays within the data transfer, because the sender will be idle during the timeout interval and upon restarting will recommence with a single packet exchange, gradually recovering the data rate that was active prior to the packet loss. Many networks exhibit transient congestion conditions, where a data stream may experience loss of a single packet within a packet train. To address this, TCP introduced the mechanism of "fast recovery." This mechanism is triggered by a sequence of three duplicate ACKS received by the data sender. These duplicate ACKs are generated by the packets that trail the lost packet, where the sender ACKs each of these packets with the ACK sequence value of the lost packet. In this mode the sender immediately retransmits the lost packet and then halves its sending rate, continuing to send additional data as permitted by the current TCP sending window. In this mode of operation, "congestion-avoidance" TCP increases its sending window at a linear rate of one segment per *Round-Trip Time* (RTT). This mode of operation is referred to as *Additive Increase, Multiplicative Decrease* (AIMD), where the protocol reacts sharply to signs of network congestion, and gradually increases its sending rate in order to equilibrate with concurrent TCP sessions.

TCP Design Assumptions

It is difficult to design any transport protocol without making some number of assumptions about the environment in which the protocol is to be used, and TCP certainly has some inherent assumptions hidden within its design. The most important set of assumptions that lie behind the design of TCP are as follows:

- *A network of wires, not wireless:* As we continually learn, wireless is different. Wireless systems typically have higher *bit error rates* (BERs) than wire-based carriage systems. Mobile wireless systems also include factors of signal fade, base-station handover, and variable levels of load. TCP was designed with wire-based carriage in mind, and the design of the protocol makes numerous assumptions that are typical of such of an environment. TCP makes the assumption that packet loss is the result of network congestion, rather than bit-level corruption. TCP also assumes some level of stability in the RTT, because TCP uses a method of damping down the changes in the RTT estimate.
- *A best-path route-selection protocol:* TCP assumes that there is a single best metric path to any destination because TCP assumes that packet reordering occurs on a relatively minor scale, if at all. This implies that all packets in a connection must follow the same path within the network or, if there is any form of load balancing, the order of packets within each flow is preserved by some network-level mechanism.

- *A network with fixed bandwidth circuits, not varying bandwidth:* TCP assumes that available bandwidth is constant, and will not vary over short time intervals. TCP uses an end-to-end control loop to control the sending rate, and it takes many RTT intervals to adjust to varying network conditions. Rapidly changing bandwidth forces TCP to make very conservative assumptions about available network capacity.
- *A switched network with first-in, first-out (FIFO) buffers:* TCP also makes some assumptions about the architecture of the switching elements within the network. In particular, TCP assumes that the switching elements use simple FIFO queues to resolve contention within the switches. TCP makes some assumption about the size of the buffer as well as its queuing behavior, and TCP works most efficiently when the buffer associated with a network interface is of the same order of size as the delay bandwidth product of the associated link.
- *The duration of TCP sessions:* TCP also makes some assumptions about the nature of the application. In particular, it assumes that the TCP session will last for some number of round-trip times, so that the overhead of the initial protocol handshake is not detrimental to the efficiency of the application. TCP also takes numerous RTT intervals to establish the characteristics of the connection in terms of the true RTT interval of the connection as well as the available capacity. The introduction of short-duration sessions, such as found in transaction applications and short Web transfers, is a new factor that impacts the efficiency of TCP.
- *Large payloads and adequate bandwidth:* TCP assumes that the overhead of a minimum of 40 bytes of protocol per TCP packet (20 bytes of IP header and 20 bytes of TCP header) is an acceptable overhead when compared to the available bandwidth and the average payload size. When applied to low-bandwidth links, this is no longer the case, and the protocol overheads may make the resultant communications system too inefficient to be useful.
- *Interaction with other TCP sessions:* TCP assumes that other TCP sessions will also be active within the network, and that each TCP session should operate cooperatively to share available bandwidth in order to maximize network efficiency. TCP may not interact well with other forms of flow-control protocols, and this could result in unpredictable outcomes in terms of sharing of the network resource between the active flows as well as poor overall network efficiency.

If these assumptions are challenged, the associated cost is that of TCP efficiency. If the objective is to extend TCP to environments where these assumptions are no longer valid, while preserving the integrity of the TCP transfer and maintaining a high level of efficiency, then the TCP operation itself may have to be altered.

There are two basic ways of altering TCP operation: by altering the actions of the end host by making changes to the TCP protocol, or by altering the characteristics of the network, making them more "friendly" to TCP. We will look at the potential for both responses in examining various scenarios for adapting TCP to suit these changing environments.

Some caution should be noted about making changes to the TCP protocol. The major constraint is that any changes that are contemplated to TCP should be backward compatible with existing TCP behavior. This constraint requires a modified TCP protocol to attempt to negotiate the use of a specific protocol extension, and the knowledge that a basic common mode of protocol operation may be required if the negotiation fails. The second constraint is that TCP does assume that it is interacting with other TCP sessions within the network, and the outcome of fair sharing of the network between concurrent sessions depends on some commonality of the protocol used by these sessions. Major changes to the protocol behavior can lead to unpredictable outcomes in terms of sharing of the network resource between "unmodified" and "modified" TCP sessions, and unpredictable outcomes in terms of efficiency of the use of the network. For this reason there is some understandable reluctance to undertake modifications of TCP that radically alter TCP startup behavior or behavior in the face of network congestion.

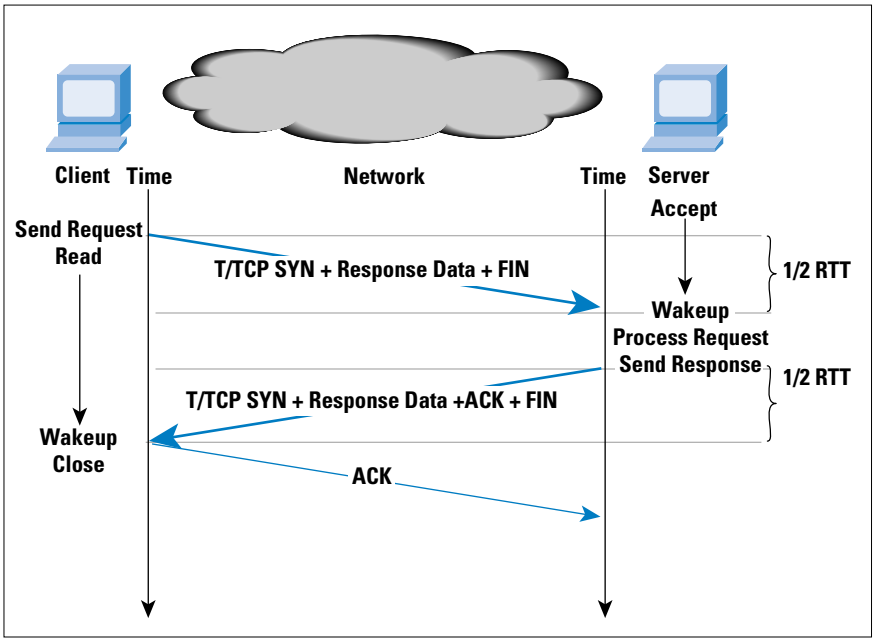
Short-Duration Sessions—TCP for Transactions

For network applications that generate small transactions, the application designer is faced with a dilemma. The application may be able to use the *User Datagram Protocol* (UDP), in which case the sender must send the query and await the response. This operation is highly efficient, because the total elapsed time for the client is a single RTT. However, this speed is gained at the cost of reliability. A missing response is ambiguous, in that it is impossible for the initiator to tell whether the query was lost or the response was lost. If multiple queries are generated, it is not necessarily true that they will arrive at the remote server in the same order as they were generated. Alternatively, the application can use TCP, which will ensure reliability of the transaction. However, TCP uses a three-way handshake to complete the opening of the connection, and uses acknowledged FIN signals for each side to close its end of the connection after it has completed sending data. Under the control of TCP, the sender will retransmit the query until it receives an acknowledgment that the query has arrived at the remote server. Similarly, the remote host will retransmit the response until the server receives an indication that the response has been successfully delivered. The cost of this reliability is application efficiency, because the minimum time to conduct the TCP transaction for the client is two RTT intervals.

TCP for Transactions (commonly referred to as T/TCP[5]) attempts to improve the performance of small transactions while preserving the reliability of TCP. T/TCP places the query data and the closing FIN in the initial SYN packet. This can interpreted as attempting to open a session, pass data, and close the sender's side of the session within a single packet. If the server accepts this format, the server responds with a single packet, which contains its SYN response, an ACK of the query data, the server's data in response, and the closing FIN. All that is required to complete the transaction is for the query system to ACK the server's data and FIN (Figure 1). If the server does not accept this format, the client can back off to a conventional TCP handshake followed by a data exchange.

For the client, the time to undertake this T/TCP transaction is one RTT interval, a period equal to the UDP-supported transaction, while still allowing for the two systems to use TCP to negotiate a reliable exchange of data as a backup.

Figure 1: T/TCP Operation



T/TCP requires changes to the protocol stack of both the sender and the receiver in order to operate correctly. The design of the protocol explicitly allows the session initiator to back off to use TCP if the receiver cannot correctly respond to the initial T/TCP packet.

T/TCP is not in common use in the Internet today, because while it improves the efficiency of simple transactions, the limited handshake makes it more vulnerable from a security perspective, and concerns over this vulnerability have been a prohibitive factor in its adoption. This is illustrative of the nature of the trade-offs that occur within protocol design, where optimizing one characteristic of a protocol may be at the expense of other aspects of the protocol.

Long Delay—TCP for Satellite Paths

Satellite-based services pose a set of unique issues to the network designer. Most notably, these issues include delay, bit errors, and bandwidth.

When using a satellite path, there is an inherent delay in the delivery of a packet due to signal propagation times related to the altitude of communications satellites. Geo-stationary orbit spacecraft are located at an altitude of some 36,000 km, and the propagation time for a signal to pass from an earth station directly below the satellite to the satellite and back is 239.6 ms. If the earth station is located at the edge of the satellite view area, this propagation time extends to 279.0 ms. In terms of a round trip that uses the satellite path in both directions, the RTT of a satellite hop is between 480 and 560 ms.

The strength of a radio signal falls in proportion to the square of the distance traveled. For a satellite link, the signal propagation distance is large, so the signal becomes weak before reaching its destination, resulting in a poor signal-to-noise ratio. Typical BERs for a satellite link today are on the order of 1 error per 10 million bits (1×10^{-7}). *Forward error correction* (FEC) coding can be added to satellite services to reduce this error rate, at the cost of some reduction in available bandwidth and an increase in latency due to the coding delay.

There is also a limited amount of bandwidth available to satellite systems. Typical carrier frequencies for commercial satellite services are 6/4 GHz (C-band) and 14/12 GHz (Ku band). Satellite transponder bandwidth is typically 36 MHz[6].

When used in a data carriage role for IP traffic, satellite channels pose several challenges for TCP.

The delay-bandwidth product of a transmission path defines the amount of data TCP should have within the transmission path at any one time, in order to fully utilize the available channel capacity. The delay used in this equation is the RTT and the bandwidth is the capacity of the bottleneck link in the network path. Because the delay in satellite environments is large, a TCP flow may need to keep a large amount of data within the transmission path. For example, a typical path that includes a satellite hop may have a RTT of some 700 ms. If the bottleneck bandwidth is 2 Mbps, then a sender will need to buffer 180 kB of data to fully utilize the available bandwidth with a single traffic flow. For this to be effective, the sender and receiver will need to agree on the use of TCP Window Scaling to extend the available window size beyond the protocol default limit of 64 kB. A sender using an 8 kB buffer would be able to achieve a maximum transfer rate of 91 kbps, irrespective of the available bandwidth on the satellite path.

Even with advanced FEC techniques, satellite channels exhibit a higher BER than typical terrestrial networks. TCP interprets packet drop as a signal of network congestion, and reduces its window size in an attempt to alleviate the situation. In the absence of certain knowledge about whether a packet was dropped because of congestion or corruption, TCP must assume the drop was caused by congestion in order to avoid congestion collapse[7, 8]. Therefore, packets dropped because of corruption cause TCP to reduce the size of its sending window, even though these packet drops do not signal congestion in the network. To mitigate this, some care must be taken with the satellite hop *Maximum Transmission Unit* (MTU) size, to reduce the probability of packet corruption. This is an area of compromise, in that the consequence is the potential for a high level of IP packet fragmentation on the satellite feeder router. In addition, the sender needs to use the TCP fast retransmit and fast recovery algorithms[9] in order to recover from the packet loss in a rapid, but stable fashion. In addition, the sender needs to use larger sending windows to operate the path more efficiently, with a consequent risk of multiple packet drops per RTT window. For this reason the use of *Selective Acknowledgements* (SACKs) is necessary in order to recover from multiple packet drops in a single RTT interval.

The long delay causes TCP to react slowly to the prevailing conditions within the network. The slow start of TCP commences with a single packet exchange, and it takes some number of RTT intervals for the sender's rate to reach the same order of size as the delay bandwidth product of the long delay path. For short-duration TCP transactions, such as much of the current Web traffic, this is a potential source of inefficiency. For example, if a transaction requires the transfer of ten packets, the slow-start algorithm will send a single packet in the first RTT interval, two in the second interval, four in the third, and the remaining three packets in the fourth RTT interval. Irrespective of the available bandwidth of the path, the transaction will take a minimum of four RTT intervals. This theoretical model is further exacerbated by delayed ACKs [RFC 1122], where a receiver will not immediately ACK a packet, but will await the expiration of the 500ms ACK timer, or a second full-sized packet. During slow start, where a sender sends an initial packet, and then awaits an ACK, the receiver will delay the ACK until the expiration of the delayed ACK timer, adding up to 500ms additional delay in the first data exchange. The second part of the delayed ACK algorithm is that it will only ACK every second full-sized data packet, slowing down the window inflation rate of slow start. Also, if congestion occurs on the forward data path, the TCP sender will not be aware of the condition until it receives duplicate ACKs from the receiver. A congestion condition may take many RTT intervals to clear, and in the case of a satellite path, transient congestions may take tens of seconds to be resolved.

The TCP mechanisms that assist in mitigating some of the more serious effects of satellite systems include *Path MTU Discovery*[10], *Fast Retransmit* and *Fast Recovery,* window scaling options, in order to extend the sender's buffer beyond 65,535 bytes[11], and the companion mechanisms of *Protection Against Wrapped Sequence Space* (PAWS) and *Round-Trip Time Measurements* (RTTM) and SACKs[12]. A summary of TCP options is shown in Figure 2.

Figure 2: TCP Options for Satellite Paths (after RFC 2488)

Mechanism	Use	Location
Path-MTU Discovery	**Recommended**	**Sender**
FEC	**Recommended**	**Link**
TCP		
Slow Start	**Required**	**Sender**
Congestion Avoidance	**Required**	**Sender**
Fast Retransmit	**Recommended**	**Sender**
Fast Recovery	**Recommended**	**Sender**
Window Scaling	**Recommended**	**Sender and Receiver**
PAWS	**Recommended**	**Sender and Receiver**
RTTM	**Recommended**	**Sender and Receiver**
SACK	**Recommended**	**Sender and Receiver**

Further refinements to the TCP stack have been considered in relation to satellite performance[13].

The options considered include the use of T/TCP as a means of reducing the overhead of the initial TCP three-way handshake. This is effective for short transactions where the data to be transferred can be held in a single packet, or in a small number of packets.

The use of delayed acknowledgements also is an issue for long-delay network paths, particularly if the sender is using slow start with an initial window of a single segment. In this case, the receiver will not immediately acknowledge the initial packet, but will wait up to one-half second for the delayed ACK timer to trigger. Altering the initial window size to two segments allows the receiver to trigger an ACK on reception of the second packet, bypassing the delayed ACK timer. However, even this change to TCP does not completely address the performance issue relating to delayed ACKs on long delay paths for TCP slow start. The delayed ACK algorithm triggers an ACK on every second full-sized packet. Because the sender's congestion window is opened on receipt of ACKs, this causes the slow-start window to open more slowly than if the receiver generated an ACK every packet. One variant of TCP congestion control allows the TCP sender to count the number of bytes acknowledged in an ACK message to control the expansion of the congestion window, making the algorithm less sensitive to delayed ACKs[9]. Although this approach has some merit for long delay paths, this is a case where the correction is potentially as bad as the original problem. The byte counting mode of congestion control allows a sender to sharply increase its sending rate, causing potential instabilities within the network and impacting concurrent TCP sessions.

One approach to address this is to place a limit on the size of the window expansion, where each increment of the congestion window is limited to the minimum of one or two segment sizes and the size of the data spanned by the ACK. If the limit is set to a single segment size, the window expansion will be in general slightly more conservative to the current TCP ACK-based expansion mechanism. If this upper limit is set to two segments, the congestion window expansion will account for the delayed ACKs, expand at a rate equal to one segment for every successfully transmitted segment during slow start, and expand the window by one segment size each RTT during congestion avoidance. Because a TCP receiver will ACK a large span of data following recovery, this byte counting is bounded to a single segment per ACK in the slow-start phase following a transmission timeout. Another approach that has been explored is for the receiver to disable delayed ACKs until the sender has completed the slow-start phase. Although such an approach shows promising results under simulated conditions, the practical difficulty is that it is difficult for the receiver to remotely determine the current TCP sending state, and the receiver cannot reliably tell if the sender is in slow start, congestion avoidance, or in some form of recovery mode. Explicit signaling of the sender's state as a TCP flag is an option, but the one-half RTT delay in the signaling from the sender to the receiver may prove to be an issue here. This area of congestion control for TCP remains a topic of study.

All of these approaches can mitigate only the worst of the effects of the long delay paths. TCP, as an adaptive reliable protocol that uses end-to-end flow control, can undertake only incremental adjustments in its flow rates in intervals of round-trip times. When the round-trip times extend, then TCP is slower to speed up from an initial start, slower to recover from packet loss, and slower to react to network congestion.

Tuning TCP—ACK Manipulation

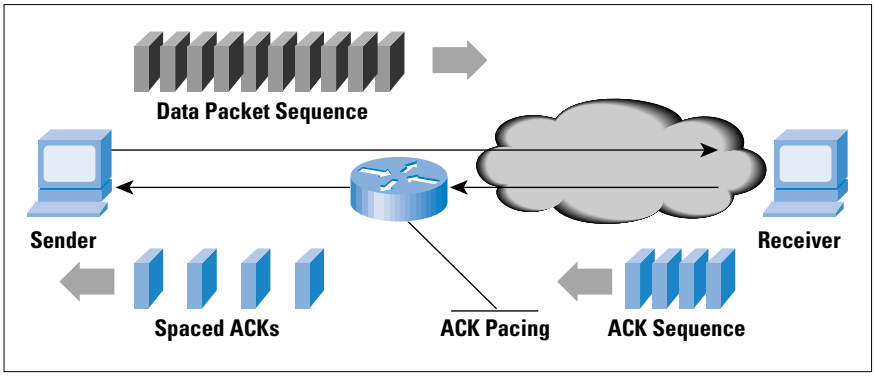
The previous article of TCP Performance discussed numerous network responses to congestion using *Random Early Detection* (RED) for active queue control and *Explicit Congestion Notification* (ECN) as an alternative to RED packet drop. It is feasible for a network control point to impose a finer level of control on a TCP flow by using an approach of direct manipulation of the TCP packets.

The approaches described above to mitigate some of the side effects of satellite paths all share in the side effect of having some latency associated with the congestion response. The sender must await the reception of trailing packets by the receiver, and then await the reception of the matching ACK packets from the data receiver back to the sender to learn of the fate of the original data packet. This may take up to one RTT interval to complete. An alternative approach to congestion management responses is to manipulate the ACK packets to modify the sender's behavior.

The prerequisite to perform this manipulation is that the traffic path be symmetric, so that the congestion point can identify ACK packets traveling in the opposite direction. If this is the case, a couple of control alternatives can mitigate the onset of congestion:

- *ACK Pacing:* Each burst of data packets will generate a corresponding burst of ACK packets. The spacing of these ACK packets determines the burst rate of the next sending packet sequence. For long-delay systems, the size of such bursts becomes a limiting factor. TCP slow start generates packet bursts at twice the bottleneck data rate, so that the bottleneck feeder router may have to absorb one-half of every packet burst within its internal queues. If these queues are not dimensioned to the delay bandwidth product of the next hop, these queues become the limiting factor, rather than the path bandwidth itself. If you can slow down the TCP burst rate, the pressure on the feeder queue is alleviated. One approach to slow down the burst rate is to impose a delay on successive ACKs at a network control point (Figure 3). This measure will reduce the burst rate, but not impact the overall TCP throughput. ACK pacing is most effective on long delay paths, and it is intended to spread out the burst load, reducing the pressure on the bottleneck queue and increasing the actual data throughput.

Figure 3: ACK Pacing



- *Window Manipulation:* Each ACK packet carries a receiver window size. This advertised window determines the maximum burst size available to the sender. Manipulating this window size downward allows a control point to control the maximal TCP sending rate. This manipulation can be done as part of a traffic-shaping control point, enforcing bandwidth limitations on a flow or set of flows.

Both of these mechanisms make some sweeping assumptions about the network control point that must be carefully understood. The major assumption is that these mechanisms assume symmetry of data flows at the network control point, where the data and the associated ACKs flow through this control point (but in opposite directions, of course). Both mechanisms also assume that the control point can cache per-flow state information, so that the current flow RTT and the current transfer rate and receiver window size are available to the service controller.

ACK pacing also implicitly assumes that a single ACK timing response is active at any time along a network path. A sequence of ACK delay actions may cause the sender's timers to trigger, and the sender to close down the transfer and reenter slow-start mode. These environmental conditions are more common at the edge of the network, and such mechanisms are often part of a traffic control system for Web-hosting platforms or similar network service delivery platforms. As a network control tool, ACK manipulation makes too many assumptions, and the per-flow congestion state information represents a significant overhead for large network systems. In general, such manipulations are more appropriate as an edge traffic filter, rather than as an effective congestion management response. For this reason, the more indirect approach of selective data packet discard is more effective as a congestion management measure.

Assisting Short-Duration TCP Sessions—Limited Transmit

One of the challenges to the original set of TCP assumptions is that of short-duration TCP sessions. The Web has introduced a large number of short-duration sessions, and the issue with these sessions is that they use small initial windows. If congestion loss occurs within this early period of TCP slow start, there are not enough packets in the network to generate the three duplicate ACKs required to initiate fast retransmit and fast recovery. Instead the TCP sender must await the expiry of the *retransmission timeout* (RTO), a timer that uses a minimum value of one second. For short-duration TCP sessions that may last six or seven RTT intervals of a small number of milliseconds, the incremental penalty of single packet loss is then extremely severe. A study of this problem indicates that approximately 56 percent of retransmissions are sent following an RTO timeout[25].

One potential mitigation to this is a mechanism termed "Limited Transmit." With this mechanism, a duplicate ACK may trigger an immediate transmission of a segment of new data. Two conditions are applied to this; the receiver's advertised window allows the transmission of this segment, and the amount of outstanding data would remain less than the congestion window plus the duplicate ACK threshold used to trigger Fast Retransmit. This second condition implies that the sender can send only two segments beyond the congestion window, and will do so only in response to the receiver lifting a segment off the network. The basic principle of this strategy is to continue the signaling between the sender and receiver in the face of packet loss, increasing the probability that the sender will recover from packet loss using duplicate ACKs and fast recovery, and reducing the probability of the one-second (or longer) RTO timeout as being the recovery trigger. The limited transmit also reduces the potential for the recovery actions to burst into the network at a level that may cause further packet loss.

Low Bandwidth and High Error Rates—TCP for Wireless Systems

One of the more challenging environments for a the Internet Protocol, and TCP in particular, is that of mobile wireless.

One approach to supporting the wireless environment is that of the so-called "walled garden." Here the protocols in use within the wireless environment are specifically adapted to the wireless world. The transport protocols can account for the low bandwidth, the longer latency, the BERs, and the variability within all three of these metrics. In this model, Internet applications interact with an application gateway to reach the wireless world, and the application gateway uses a wireless transport protocol and potentially a modified version of the application data to interact with the mobile wireless device. The most common approach is extension of the World Wide Web client into the mobile wireless device, using some form of proxy server at the boundary of the wireless network and the Internet. This is the approach adopted by the *Wireless Access Protocol Forum* (WAP)[14].

An alternative approach lies in extending not only the World Wide Web to a mobile handset, but also allowing mobile devices to access a complete range of Internet-based services as the functional objective. In this approach, the intent is to allow the mobile wireless device to function as any other Internet-connected device, and there is a consequent requirement for some form of end-to-end direct IP continuity, and an associated requirement for end-to-end TCP functionality, where the TCP path straddles both wired and wireless segments. Ensuring the efficient operation of TCP in this environment is an integral part of the development of such an environment. Given that TCP must now work within a broader environment, it is no longer a case of adjusting TCP to match the requirements of the wireless environment, but one of attempting to provide seamless interworking between the wired and wireless worlds (Figure 4).

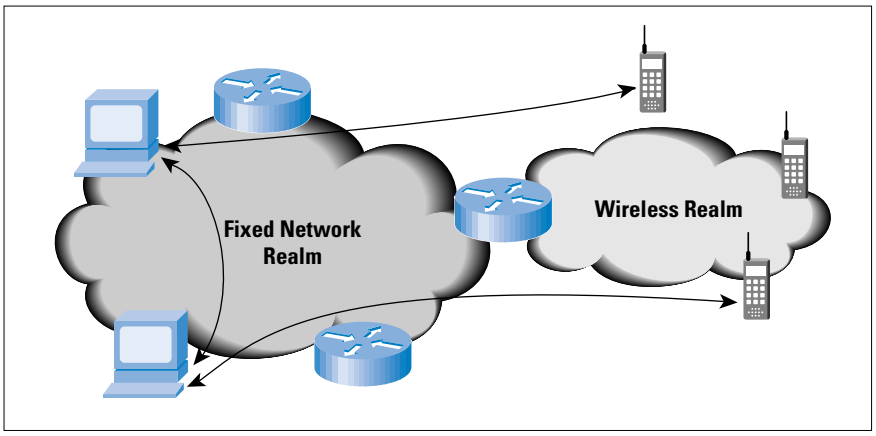


Figure 4: Linking the Wired and Wireless Worlds

The wireless environment challenges many of the basic assumptions of TCP noted above. Wireless has significant levels of bit error rates, often with bursting of very high error rates. Wireless links that use forward error correcting codes have higher latency. If the link level protocol includes automatic retransmission of corrupted data, this latency will have high variability. Wireless links may also use adaptive coding techniques that adjust to the prevailing signal to noise ratio of the link, in which case the link will have varying bandwidth. If the wireless device is a hand-held mobile device, it may also be memory constrained. And finally, such an environment is typically used to support short duration TCP sessions.

The major factor for mobile wireless is the BER, where frame loss of up to 1 percent is not uncommon, and errors occur in bursts, rather than as evenly spaced bit errors in the packet stream. In the case of TCP, such error conditions force the TCP sender to initially attempt fast retransmit of the missing segments, and when this does not correct the condition, the sender will have an ACK timeout occur, causing the sender to collapse its sending window and recommence from the point of packet loss in slow-start mode. The heart of this problem is that assumption on the part of TCP that packet loss is a symptom of network congestion rather than packet corruption. It is possible to use a model of TCP AIMD performance to determine the effects of this loss rate on TCP performance. If, for example the link has a 1-percent average packet loss rate, a *Maximum Segment Size* (MSS) size of 1000 bytes, and a 120ms RTT, then the AIMD models predict a best-case performance of 666Kbps throughput, and a more realistic target of 402Kbps throughput[15]. (See the appendix on page 24 for details of these models.) TCP is very sensitive to packet loss levels, and sustainable performance rapidly drops when packet drop levels exceed 1 percent.

Link-level solutions to the high BER are available to designers, and FEC codes and *automatic retransmission systems* (ARQ) can be used on the wireless link. FEC introduces a relatively constant coding delay and a bandwidth overhead into the path, but cannot correct all forms of bit error corruption. ARQ uses a "stop and resend" control mechanism similar to TCP itself. The consequent behavior is one of individual packets experiencing extended latency as the ARQ mechanisms retransmit link-level fragments to correct the data corruption, because the packet flow may halt for an entire link RTT interval for the link-level error to be signaled and the corrupted level 2 data to be retransmitted. The issue here is that TCP may integrate these extended latencies into its RTT estimate, making TCP assume a far higher latency on the path than is the case, or, more likely, it may trigger a retransmission at the same time as the level 2 ARQ is already retransmitting the same data. An alternative Layer 2 approach to bit-level corruption is to deliver those level 2 frames that were successfully transmitted, while resending any frames that were corrupted in transmission.

The problem for TCP here is that the level 2 drivers are adding packet reordering to the extended latency, and from TCP perspective the delivery of the out-of-order packets will generate duplicate ACKs that may trigger a simultaneous TCP fast retransmit.

Perversely, some approaches have advocated TCP delaying its duplicate ACK response in such situations[13]. To quote from RFC 2488, "The interaction between link-level retransmission and transport-level retransmission is not well understood."[6]

If ARQ is not the best possible answer to addressing packet loss in mobile wireless systems, then what can be done at the TCP level to address this? TCP can take numerous basic steps to alleviate the worst aspects of packet corruption on TCP performance. These include the use of Fast Retransmit and Fast Recovery to allow a single packet loss to be repaired moderately quickly. This mechanism triggers only after three duplicate ACKs, so the associated action is to ensure that the TCP sender and receiver can advertise buffers of greater than four times the MSS. SACKs allow a sender to repair multiple segment losses per window within a single RTT, and where large windows are operated over long delay paths, SACK is undoubtedly useful.

However, useful as these mechanisms may be, they are probably inadequate to allow TCP to function efficiently over all forms of wireless systems. Particularly in the case of mobile wireless systems, packet corruption is sufficiently common that, for TCP to work efficiently, some form of explicit addressing of network packet corruption appears to be necessary.

One approach is to decouple TCP congestion control mechanisms from data recovery actions. The intent is to allow new data to be sent during recovery to sustain TCP ACK clocking. This approach is termed *Forward Acknowledgements with Rate Halving* (FACK)[13], where one packet is sent for every two ACKs received while TCP is recovering from lost packets. This algorithm effectively reduces the sending rate by one-half within one RTT interval, but does not freeze the sender to wait the draining on one-half of the congestion window's amount of data from the network before proceeding to sending further data, nor does it permit the sender to burst retransmissions into the network. This is particularly effective for long-delay networks, where the fast recovery algorithm causes the sender to cease sending for up to one RTT interval, thereby losing the accuracy of the implicit ACK clock for the session. FACK allows the sender to continue to send packets into the network during this period, in an effort to allow the sender to maintain an accurate view of the ACK clock. FACK also provides an ability to set the number of SACK blocks that specify a missing segment before resending the segment, allowing the sender greater levels of control over sensitivity to packet reordering. The changes to TCP to support FACK are a change in the sender's TCP to use the FACK algorithm for recovery, and, for optimal performance, use of SACK options by the receiver.

In looking for alternative responses to packet corruption, it is noted that TCP segments that are corrupted are often detected at the link level, and are discarded by the link-level drivers. This discard cannot be used to generate an error message to the packet sender, given that the IP header of the packet may itself be corrupted, nor can the discard signal be reliably passed to the receiver, for the same reason. However, despite this unreliability of information, this signaling from the link level to the transport level is precisely the objective here, because, at the TCP protocol level, the sender needs to be aware that the packet loss was not due to network congestion, and that there is no need to take corrective action in terms of TCP congestion behavior.

One approach to provide this signaling from the data link level to the transport level calls for the link-level device to forward a "corruption experienced" *Internet Control Message Protocol* (ICMP) packet when discarding a corrupted packet[13]. This approach has the ICMP packet being sent in the forward direction to the receiver, who then has the task of converting this message and the associated lost packet information into a signal to the sender that the duplicate ACKs are the result of corruption, not network congestion. This signal from the receiver to the sender can be embedded in a TCP header option. The sending TCP session will maintain a corruption experienced state for two RTT intervals, retransmitting the lost packets without halving the congestion window size.

As we have noticed, corruption may have occurred in the packet header, and the sender's address may not be reliable. This approach addresses this by having the router keep a cache of recent packet destinations, and when the IP header information is unreliable because of a failed IP header checksum, the router will forward the ICMP message to all destinations in the cache. The potential weakness in this approach is that if network congestion occurs at the same time as packet corruption, the sender will not react to the congestion, and will continue to send into the congestion for a further two RTT intervals. This approach is not without some deployment concerns. It calls for modification to the wireless routers and to the receiver's link-level drivers to generate the ICMP corruption experienced messages, modification to the receiver's IP stack in order to take signals from the IP ICMP processor and from the link-level driver and convert them to TCP corruption loss signals within the TCP header of the duplicate ACKs, and modifications to the TCP processor at the sender to undertake corruption-experienced packet loss recovery. Even with these caveats in mind, this approach of explicit corruption signaling is a very promising approach to addressing performance issues with TCP over wireless.

Of course high levels of bit errors is not the only problem facing TCP over wireless systems. Mobile wireless systems are typically small handsets or personal digital assistants, and the application transactions are often modified to reduce the amount of data transferred, given that a limited amount of data can be displayed on the device.

In this case, the ratio between payload and IP and TCP headers starts to become an issue, and some consideration of header compression is necessary. Header compression techniques typically take the form of stripping out those fields of the header that do not vary on a packet-by-packet basis, or that vary by amounts that can be derived from other parts of the header, and then transmitting the delta values of those fields that are varying[16, 17].

Although such header compression schemes can be highly efficient in operation, the limitation of such schemes is that the receiver needs to have successfully received and decompressed the previous packet before the receiver can decompress the next packet in the TCP stream. In the face of high levels of bit error corruption, such systems do introduce additional latencies into the data transfer, and multiple packet drops are difficult to detect and signal via SACK in this case.

A more subtle aspect of mobile wireless is that of temporary link outages. For example, a mobile user may enter an area of no signal coverage for a period of time, and attempt to resume the data stream when signal is obtained again. In the same way that there is no accepted way of a link-level driver informing TCP of packet loss due to corruption, there is no way a link-level driver can inform TCP of a link-level outage. In the face of such link-level outages, TCP will assume network-level congestion, and in the absence of duplicate ACKs, TCP retransmission timers will trigger. TCP will then attempt to restart the session in slow-start mode, commencing with the first dropped packet. Each attempt to send the packet will result in TCP extending its retransmission timer using an exponential backoff on each attempt, so that successive probes are less and less frequent. Because the link level cannot inform the sender on the resumption of the link, TCP may wait some considerable time before responding to link restoration. The intention is for the link level to be able to inform the TCP for resumption of the connection following a link outage. One approach is for the link level to retain a packet from each TCP stream that attempted to use the link. When the link becomes operational again, the link-level driver immediately transmits these packets on the link. The result is that the receiver will then generate a response that will then trigger the sender into transmission within a RTT interval. Only a single packet per active TCP stream is necessary to trigger this response, so that the link level does not need to hold an extensive buffer of undeliverable packets during a link outage. Of course if the routing level repaired the link outage in the meantime, the delivery of an out-of-order TCP packet would normally be discarded by the sender.

The bottom line here is the question: Is TCP suitable for the mobile wireless environment? The answer appears to be that TCP can be made to work as efficiently as any other transport protocol for the mobile wireless environment.

However, this does imply that some changes in the operation of TCP need to be undertaken, specifically relating to the signaling of link-level states into the TCP session and use of advanced congestion control and corruption signaling within the TCP session. Although it is difficult to conceive of a change to every deployed TCP stack within the deployed Internet to achieve this added functionality, there does exist a middle ground between the "walled garden" approach and open IP. In this middle ground, the wireless systems would have access to "middleware," such as Web proxies and mail agents. These proxies would use a set of TCP options when communicating with mobile wireless clients that would make the application operate as efficiently as possible, while still permitting the mobile device transparent access to the Internet for other transactions.

Unbundling TCP—Stream Control Transmission Protocol

There are occasions where the application finds the control functions of TCP too limiting. In the case of handling *Public Switched Telephone Network* (PSTN) signaling across an Internet network, the application requirements are somewhat different from those of TCP delivered service. PSTN signaling reliable delivery is important, but the individual transactions within the application are included within each packet, so the concept of preservation of strict order of delivery is unnecessary. Relaxation of this requirement of strict order of packet delivery allows the transport protocol to function more efficiently, because there is no head-of-line blocking at the receiver when awaiting retransmission of lost packets. TCP also assumes the transfer of a stream of data, so that applications that wish to add some form of record delineation to the data stream have to add their own structure to the data stream. In addition, the limited scope of TCP sockets complicates the support of a high-availability application that may use multihomed hosts, and TCP itself is vulnerable to many attacks, such as SYN attacks. The intention of the *Stream Control Transmission Protocol* (SCTP) is to address these application requirements[16].

The first major difference between SCTP and TCP occurs during initialization, where the SCTP endpoints exchange a list of SCTP endpoint addresses (IP addresses and port numbers) that are to be associated with the SCTP session. Any pair of these source and destination addresses can be used within the SCTP session.

The startup of SCTP is also altered into a four-way handshake, where the initiator sends a tag value to the other end, which then responds with a copy of this tag and a tag of its own. At this stage the recipient does not allocate any resources for the connection, making the initialization sequence more robust in the face of TCP SYN-styled attacks. The initiator can then respond to this with an echo of the recipient's tag (COOKIE-ECHO), and can also attach data to the response, allowing data to be transferred as early as possible in the handshake process.

After the recipient ACKs this message, the SCTP session is now established. The closing of an SCTP session is also different from TCP. In TCP, one side can close its sending function via a FIN TCP packet, and continue to receive packets, operating in a "half-open" state. In SCTP, a close from one side will cause the other end to drain its send queues and also shut down.

SCTP also functions in a form of transport-level multiplexing, where numerous logical streams can be supported across a single transport-level association. Although message order within an individual stream is preserved by SCTP, retransmission within one stream does not impact the operation of any other stream that is supported across the same SCTP transport association. Each stream has an explicit identification and a per-stream sequence identification to support this function. SCTP also provides for nonsequenced message delivery, where a message within a stream is marked for immediate delivery, irrespective of the relative order of the message within a stream (Figure 5).

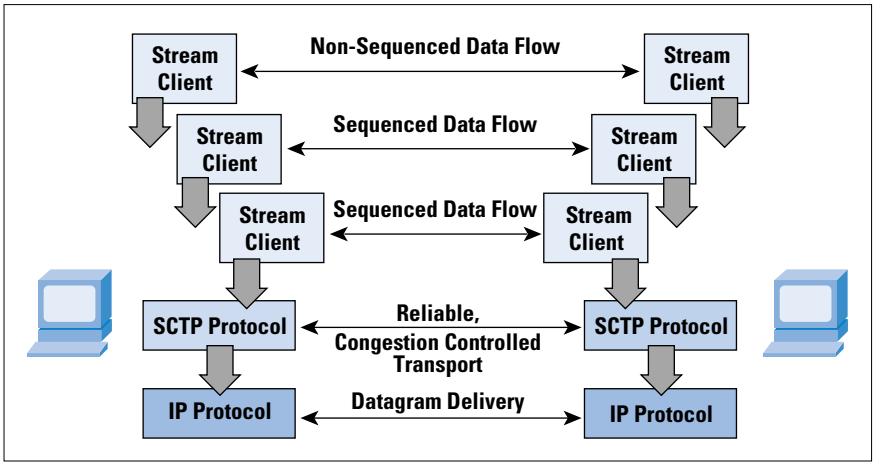


Figure 5: The SCTP Transport Service Model

SCTP explicitly uncouples transport-level reliability and congestion control from per-stream sequenced delivery through the use of a separate transport-level interaction. The transport-level data and ACKs and the corresponding transport-level congestion window controls operate using a transport-level sequence space. This sequence space counts transport-level messages, not byte offsets within the message, so that no explicit window scaling option is necessary for SCTP. The congestion control functions reference those of TCP with fast retransmit and fast recovery, with an explicit specification of the SACK protocol and specification of the maintenance of the transmission timers and congestion control. SCTP also requires the use of MTU path discovery, so that larger transactions will use SCTP-level segmentation, avoiding the IP retransmission problem with lost fragments of a fragmented IP packet. SCTP does use a modified retransmission mechanism to that of TCP. Like TCP, SCTP associates a retransmission timer with each message, and if the timer expires the message is retransmitted and SCTP collapses the congestion window to a single message size. The SCTP receiver will generate SACK reports for a minimum of every second received packet.

If a message is within a SACK gap, then after three further such SACK messages, the sender will immediately send the missing messages, and half its congestion window, analogous to the fast retransmit and fast recovery of TCP.

The use of multiple endpoint addresses assumes that each of the endpoint addresses is associated with the same end host, but with a potentially different network path between the two endpoints. SCTP refreshes path availability to each of the endpoint addresses with a periodic keepalive, so that in the event of primary path failure, SCTP can continue by using one of the secondary endpoint addresses.

One could describe SCTP as being overly inclusive in terms of its architecture, and there is certainly a lot of capability in the protocol that is not contained within TCP. The essential feature of the protocol is to use a single transport congestion state between two systems to allow a variety of applications to attach as stream clients. In itself, this is analogous to TCP multiplexing. It also implicitly assumes that every stream is provided the same service level by the network, an assumption shared by almost all transport multiplexing systems. The essential alteration with SCTP is the use of many transport modes: reliable sequenced message streams, reliable sequenced streams with interrupt message capability, and reliable nonsequenced streams. It remains to be seen whether the utility provided by this protocol will become widely deployed within the Internet environment, or whether it will act as a catalyst for further evolution of transport service protocols.

Sharing TCP information—Endpoint Congestion Management

The notion of sharing a single TCP congestion state across multiple reliable streams is one that may also be applied to a mix of reliable and nonreliable data streams that operate concurrently between a pair of endpoints. It is this form of the multiplexing service model that is explored by the congestion manager model. The Congestion Manager is an end-system module that allows a collection of concurrent streams from the host to a single destination to share a common congestion control function, and permits various forms of reliable and nonreliable streams to use the network in a way that cooperates with concurrent congestion controlled flows[19].

One of the major motivations for the congestion manager is the observation that the most critical part of network performance management is that of managing the interaction between congestion-controlled TCP streams and nonresponsive UDP data streams. In the extreme cases of this interaction, either traffic class can effectively deny service to the other by placing sufficient pressure on the network queuing resources that starve the other traffic class of any usable throughput. The observation made in the motivation for the congestion manager is that applications such as the Web typically open up a set of parallel connections to provide service, sending a mix of reliable flow-controlled data

along one connection and unreliable real-time streaming content along another. If the set of flows used a common congestion-control function at the sending host, the collection of flows would utilize the network resources in a manner analogous to a single TCP connection.

The manner of providing this common congestion control function is an advisory function to applications, as shown in Figure 6. One mechanism is that of a *callback*, where an application inserts a request to send a single message segment with the congestion manager. The Congestion Manager responds with invoking a callback to the requestor when the application may pass the data segment to the protocol driver. The other supported mechanism is that of *synchronous transmission*, where the Congestion Manager has a callback function that updates the application with a maximal available bit rate, the smoothed round-trip time estimate, and the smoothed linear deviation in the round-trip time estimate. In this mode the application can request further notification only when the network state changes by some threshold amount.

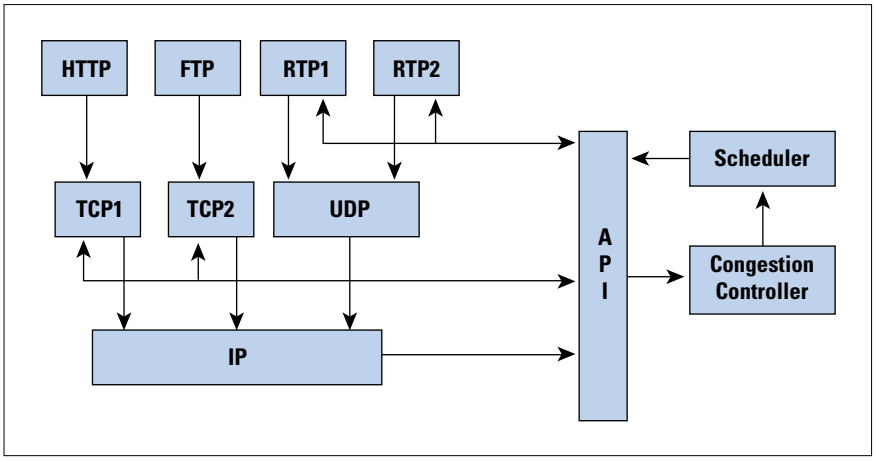


Figure 6: The CM Model, (after "The Congestion Manager"[19])

For the Congestion Manager to maintain a current picture of the congestion state of the path to the destination, each active stream needs to update the congestion manager as to the response from the remote host. It does this by informing the congestion manager of the number of bytes received, the number of bytes lost, and the RTT measurement, as measured at the application level. The application is also expected to provide an indication of the nature of the loss, as a timeout expiry, a transient network condition, or based on the reception of an ECN signal.

There has been little practical experience as yet with this model of shared congestion control within the Internet environment. There also remains a number of issues about how network performance information is passed back from the receiver to the sender in the absence of an active concurrent TCP session. The concurrent operation of a TCP session with a UDP streaming session to the same destination allows Congestion Manager to use the TCP congestion state to determine the sending capability of the streaming flow.

If the TCP session is idle, or if there is no TCP session, then the UDP streaming application will require some form of receiver feedback. The feedback will need to report on the span of data covered by the report, and the data loss rates and jitter levels, allowing the sender to assess the current quality and capacity of the network path.

This approach, and that of SCTP, are both illustrative of the approach of unbundling the elements of TCP and allowing applications to use combinations of these elements in ways that differ from the conventional monolithic transport-level protocol stack, with the intention of allowing the TCP congestion control behavior to be applied to a wider family of applications.

Better than TCP?

Recently, numerous "better-than-TCP" protocol stacks have appeared on the market, most commonly in conjunction with Web server systems, where the performance claim is that these protocol stacks can interoperate with standard TCP clients, but offer superior download performance to a standard TCP protocol implementation.

This level of performance is achieved by modifying the standard TCP flow control systems in a number of ways. The modified implementation may use a lower initial RTT estimate to provide a more aggressive startup rate, and a more finely grained RTT timer system to allow the sender to react more quickly to network state changes. Other modifications may include using a larger initial congestion window size or may use an even faster version of slow start, where the sending rate is tripled, or more, every round-trip time interval. The same technique of incremental modification can be applied to the congestion avoidance state, where the linear rate increase of one segment size per round-trip time interval can be increased to some multiple of the segment size, or use a time base other than the round-trip time for linear expansion of the congestion window. The backoff algorithm can also be altered such that the congestion window is reduced by less than half during congestion backoff. Resetting the TCP session to slow-start mode following the ACK timeout can also be avoided in such modified protocol implementations.

These techniques are all intended to force the sender to behave more aggressively in its transmission of packets into the network, thereby increasing the pressure on the network buffers. The network is not the only subject of this increased sending pressure; such modified protocol systems tend to impose a significant performance penalty on other concurrent TCP sessions that share the path with these modified protocol hosts. The aggressive behavior of the modified TCP systems in filling the network queues tends to cause the other concurrent standard TCP sessions to reduce their sending rate. This in turn opens additional space in the network for the modified TCP session to increase its transmission rate.

In an environment where the overall network resource-sharing algorithm is the outcome of dynamic equilibration between cooperative sending systems, such aggressive flow control modification can be considered to be extremely antisocial behavior at the network level. Paradoxically, such systems can also be less efficient than a standard TCP implementation. TCP server systems modified in this way tend to operate with higher levels of packet loss because their efforts to saturate the network with their own data packets make them less sensitive to the signals of network congestion.

Consequently, when delivering large volumes of traffic, or where there are moderately low levels of competitive pressure for network resources, the modified TCP stack may often perform less efficiently than a standard TCP implementation. Accordingly, these modified better-than-TCP implementations remain in the experimental domain. Within the production environment, their potential to impose undue performance penalties on concurrent TCP sessions and their potential to reduce overall network efficiency are reasonable indicators that such modified stacks should be used in private network environments, and with considerable care and discretion, if at all. Their utility in the public Internet is highly dubious.

TCP Evolution

The evolution of TCP is a careful balance between innovation and considered constraint. The evolution of TCP must avoid making radical changes that may stress the deployed network into congestion collapse, and also must avoid a congestion control "arms race" among competing protocols[20]. The Internet architecture to date has been able to achieve new benchmarks of network efficiency, and translate this carriage efficiency into ground-breaking benchmark prices for IP-based carriage services. Much of the credit for this must go to the operation of TCP, which manages to work at that point of delicate balance between self-optimization and cooperative behavior.

Widespread deployment of transport protocols that take a more aggressive position on self-optimization will ultimately lead to situations of congestion collapse, while widespread deployment of more conservative transport protocols may well lead to lower jitter and lower packet retransmission rates, but at a cost of considerably lower network efficiency.

The challenges faced with the evolution of TCP is to maintain a coherent control architecture that has consistent behavior within the network, consistent interaction with instances of data flows that use the same control architecture, and yet be adequately flexible to adapt to differing network characteristics and differing application profiles. It is highly likely that we will see continued innovation within Internet transport protocols, but the bounds of such effort are already well recognized.

We can now state relatively clearly what levels of innovation are tolerable within an Internet network model that achieves its efficiency not through enforcement of rigidly enforced rules of sharing of the network resource, but through a process of trust between competing user demands, where each demand is attempting to equilibrate its requirements against a finite network capacity. This is the essence of the TCP protocol.

Appendix: TCP Performance Models

This appendix is an extract from "Advice for Internet Subnet Designers," work in progress[15].

The performance of the TCP AIMD Congestion Avoidance algorithm has been extensively analyzed. The current best formula for the performance of the specific algorithms used by Reno TCP is given by Padhye et. al.[21], this formula is:

$$BW = \frac{MSS}{(RTT \times \sqrt{(1.33 \times \rho)}) + (RTO \times \rho \times [1 + 32 \times \rho^2] \times min(1, 3 \times \sqrt{0.75 \times \rho}))}$$

MSS is the segment size being used by the connection.
RTT is the end-to-end round-trip time of the TCP connection.
RTO is the packet timeout (based on RTT).
ρ is the packet loss rate for the path (that is, 0.01 if there is 1-percent packet loss)

This is currently considered to be the best approximate formula for Reno TCP performance. A further simplification to this formula is generally made by assuming that RTO is approximately $5 \times$ RTT.

TCP is constantly being improved. A simpler formula, which gives an upper bound on the performance of any AIMD algorithm that is likely to be implemented in TCP in the future, was derived by Ott, et.al.[22, 23].

$$BW = 0.93 \times \frac{MSS\ 1}{RTT\sqrt{\rho}}$$

Assumptions of these formulae:

- Both of these formulae assume that the TCP Receiver Window is not limiting the performance of the connection in any way. Because the receiver window is entirely determined by end hosts, we assume that hosts will maximize the announced receiver window in order to maximize their network performance.
- Both of these formulae allow for bandwidth to become infinite if there is no loss. This is because an Internet path will drop packets at bottleneck queues if the load is too high. Thus, a completely lossless TCP/IP network can never occur (unless the network is being underutilized).
- The RTT used is the average RTT including queuing delays.

- The formulae are calculations for a single TCP connection. If a path carries many TCP connections, each will follow the formulae above independently.
- The formulae assume long-running TCP connections. For connections that are extremely short (<10 packets) and don't lose any packets, performance is driven by the TCP slow-start algorithm. For connections of medium length, where on average only a few segments are lost, single-connection performance will actually be slightly better than given by the formulae above.
- The difference between the simple and complex formulae above is that the complex formula includes the effects of TCP retransmission timeouts. For very low levels of packet loss (significantly less than 1 percent), timeouts are unlikely to occur, and the formulae lead to very similar results. At higher packet losses (1 percent and above), the complex formula gives a more accurate estimate of performance (which will always be significantly lower than the result from the simple formula).

Note that these formulae break down as ρ approaches 100 percent.

Addendum: An Update on Explicit Congestion Notification

The previous article on TCP performance noted that there was no explicit standardization of the IPv4 header field to carry the *Explicit Congestion Notification* (ECN) signals. As an update to the status of ECN, RFC 2481, the document that describes ECN, categorizes this proposal as an "Experimental" RFC document[27]. The Internet Standards process[28] describes this category as follows: "The 'Experimental' designation typically denotes a specification that is part of some research or development effort. Such a specification is published for the general information of the Internet technical community ..." ECN is the only experimental proposal to use these two bits of the IP header, and the use of the category "Experimental" reflects the current status of the proposal, in that the Internet Engineering Steering Group has, at the time of publication, yet to make a final decision to allocate these two bits of the IP header to ECN.

Some encouragement to use ECN is certainly timely. As RFC 2481 notes: "Given the current effort to implement RED, we believe this is the right time for router vendors to examine how to implement congestion avoidance mechanisms that do not depend on packet drops alone. With the increased deployment of applications and transports sensitive to the delay and loss of a single packet (e.g., realtime traffic, short web transfers), depending on packet loss as a normal congestion notification mechanism appears to be insufficient (or at the very least, non-optimal)."

References and Further Reading

[1] Huston, G., TCP Performance, *The Internet Protocol Journal,* Vol. 3, No. 2, Cisco Systems, June 2000.

[2] Huston, G., *Internet Performance Survival Guide: QoS Strategies for Multiservice Networks,* ISBN 0471-378089, John Wiley & Sons, January 2000.

[3] Postel, J., "Transmission Control Protocol," RFC 793, September 1981.

[4] Claffy, K., Miller, G., Thompson, K., "The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone," INET'98 Proceedings, Internet Society, July 1998. Available at: **http://www.isoc.org/inet98/proceedings/6g/6g_3.htm**

[5] Braden, R., "T/TCP—TCP Extensions for Transactions Functional Specification," RFC 1644, July 1994.

[6] Allman, M., Glover, D., Sanchez, L., "Enhancing TCP over Satellite Channels Using Standard Mechanisms," RFC 2488, January 1999.

[7] Jacobson, V., "Congestion Avoidance and Control," ACM SIGCOMM, 1988.

[8] Floyd, S., Fall, K., "Promoting the Use of End-to-End Congestion Control in the Internet," Submitted to IEEE *Transactions on Networking*.

[9] Allman, M., Paxson, V., Stevens, W., "TCP Congestion Control," RFC 2581, April 1999.

[10] Mogul, J., Deering. S., "Path MTU Discovery," RFC 1191, November 1990.

[11] Jacobson, V., Braden, R., Borman, C., "TCP Extensions for High Performance," RFC 1323, May 1992.

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

[13] Allman, M., editor, "Ongoing TCP Research Related to Satellites," RFC 2760, February 2000.

[14] Wireless Access Protocol Forum, **http://www.wapforum.org**

[15] Karn, P., Falk, A., Touch, J., Montpetit, M., Mahdavi, J., Montenegro, G., Grossman, D., Fairhurst, G., "Advice for Internet Subnet Designers," work in progress, July 2000.

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

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

[18] Stewart, R., et al., "Stream Control Transmission Protocol," work in progress, July 2000.

[19] Balakrishnan, H., Seshan, S., "The Congestion Manager," July 2000.

[20] Floyd, S., editor, "Congestion Control Principles," work in progress, June 2000.

[21] Padhye, J., Firoiu, V., Towsley, D., Kurose, J., Modeling TCP Throughput: A Simple Model and Its Empirical Validation, UMASS CMPSCI Tech Report TR98-008, Feb. 1998.

[22] M. Mathis, M., Semke, J., Mahdavi, J., Ott, T., "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," *Computer Communication Review,* Vol. 27, No. 3, July 1997.

[23] Ott, T., Kemperman, J., Mathis, M., "The Stationary Behavior of Ideal TCP Congestion Avoidance," available at:
ftp://ftp.bellcore.com/pub/tjo/TCPwindow.ps

[24] Floyd, S., Mahdavi, J., Mathis, M., Podolsky M., "An Extension to the Selective Acknowledgement (SACK) Option for TCP," RFC 2883, July 2000.

[25] Allman, M., Balakrishnan, H., Floyd, S., "Enhancing TCP's Loss Recovery Using Early Duplicate Acknowledgment Response," work in progress, June 2000.

[26] Allman, M., "TCP Congestion Control with Appropriate Byte Counting," work in progress, July 2000.

[27] Ramakrishnan, K., Floyd, S., "A Proposal to Add Explicit Congestion Notification (ECN) to IP," RFC 2481, January 1999.

[28] Bradner, S., "The Internet Standards Process—Revision 3," RFC 2026, October 1996.

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 initial build of the Internet within the Australian academic and research sector. Huston is currently the Chief Scientist in the Internet area for Telstra. He is also a member of the Internet Architecture Board, and is the Secretary of the Internet Society Board of Trustees. He is author of *The ISP Survival Guide,* ISBN 0-471-31499-4, *Internet Performance Survival Guide: QoS Strategies for Multiservice Networks,* ISBN 0471-378089, and coauthor of *Quality of Service: Delivering QoS on the Internet and in Corporate Networks,* ISBN 0-471-24358-2, a collaboration with Paul Ferguson. All three books are published by John Wiley & Sons. E-mail: **gih@telstra.net**

Securing the Infrastructure

by Chris Lonvick, Cisco Systems

People are becoming much more reliant upon the proper operation of their networks. Consequently, the administrators of these networks are being tasked with providing an ever-increasing level of service. At this time of high reliance upon the network, methods and procedures need to be instilled into the network so the operators can maintain control of their network and they can know with some certainty the effect of each potential change. This may become increasingly difficult as network resiliency techniques are being proposed and deployed with the intent of automatically keeping these networks in top operation. Having a predictable network that is secured in a proper manner results in a network that is more suitable for the users and better meets the intended purpose of the network.

Most of the current network security models start with the physical perimeter of the network as its defining boundary. All things within this boundary are supposed to be protected from the perceived inimical forces that are outside of the perimeter. We are, however, finding that the perimeter of the network is no longer solidly defined. There are many exceptions to the "hard-shell perimeter" model—companies merge, remote sites are linked through *Virtual Private Networks* (Site-to-Site VPNs) across untrusted paths, access is granted in-bound for the network users through *Access Virtual Private Networks* (Access VPNs), and there are several other exceptions. For this article, let's consider a different model. This model has a boundary of the acceptable network users rather than any geographical or logical perimeter. It is important that these users are allowed access to the services provided by the network. It is equally important that the people who are not authorized to use the network must be prevented from consuming its resources and otherwise disrupting its services.

Other models tend to focus on the restrictions of the users to access devices to provide security to the network. This model, however, looks at the effect that the users and each of the devices have upon the state of the network. To conceptualize this model, visualize that the only time this network would be running at a "steady state" is when there is no user traffic, no administrative or management traffic, and no routing update changes. The insertion of any traffic, or the addition or removal of any device or link, would change the state of this network. Changes to the state of this network may come from any number of sources, but they can be seen as coming from four different, quantifiable areas.

- Operators may enable or disable lines and devices.
- A network device publishing a new route or a different metric to a destination may cause the remainder of the network devices to dynamically recompute paths to all other destinations.
- Servers may insert traffic.
- Users may insert traffic.

Of these, the last two should be the least disruptive to the network as long as the traffic amounts are within the predicted and acceptable ranges. Changes that are within the goals of the network—for example to provide a service to the users—are considered good, while changes that cause outages or other disruptions are to be avoided. As such, it is vital that the network administrators understand the potential impact and consequences of each possible change in their network.

In this model, then, the administrators must know and understand the influences that will change the state of the network. The desire to achieve this goal sometimes leads to improper restrictions placed upon the users. Consider one extreme case of this model where each change in the network must be stringently authorized and authenticated. As a narrow example, this would mean that even traffic that is fundamentally taken for granted as a proper process of the network would have to be authenticated and authorized. *Domain Name System* (DNS) transactions would show that this extreme case is impractical. Each DNS query would have to be associated with a user or authenticated process, and that user or process would have to be authorized to make each specific query. A vastly more practical case for real networks would be for the administrators to allow any DNS query from any device without authentication—as it is done in existing dynamic networks today. In the model, the normal DNS queries and responses would be an influence upon the state of the network. For this influence to change the network in a way that meets the goals of the network, the administrators would have to feel comfortable that the servers and the available bandwidth will adequately handle the amount of DNS traffic as well as all other traffic. On the other hand, the administrators do need to establish a strict set of rules for the influences that they consider sensitive or possibly disruptive to their network. Continuing this example, the administrators may want to place restrictions upon the devices and processes that can insert and update the DNS records. It would be rather inappropriate, and potentially devastating, if any unauthorized person or network device were allowed to overwrite any existing records. If anyone were allowed to perform any DNS update that he or she wished, chaos would soon result. There must be a center position for this example that allows the operators to maintain control but still permits the dynamic freedoms expected by the users. Specifically to address this, the DNS Extensions Working Group has proposed several Internet Drafts[1].

In the broader sense, this places a very heavy responsibility upon the people who are running the network. They must find some acceptable median between the desire to rigidly control all aspects of the network and the freedoms that are expected by the users, while at the same time satisfying the business requirements of their network. However, defining the freedoms and restrictions of the users is only one part of maintaining the network. The administrators and operators must have an understanding of the influences on the network as described in the model. In this, each aspect of the parts of the network must be under-

stood well enough to predict their behavior as they are normally used, and to limit the potential for disruption if they are used beyond their means. The one area that is vital to the proper working of the network is the infrastructure. This article explores some of the thoughts that may go into the process of securing the network infrastructure.

Table 1: Sources of Change to the Network

Sources of Change to the Network	Some Examples of How the Source Influences the Network	Examples of Device Types within the Network (The 4 Groups)	
Operators and their Devices	• Add/remove new lines and circuits • Install/remove network devices		
	• Login to the network devices to change their configuration • Poll network devices for their status	• Operations Consoles • Network Management Stations	Operators
Network Devices	• Dynamically route or switch traffic • Dynamically mark lines and circuits in or out of service and then use them accordingly • Authenticate users and permit their accesses accordingly • Dynamically assign addresses and register that information for retrieval by others	• Routers and Switches • Firewalls	Infrastructure Devices
		• Authentication Servers • DNS/DHCP Servers	
Servers	• Servers send content to User's workstations to fulfill their requests • Servers broadcast and multicast content to recipients	• Servers offering Content and Servers	Servers
Users and their Devices	• Client workstations request content from servers and upload content to servers • Client workstations utilize services that are offered within the network	• Client Workstations	Users
	• A user encourages many others to visit a particular web site which causes a stampede • A user tells others that a particular service is down or unavailable causing others to not attempt access		

Description of Problem

In this abstracted network model, four sources of change were noted. As shown in Table 1, these changes, or influences to the network, may come from the operators, the network devices, the servers, and the users of the network. Let's first look at the influences that each of these groups can effect upon the network by first categorizing the network devices. All the devices on the network may be somewhat separated into four groups that correspond to the four sources. These groups of network devices can be seen in the third column of the table.

- *Operators:* For the purpose of this article, let's describe the Operators as all the people who operate the network, including the network engineers, the installers, the people who monitor the net-

work, and all the other people who make it work. The first group then is made of the operators and the devices that these operators use to run the network, such as the network management stations and all other operations consoles. Operators periodically make changes for moves and additions for better network performance, or to overcome disruptions. They will also monitor the network through polling, receiving alerts, and sometimes directly interacting with the network devices. Generally the amount of traffic inserted into the network from their activities is minimal. Because they generally have physical access to all locations, they can insert or remove network devices. Operators can have influence over all aspects of the network at all layers—from the physical layer, all the way up the stack. Operators can influence the network either in band or out of band, and they should be the only people who directly access the network infrastructure devices such as the routers and DNS servers. Usually this access will be from the management platforms, but in many situations, operators require access from devices that would otherwise be classified as a user's workstation.

- *Infrastructure Devices:* The network infrastructure devices themselves have the ability to change the network as well. This is mostly done through the dynamic nature of the network. At some times the physical portions of the network might fail and cause outages. In some cases, such as self-healing ring topologies, physical-layer devices may heal the network. In other cases, such as when a router is taken out of the network for maintenance, the routing updates will heal the network to the best of their abilities. The network infrastructure devices can be somewhat separated into two categories. The first of these would be the infrastructure devices that have no direct interaction with the users of the network. This category would consist of the devices such as the routers, switches, access control devices, and perhaps even the physical-layer devices such as multiplexers and modems. The user machines and content servers normally would not form sessions or require any information from these devices. The second category would be the devices with which customers indirectly interact. These would be devices such as the DNS servers, *Dynamic Host Configuration Protocol* (DHCP) servers, *Network Time Protocol* (NTP) servers, authentication servers, and the like. The users and servers would form sessions with these supporting devices and would require information from them for the basic operation of the network. In some cases, such as with a DNS/DHCP server, the results of the indirect user interaction would even update the servers with information. This latter group may be called "supporting devices." These two categories can be taken together with all the wires, circuits, and lines to form the infrastructure of the network. Although the users do not actively see their presence, this infrastructure must be available and functioning before any user can actually do anything productive on the network.

- *Servers:* The servers in this group are those that contain content or services with which the users directly interact. These would be databases, Web servers, application servers, and the like. Like the operators group, this group is not considered to be part of the network infrastructure.
- *Users:* The users and their machines constitute the bulk of the network. The changes that the users make upon the network will probably come through transferring content or requesting and utilizing services. They can change the nature of the network by withdrawing from the network, or by causing others to withdraw from the network. In a nonmalicious way, the user base can degrade the state of the network by using it beyond its expected capacity. In certain situations, users with malicious intent may find exploitable network vulnerabilities. In most normal cases, however, the influence from the users upon the network will be through their interactions with the servers.

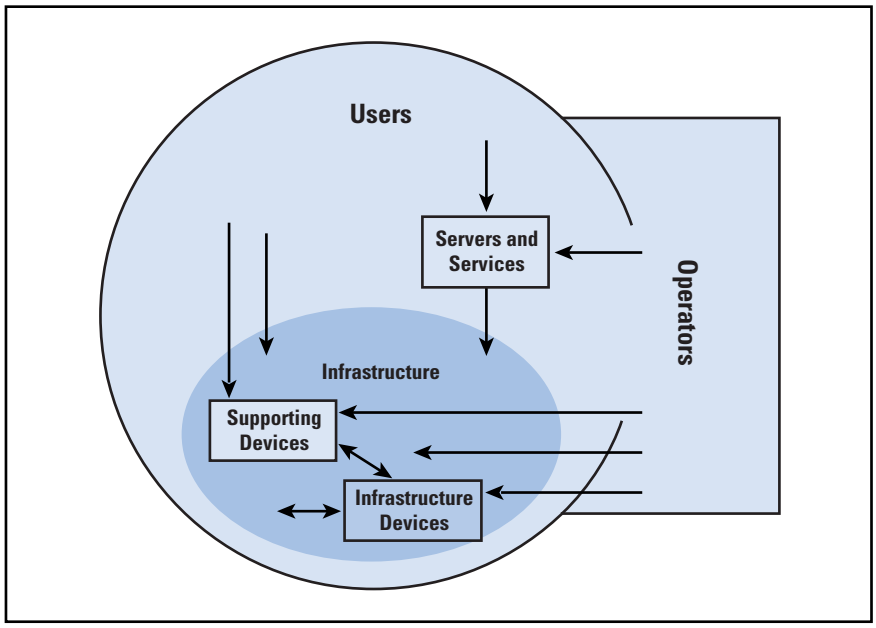
Each type of influence may also be considered to have a different weight. For example, the insertion of a new router into an existing network would be expected to have a larger effect upon the operations of the network than the change to the network caused by a user retrieving some information through a Web browser. To quantify some of the expected network changes, consider that there may be spheres and levels of influence. Any influence that may cause a change over the entire network may be considered to have a global sphere of influence. A router recently inserted into the network would start exchanging routing information with its neighbors. With no restrictions placed upon routing updates, this router could announce a new network, or it could announce the best path to an otherwise difficult-to-reach network. The remainder of the network would be affected, and all other routers would have to recalculate their paths. If the announcements were true, then the network would continue servicing the needs of the users. If the announcements were false, possibly because of an incorrect configuration, then the whole network could suffer. In this case, it is possible to limit the sphere of influence by restricting the acceptance of routing updates. In one method, all the routers could be restricted to disallow the acceptance of an announcement to the "default" network. Additionally, all the routers may be restricted to accept only announcements that are known to be within an acceptable address range. In another method, the routers could be grouped to accept announcements only from a select set of other routers. Additionally, some routing protocols have an option to include an authentication and integrity check through signing the updates. Any of these methods would help to reduce the sphere of influence and thus the potential for changes that could be made by the insertion of a router. There is, however, a cost associated with this; the operators would have to diligently enforce this control.

The level of influence can also be considered a factor in this model. The sphere of influence of a single transmission line can be defined to include any portion of the network that uses that line. If that line develops a fault, it may corrupt or discard packets and the associated network devices may automatically disable that line. If there is a backup line or an alternate path, then this change will be a small problem to the operations staff and its loss may go unnoticed by the users. That would be a low level of influence upon the network. On the other hand, if the line has an intermittent fault that can cause a route flap, or if the line has no backup, then major disruptions may occur. That would be considered a high level of influence.

If the goal of the network is to provide a service to its users, then its operators must try to quantify each of the influences. In a theoretically ideal network, the administrators would appropriately limit the sphere and would try to minimize the level for every influence. As was noted above, however, attempting to do this would require numerous operations tasks. Many of those may be unnecessary for their specific environment. For example, in a small business where there is a high degree of trust that no one has any malicious intent, controls would still be placed upon the influences that would most probably cause network problems through accidents. If the security policy allowed anyone to connect any device to the network, it may still be prudent to disallow the routers from receiving routing updates from any source other than the other routers.

A well-running network is the result of a well-controlled network. These networks must have a separation of authorized administration from other influences, and these other influences must be understood well enough to know how they will change the network. The following diagram shows the network and the groupings of influences upon it, and the table below that describes the elements of this model. This model does not show access paths, but rather the influences that each grouping of devices has upon the infrastructure and upon other devices. As can be seen, the users are pervasive throughout the network (because they are a principal reason for its existence), and they must have the access paths to contact the servers and necessary infrastructure devices. The users will influence the infrastructure as they insert traffic upon the lines, but they should have no direct influence upon the infrastructure devices such as the routers and digital access cross-connects. The operators do have influence upon the infrastructure devices and must have an access path to those devices. It would be most appropriate if the users were not allowed to usurp the access paths of the operators. However, because the two are sometimes nearly indistinguishable, the task of separating the administrative channels from the user channels becomes difficult.

Figure 1: The Network Security Model



The following table describes the elements in this model.

Table 2: Network Model Elements

Element	Description	Example
Operators	The devices and people who operate, manage and support the network	Monitoring and Management Workstations, Syslog servers
Infrastructure	This composite area denotes the entire infrastructure. This is broken out to show the actual infrastructure devices as well as the supporting devices.	• Infrastructure Devices: Routers and Switches • Supporting Devices: DNS and DHCP servers • All other infrastructure components: wires, circuits, DSU/CSUs, SONET equipment, repeaters, etc.
Servers and Services	The devices that host content and services for the users	Web servers, file servers
Users	All of the users of the network and their workstations	Alice, Bob, Carol, Dan and their workstations
Arrows	Define which element influences or changes which other element	Users insert traffic into the network and thus influence the Servers and Services. Operators may also influence each of the components of the Infrastructure

Know Your Business

All well-running networks must have a *security policy* defined. This must reflect the goals of the network and must also be acceptable to the users and administrators. There are good examples of policies as well as methods that can be used to generate them. RFC 2196[2] contains several thoughts about constructing a policy and SANS[3] offers courses on this. While defining a network security policy, it will be advantageous to list the most likely disruptive influences to the network. This is commonly called *The Threat Model.* All potentially disruptive factors should be considered when forming the threat model, and they must be addressed when writing the security policy. It may, however, be beyond the capabilities of the operations staff to negate all of them. It may also be prohibitively expensive to try. In those cases, the writers of the policy should acknowledge the factors that won't be negated, but they should still find ways to minimize them. For example, in an Enterprise network the operators are somewhat likely to require access to routers and switches from any physical location in the network. In Service Provider networks, there may be less of a chance of that because the operators traditionally reside with the network management devices. In both cases, it would not be considered good for the network if a user could gain control of a router. The security policy for an Enterprise network may explain that network access to routers will be opened and available for any other device within the network. This will allow any operator to access the routers from any location. On the other hand, the security policy for a Service Provider network may state that access to the routers will be opened only for specific address ranges. Implementing this will prevent users, who reside within the address spaces assigned to the users, from accessing the infrastructure devices while allowing the operators, who reside within their own address space, to access the routers. In both cases, however, strong authentication will probably be required to additionally limit access to only authorized people.

Within the business of the network, operators must have the ability to control the infrastructure devices. Traditionally, the ways to interact with a device have been called "interfaces." A terminal with keyboard attached to the console port of a router is an interface just as is a Web browser accessing the router via the *Hypertext Transfer Protocol* (HTTP) through the network. Also, the path between the controlling device and the infrastructure device has traditionally been called a "channel." The wire connecting the terminal to the router is a channel just as is the TCP session that transports the HTTP in the prior example. The channels between the operators and the infrastructure devices must be secured, as well as the channels between the infrastructure devices. The first step in obtaining this goal is to identify all of the interfaces needed by the operations staff to access each of the remote devices. Along with this, they also need to identify each of the interfaces needed for the proper functioning of the network. The following lists are some of the possible network-available interfaces to some of the infrastructure devices in a dynamic network. This is somewhat broken down

into the interfaces needed by the operations staff, the interfaces usually needed by the other infrastructure devices, and some ancillary interfaces.

Table 3: Some Interfaces of Infrastructure Devices

Operations and Interfaces	Infrasturcture Interfaces	Ancillary Interfaces
telnet, Kerberized telnet, SSH, rsh, rcmd, rexec, HTTP, FTP, tftp, rcp, scp, SNMP, LDAP, COPS, Finger	Syslog, ICMP, DNS, DHCP, RIP, OSPF, BGP, IS-IS, IGRP, EIGRP, HSRP, NTP, SNMP, Multicast controls	RADIUS, TACACS+, Kerberos Authentication, PAP, CHAP, EAP, chargen, echo, time, discard, Auth (Ident)

Each of these interfaces may be exposed to the nefarious forces that are known to inhabit large networks, and each of these exposures has vulnerabilities that may be exploited. Telnet sessions may be hijacked, DNS queries may be answered by nonauthoritative and possibly maliciously incorrect responses, and sinister people can insert forged routing updates to confound and disrupt the network. The network security policy should expect that these vulnerabilities may be exploited and it should address the mechanisms that may be used to either negate the vulnerabilities or to minimize the exposures. In this model, the process may be used to limit the sphere and level of the influences. The policy may also make some attempt to identify the potential consequences of the disruption caused by the exploitation of these interfaces. It should also describe an escalation procedure for dealing with encountered problems.

Possibly, during the exercise of identifying the open interfaces in an existing network, some of them may be closed or removed if it is determined that they are not needed or if their function can be fulfilled by the use of another interface. As an example, consider a UNIX host that has both the *Secure Shell Protocol* (SSH) and *finger* services running on it. If the policy of the network is to tightly control the information that anyone can obtain from any device, then the operators may want to remove the *finger* service. The operators will be able to obtain similar information by running the *who* command on the UNIX system through an SSH remote execution request. On the other hand, if the operations processes have been built upon the format of the information returned by *finger*, then the operators may want to prevent direct access to *finger* from the network and require that it be run on the device or through the SSH request.

At some point, it would be a good idea to run a scanner against the infrastructure devices. The *Network Mapper* (NMAP)[4] is a freely available tool that can pick out some of the active interfaces of a device. This, or a similar tool, should be periodically used by the operations staff to ensure that the open ports of an infrastructure device are those that are known to be open. This investigation should not be limited to operations channels, but should also include application channels. For example, the question should be asked if the operations workstations should have open application interfaces—such as *Simple Mail Transfer Protocol* (SMTP) or *Network File System* (NFS). There are exploitable

vulnerabilities associated with some application interfaces that should be addressed in the security policy. In most cases, it would be prudent to remove applications that are not needed from infrastructure devices and supporting servers, as well as from operations devices when they are not needed. In all cases, it is usually considered to be a good practice to review the entries in the *inetd* configuration in UNIX systems.

It should be remembered that there will almost certainly be an access path between the users and the network interfaces of the infrastructure devices. The network security model diagram shows that neither the users nor the servers should have any direct influence to change or control the infrastructure devices. This is somewhat analogous to the policy of giving privileges on a multiuser system. In most well-run multiuser computing systems, the operators give only the most meager of privileges to the users of the system. This prevents most accidental and malicious disruptions. If the users need to run a privileged process or to access the files of other users, processes that utilize setuid are used or consensual groups are established. Generally, efforts are made to prevent users from having significant privileges on these machines. The alternative of giving each user high-level privileges usually results in disaster after a short time because the users then have the ability to overwrite or delete files, and may run processes that are generally disruptive to the operating system and to others.

Similarly, giving users high-level access to the routers of a network would have a deleterious effect. In the case of *Quality of Service* (QoS), a user given the privileges to reconfigure routers along a path would be able to provide his/her own designated flows with bandwidth and priority assurances. Subsequent users would also have that capability, and their modifications may leave the first user without his/her expected QoS—and possibly without a session at all. A far better mechanism to fairly deploy QoS is through the use of a brokering service. In a "policy network," users or authenticated processes may request a level of service for their flows through a *Policy Manager*. This Policy Manager should have the capability to arbitrate requests to provide a semblance of fairness. The Policy Manager would then directly control the appropriate routers within the rules established by the administrators.

Along these lines, conveying security-related policy to infrastructure devices should take a similar path. For example, if the network security policy states that user access to a particularly sensitive network resource must be authenticated and controlled, the operators may elect to place a firewall between the users and that resource. That firewall would be classified as an infrastructure device and users should not directly access or control it. Rather, the users may authenticate themselves to an authentication service, which would notify the firewall that their access to the resource is permitted or denied. The authentication service may also send a set of restrictions for the access method; it may permit HTTP access but deny Telnet and *File Transfer Protocol* (FTP) for one person, but for another it may permit only Telnet.

The reasons for authentication, authorization, and access control must be described in the network security policy. It would be simple to mandate strict controls at many places in the network. However, that may not meet the needs of the business or the tolerance of the users. More to the point in this article is the requirement in the model that the disruptive influences be negated or minimized. Having a firewall or other access control device silently discard disruptive packets may be preferable to having a user or unconstrained process continue to spew garbage around the network.

Decide on the Methods of Securing the Channels and Interfaces

Some of the very first computing devices were designed to be managed locally and not remotely. Consoles consisting of a teletype device and a roll of paper were among the first interfaces to modern computing devices. Various methods were devised to extend these administrative interfaces beyond the confines of the frigid "Computer Room." The first efforts were to keep these interfaces out of band, a scenario that meant separate wires from the physical port on the machine to a console in the operations room. In many cases, the wires from the remote terminals to the system were still visible because they were laid along the floor and could, therefore, be considered a secure channel. While this maintained a secure administrative channel—or path—that could not be tapped or exploited by others, it didn't scale as more and more computing and ancillary devices were placed into the computer room, each requiring its own console. When remote terminals became commonplace, administrative functions were allowed over that channel. In almost all cases, the operating systems were mature enough to require some form of authentication before critical management operations were allowed.

The out-of-band channels for secure remote administration of devices may no longer be applicable to large networks. There are costs associated with running separate secure networks for the sole purpose of out-of-band operations, and there is the impracticality of one-at-a-time access through the console port of each device. This applies equally to the practice of placing a modem on the console ports of devices—a deployment that is not considered secure because there are still many automated dialers looking for answering modems. For these reasons, in-band access of operations has become the preferred method for modern networks. Telnet has been the oldest remote channel—and interface—for remote operations. Since then, other remote interfaces have been opened for controlling, commanding, and operating devices.

Many attempts have been made to "secure" Telnet and its use as a command and control channel. These efforts address the vulnerabilities of the protocol, and some address the interface itself. The *Berkeley Software Distribution* (BSD) "r" command set, such as *rlogin*, *rsh*, *rexec*, and others, were meant to be a substitute for the most common uses of Telnet within a trusted environment. It was assumed that the person initiating the command had previously been successfully authenticated.

SSH was meant to be a secure replacement of the Berkeley "r" tools. The SSH console session has been widely deployed to remotely operate devices. This replicated the Telnet interface while replacing the channel. The protocol addressed machine authentication, user authentication, and session confidentiality and integrity. When used as it was intended, it can effectively replace Telnet as a secure interface and channel. The *scp* feature of SSH can also securely replace *rcp,* and it has been used as a replacement for FTP. Likewise, a Kerberized Telnet and Kerberized FTP have been released to do the same.

Several other efforts have also been undertaken to secure some of the other administrative interfaces and channels. For example, the security issues of *Simple Network Management Protocol* (SNMP) are being addressed with the options of SNMPv3[5]. Also, applications that utilize HTTP can be secured with HTTP over SSL (HTTPS) (*Secure Sockets Layer/Transport Layer Security* [SSL/TLS])[6]. At this time, it appears that SSL/TLS is emerging as a mechanism that can be utilized to provide some security to many different applications. Beyond the operational interfaces and channels, work has been done to secure some of the infrastructure and ancillary interfaces. Some routing protocols have built-in authentication and integrity through the use of signing the routing updates with a shared key. Each mechanism that has been secured has been the subject of a focused effort to address that specific interface and channel. However, unlike those named above, some channels, such as Syslog and *Trivial File Transfer Protocol* (TFTP), have not been explicitly secured at this time.

IP Security (IPSec)[7] was developed as a general-purpose mechanism that may be used to provide a secure wrapper around any unicast flow. Its cryptographic mechanisms can provide strong authentication, confidentiality, and integrity. While IPSec can be used to secure any flow, it may require additional infrastructure. A *Public Key Infrastructure* (PKI) must be established within the network. The alternative is to use preshared keys, a solution that is operationally intensive and doesn't scale well. IPSec also requires consistent time synchronization between the devices, as well as a consistent DNS. If these pieces are in place, the operations staff can utilize IPSec to secure each of the needed operations channels. If the operators and administrators choose this method, then they should ensure that the unsecured channels are unavailable to anyone but themselves. For example, if the Telnet channel is secured with IPSec, then the Telnet port on remote devices should be closed for inbound access.

One method of closing the exposures is through *Access Control Lists*. Routers and switches usually have mechanisms that can be used to allow inbound and outbound sessions from only certain devices. UNIX devices usually have the ability to run TCP wrappers that can provide access-control mechanisms for inbound and outbound sessions. If infrastructure devices can be grouped together, the operators may decide to

place them behind an internal firewall. The decision to do that should be thought through. Generally the internal firewall will limit access of the protected devices to the specified interfaces[8]. If this is done for a group of network management stations, the net effect may be that any attempts to access those workstations from outside of the firewall would be denied. The only inbound flows may be SNMP responses and traps. This implementation would limit the operations staff to being physically present before they could operate those devices. On the other hand, the firewall would prevent users from mistakenly or intentionally forming sessions with those devices. Because any received packet would have to be assessed by the device, a firewall that would discard packets before they are received by the device would help to prevent denial-of-service attacks. The use of internal firewalls should not be used as an excuse for poor security measures on the protected devices. Regardless of how effective the operators feel their firewall is, the protected devices must be treated as if they were otherwise exposed.

In determining the channels that will be used for the administration of the infrastructure devices, the packages will also be selected. At this time, many devices are sporting Telnet, FTP, and HTTP channels and the operators may utilize workstations that have these packages already loaded onto them. Also, networks comprising Microsoft NT servers may be managed remotely by the NT administrative tools, which commonly run on NetBIOS over TCP/IP (NBT). When given the choice, most often the operations staff will select easy-to-use and commonly available packages to access the interfaces of the infrastructure devices for remote operations and control. In all cases, these will be packages that will be available to the user community of the network as well. The users of the network may also easily download packages of these types if they don't already have them on their machines. For example, the operators may choose to utilize SSH for secured access to some devices. It is a trivial task for the users to also download an SSH client package and to start poking around the network to see what they can find. Even SNMP packages can be easily downloaded to the workstations of the users.

The operators and administrators must avoid the temptation to select a less-well-known package for infrastructure management based upon the thought that the users probably won't know about it. Users may not be initially aware that some packages are being used, but they can also download sniffer packages. Given enough time, even passive sniffing will give them enough clues to determine the channels used for administration. When they know that, they can then probably download the package themselves, and may then attempt to use it to explore the network. It should also be noted that the more heavily used packages have been scrutinized much more than the newer or less used packages. As a very general rule, the older a package gets, the more it becomes trusted because more people have been using it and *probably* attempting to break it.

As described above, and as it is seen in the diagram of the model, some of the channels that are available to the operators are also available to the users. This means that if the operators utilize Telnet to control their routers, it may be possible for a user to also initiate a Telnet session to a router. There must be an extremely strong discriminator to differentiate between the authorized operators and the unauthorized users before access to control the device is granted. Almost exclusively, the discriminator used is some form of authentication. An operator should be able to satisfy an authorization challenge, whereas an unauthorized user should not. A username and password is the most common form of in-band authentication. Specifically within Telnet and FTP, an in-stream challenge is presented to the user attempting a session; the user is asked for a username and then for a password. If these credentials match the values stored on the host, then the session is permitted. In these sessions, the credentials are exposed to casual observation. Anyone with a packet-sniffing device will be able to plainly see the username and password. These credentials must be regarded as secrets that must be protected. If they are compromised or stolen, then the operators have lost their control of their network. Some packages, such as SSH and Kerberos, have addressed these problems and have found ways to prevent secrets from being passed during authentication.

It must also be noted that some infrastructure devices do not offer any in-band channels for control. Many *Channel Service Units/Data Service Units* (CSU/DSUs) are not IP aware and do not offer any in-band channels for control. In cases like those, physical access may be the discriminator that prevents unauthorized users from controlling the device. Typically, a lock on a door or a cabinet would be the "challenge," and the key would be the authentication credential, which must be treated like a secret. It cannot be emphasized enough that these secrets must be protected. The *CERT Coordination Center* has written a very broad Tech Tip, which explores the topic of password security[9]. Many companies have found it very beneficial to periodically hold training courses to highlight the importance of this subject both to their operators and to their users.

Ancillary Channels Also Require Security

One of the parallel problems with using authentication credentials is its distribution. Many devices are capable of maintaining a local database of usernames and passwords. However, maintaining identical databases on each device throughout large networks is infeasible. More often, the authentication credentials are stored in a centralized database and an *Authentication, Authorization, and Accounting* (AAA) protocol is used to transfer them as needed. The AAA protocols most often used are *Remote Access Dial-In User Service* (RADIUS), TACACS+, and *Kerberos* authentication. Each of these has different characteristics and security mechanisms. Kerberos authentication was designed to securely transport authentication material. A password is never transferred across the network in this architecture. This protocol has withstood the test of

time, but it has been difficult to establish in networks that aren't committed to maintaining it. This situation seems to be changing because more "productized" versions are becoming available on the market. TACACS+ has a mechanism to hide the exchanges between the TACACS+ client and the server. It is also capable of transferring authorization rules for each user. RADIUS uses a mechanism to hide portions of the exchange between the RADIUS client and server as well.

Beyond this, the channels for telemetry, audit, and accounting may need to be secured. There are no inherent mechanisms to secure syslog at this time, and SNMPv1 may be protected with a Community String, but that solution is considered weak. It is possible to allow read-only access to the SNMP interface, but SNMPv3 has many of the security features that have been requested to secure this protocol. Other channels that are required by the operations staff should also be critically reviewed because many forms of attacks are on open channels.

It would be appropriate for the operations staff to keep up with new exploits and to assume that the users of the network have access to the latest "hacker" tools. It is quite common for people to hear about an exploit or published vulnerability and then "try it out" in the nearest available network. For this reason, it should be in the security policy of the network that "security patches" be given the highest priority and should be loaded on the affected platforms as soon as they are available and have been approved for the environment.

Conclusions

When any security mechanism is applied, the appropriateness and applicability of the solution should be questioned. On the surface, some security solutions may appear to be good; however, their applicability to the situation must be verified. As an example, SSL may be used to secure HTTP traffic, and it is commonly found in many Web browsers. Unfortunately, not many people explore the browser options that are enabled by default. In most browsers, SSLv2 is still available, even though it has published and exploitable vulnerabilities. Additionally, even in SSLv3—which negates the vulnerabilities of SSLv2—low key-length cipher suits are still available and enabled by default. In many cases, a null-cipher crytpo algorithm is available. In the internal networks of many companies, SSL may be selected and implemented using a self-signed certificate. Care must be taken to ensure that this certificate is the one distributed to each administrative workstation. SSL sessions may be formed without certificates supplied by either endpoint. An attacker could exploit this through a man-in-the-middle attack. Another example would be the use of SSH. SSHv1 has known vulnerabilities. If the administrators decide to deploy SSH for the control of the remote infrastructure devices, they should first decide if they should be worried about attacks against those known vulnerabilities in their infrastructure. If they are, then they should either deploy SSHv2, which addresses the vulnerabilities of SSHv1, or they should explore the use of Telnet with IPSec.

In many cases, rather than using the "most secure" solution, perhaps a simpler solution would still provide adequate protection. The "most secure" solution—the one that mitigates all perceived threats—is usually too costly to implement. In many cases, network operators and administrators with many years of experience have decided that SSHv1 is adequate for their needs and they can mitigate or minimize the exposure. In other cases, some operators are turning to SSHv2 or IPSec to cover the vulnerabilities that have been found in SSHv1. In some cases, the use of SNMPv1 may also be acceptable as long as its exposures are understood and the operators determine that its use will not pose a problem.

Excessive "security" may also intolerably reduce the usability of the network. It is important to remember that the network is there for the users. Placing security restrictions upon them to keep them out of the infrastructure is like keeping the doors locked to the building boiler room. Untrained people entering that area may hurt themselves or they may cause serious problems to others. If they have malicious intent, they could damage the machinery. Excessive security for that analogy would be similar to locking the boiler room, locking the ingress and egress points to the building, and mandating that armed guards accompany anyone that is permitted to enter the building. In some cases, that may be appropriate for the perceived threat. However, in the case that this applies to an elementary school building, it is inappropriate and would make some parents think of moving their children to other schools.

The model described in this article may be used as a thought process to review an entire network at a high layer to see the relationships between the various devices. It may also be used to design the security policy and the acceptable use policy of the network. Another use for it may be to define the operational procedures for the operators to securely administer the network and to define how the infrastructure devices will communicate. However it is used, some settlements must be made between the desire to provide security and the usefulness of the network. The cost of the security mechanisms cannot be unreasonably high, and the mechanisms cannot change the business model of the company. The enforcement of the policy must be effective, yet above all it must not change the expectations of the users. In all cases, the administrators and operators must find some balance between their need to secure the infrastructure and the need for the users to have the ability to actually use their network.

References

[1] Internet Engineering Task Force DNS Extensions Working Group, last updated July 2000,
`http://www.ietf.org/html.charters/dnsext-charter.html`

[2] Fraser, B., "Site Security Handbook," RFC 2196, September 1997.

[3] System Administration, Networking, and Security Institute,
`http://www.sans.org/`

[4] Fyodor `<fyodor@dhp.com>`, "NMAP—The Network Mapper,"
`http://www.insecure.org/nmap/index.html`

[5] Stallings, William, "Security Comes to SNMP: The New SNMPv3 Proposed Internet Standards," *The Internet Protocol Journal,* Vol. 1, No. 3, December 1998.

[6] Stallings, William, "SSL: Foundation for Web Security," *The Internet Protocol Journal,* Vol. 1, No. 1, June 1998.

[7] Stallings, William, "IP Security," *The Internet Protocol Journal,* Vol. 3, No. 1, March 2000.

[8] Avolio, Fred, "Firewalls and Internet Security," *The Internet Protocol Journal,* Vol. 2, No. 2, June 1999.

[9] CERT® Coordination Center, Tech Tips, "Protecting Yourself from Password File Attacks," Last revised February 12, 1999.

CHRIS LONVICK holds a Bachelor of Science degree from Christian Brothers College and is in the Consulting Engineering Department of Cisco Systems in Austin, Texas. He is currently the chair of the IETF Syslog Working Group. Chris can be reached at `clonvick@cisco.com`

Book Reviews

Multiwave Optical Networks

Multiwavelength Optical Networks: A Layered Approach, by Thomas E. Stern and Krishna Bala, ISBN 020130967X, Addison-Wesley, 1999.

Initial Impressions

This book attempts to fit into two camps; one, an overview of the potential choices that could be offered in wavelength-division multiplexing, or WDM, and the other, an academic text. Because of its scope, the treatment is uneven.

Organization

The first four chapters lay the groundwork. Chapter 1 starts by defining terms and positing why WDM is an enabling technology. The authors believe that the driving application will be LAN interconnection, ostensibly in metro areas. It is worthwhile noting that the authors make no claims about this text relating to an all-optical network. They simply expose the choices available to manipulate the various wavelengths, or lambda. The current methods for performing lambda manipulation are still bound in the electrical domain.

Chapter 2 covers the hierarchy or layering present in a WDM environment and some of the choices for configuration at each point in the hierarchy. The authors spend some time on the concepts of spectrum partitioning and what routing and switching in this domain means. A key point raised relates to the concept of wavelength conversion at network access points. The chapter closes with a brief review of some types of logical overlays that may sit on top of a WDM network. Three types are examined, ATM, *Synchronous Optical Network* (SONET), and IP networks.

The third chapter covers how network interconnection may occur and how the management and control features may be implemented. Four basic topologies are described, each with its salient features highlighted. These topologies include shared channel networks; wavelength routed networks, linear lightwave networks, and hybrid, logically routed networks. It is interesting to note that many commercial implementations, especially from traditional telecom providers, tend to follow the simpler topologies, while we are beginning to see newer telecom providers utilizing the more robust topologies.

Chapter 4 discusses what the authors consider enabling technology. To a large degree, these enabling technologies are the basic components of an optical system, for example, fibers, amplifiers, transmitters, and receivers. Crosstalk is mentioned in particular. The authors then delve into photonic device technologies and wavelength converters, and then they close with some simulation work on end-to-end transmission paths.

Chapters 5, 6, and 7 discuss in depth the ramifications of each of the four techniques. What is fairly intriguing here is that the authors have extensive bibliographies at the end of each chapter, and they include a series of problems that are left as an exercise to the reader.

The eighth chapter touches on the concepts involved with survivability and restoration of service. This chapter should help the practical network engineer in understanding most of the possible failure modes. In the last chapter, the authors look at current trends, and they try to predict business drivers for WDM deployment. Once again, they show their true colors as academics when they close with a statement on the importance of testbeds.

On to the Appendices! I am grateful to the authors for including some basic material on graph theory, scheduling algorithms, Markov chains and queuing, some work on minimal interference routing in the optical domain and, finally, close with a synopsis of the SONET standard.

Good Reference

Overall, there is a fair amount of practical material here, but it is tucked into large amounts of academic detail. I'm not sure this volume would work as a standalone textbook, but it clearly is a good reference for the state of optical networks in the last years of the 20th century.

—Bill Manning,
University of Southern California
Information Sciences Institute
manning@isi.edu

Net Slaves

Net Slaves: True Tales of Working the Web, Bill Lessard and Steve Baldwin, ISBN 0-07-135243-0, McGraw-Hill, 2000.

How can you not want to read a book that opens with a quote from a Guns&Roses song, "Do you know where you are? You're in the jungle, baby!"? *Net Slaves* is about the people who maintain the jungle that big game hunters come to exploit. The same jungle marketed as the digital age and the e-generation. This is the land of the "dot-coms" and future big-buck IPOs. Has hubris masked your role in this jungle? *Net Slaves* will set you straight. Exactly who are these net slaves? Well, take the 15 question quiz provided by the authors and determine your Internet exploitation quotient. Don't be shocked to find yourself among the new media caste; the only question is, what part of the jungle are you assigned to clean after?

The authors spent a year interviewing people who work for Internet-based companies. Based on their findings, they created 11 character composites: Garbagemen; Cops or Streetwalkers; Social Workers; Cab Drivers; Cowboys or Card Sharks; Fry Cooks; Gold Diggers or Gigolos; Priests or Madmen; Robots; Robber Barons; and Mole People.

For each composite the authors cite someone's real-life work experience—of course, in order to protect the innocent (and the guilty), names have been altered.

I was annoyed with David Zorn, Card Shark; his type does nothing but give the industry a bad reputation. The story of Ken Hussein, Robot, both saddened and angered me. I truly hope he and his family are doing better. How can anyone not feel sorry for Kellner after being taken in by Gigolo Mira? Jane, Cab Driver, learned the hard way that you have to roll with the blows to survive in the jungle. Finally, I must confess, I found the most disturbing of all profiles to be of Outis, a Mole Person.

For each profile the authors provide some social-economic statistics. How old is the average Social Worker? How much does it cost to hire a Cowboy? What are the career aspirations of the average Cab Driver? How do you know if a Robot is annoyed with you? You're a Garbageman; what are your chances of upward mobility? A lot of this is funny, but to leave it at that would be missing the point entirely. Every composite represents scores of real people's lives, and how they live doesn't necessarily match up with the glamour often associated with the high-tech industry.

My favorite profile is of Jason Barstow, a Madman. Barstow arrives on the scene on his Harley, ready to participate in a two-day seminar put on by the Earth Business Network. A former chicken farmer and former guitar player, Barstow now finds himself lecturing to a room full of CEOs. He begins by telling them about the 5 milligrams of LSD he bought the previous night, and proceeds to plant seeds of anxiety—did he spike their morning juice? As Barstow delivers his lecture on the future of e-commerce and builds to the climax, a frustrated Slim Clarkston of NetScathe blurts out, "Mr. Barstow, I want you to tell us the truth about your little prank." With the lecture over, Barstow returns his pass to the security desk. "How did it go?" asks the security guard. "Same bull," Barstow responds, "but they never seem to get tired of it."

Are these stories true? I don't know—it doesn't matter! What are true are the composites. This book is funny. It is also humbling. Most important, it is true. It was fun to read. After each chapter, I found myself wearing an undeniable mischievous grin as I scanned the office looking for the person I just read about; this is all in good fun as long as I remember one important thing: I'm in the book—and you are too. In my experiences, I've found that a certain animosity always exists between people who work call centers, programmers, Web designers, managers, and the like. *Net Slaves* reminds us that we are all in this jungle together.

—Neophytos Iacovou, eBenX Inc
diacovou@ebenx.com

Implementing IPSec

Implementing IPSec: Making Security work on VPNs, Internets, and Extranets, Elizabeth Kaufman and Andrew Newman, ISBN 0-471-34467-2 Wiley Computers Publishing, 1999.

Organization

The book is organized into four parts. The first three chapters of Part One should be nothing more than review for anyone who has been in networking for even a short time. Chapter 4, "Encrypting within the Law," analyzes current worldwide regulatory trends for encryption technologies and examines how existing laws will impact your ability to legally purchase and install IPSec products. Included is some good information that may help keep you on the right side of the laws pertaining to encryption. Encryption is an area of potential problems, especially when you are running your network between countries.

Part Two is a primer on the basic technological components of IPSec. Chapter 5, "A Functional Overview of IPv4," and its basic design characteristics should be old news to anyone who is seriously thinking of running any type of encryption on his/her network. Chapter Six is an overview of cryptographic technologies. Chapter 7 "The Basics of IPSec and Public Key Infrastructures (PKIs) Fundamental to Current IPSec Standards," has some good information pertaining to IPSec and its different components, but leaves out an explanation of its two basic modes of operation: *transport* and *tunnel*.

Part Three analyzes how and why the IPSec protocols can break existing IP networks, and should provide the reader with some good information. Chapter 8, "What Won't Work with IPSec," describes the root cause of IPsec performance problems and protocol conflicts. Chapter 9, "IPSec and PKI Rollout Considerations," discusses gateway-to-gateway, end host-to-gateway, and end host-to-end host configuration options and explains some of the policy elements of PKI.

Part Four provides some criteria for evaluating vendors and products; this information would be of little interest if you are unfamiliar with writing an RFI. Also included is some reference material, including an appendix, with a complete copy of the IPSec RFC (2401), "Security Architecture for the Internet Protocol." A glossary, which does *not* offer a description of IPSec, is included as well.

Who Should Read This Book

By trying to appeal to the technical as well as the nontechnical reader, the book has missed both. There are areas that will appeal to the reader with a limited networking background, as well as areas for the more technical. However, if you are the type of reader inclined to read the RFCs, you will find very little reason to read the remainder of the book. Overall the book does not provide enough information for any one group. Inclusion of RFC 2401 seems unnecessary considering how easily RFCs can be obtained from the Internet.

—Al Pruitt, CSG Systems, Inc
`al_pruitt@csgsystems.com`

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 carries tutorial articles ("What is…?"), as well as implementation/operation articles ("How to…"). It provides readers with technology and standardization updates for all levels of the protocol stack and serves 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, trouble-shooting, 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**

Scott Bradner Receives Postel Service Award

The Internet Society (ISOC) recently announced that noted Internet standards leader and Internet pioneer Scott O. Bradner has been awarded the prestigious *Jonathan B. Postel Service Award* for 2000. In presenting the award, Geoff Huston, Chair of ISOC, said, "Scott Bradner was introduced to many of us with his accurate and careful measurements of router performance. He has been a long standing participant in the *Internet Engineering Task Force* (IETF), and continues to serve on the *Internet Engineering Steering Group* (IESG) as the Area Director for Transport. He was a ISOC Trustee for six years from 1993 until 1999 and continues to serve as the Society's Vice-President for Standards. This is an impressive set of contributions and is worthy of recognition in Jon Postel's name as the 2000 recipient of the Jonathan B. Postel Service Award."

Don Heath, president and CEO of ISOC, said, "We established the award to honor the late Jon Postel by recognizing his unselfish and substantial contributions to the Internet over a 25 year period." He added, "Scott Bradner exemplifies the spirit of all that Jon brought to the Internet community and his outstanding contributions have made this year's choice an easy one. Scott's careful judgment and good humor has been a major contribution to many of the ISOC's activities, and we are pleased to be able to recognize his contributions in this unique fashion."

Bradner has been an active contributor to the IETF for over a decade, and has served as a Working Group Chair, the Area Director for Operations and currently serves as the Area Director for Transport. He also was the Director of the IPv6 area, and oversaw the process of refinement of a number of proposals into the definition of a coherent architecture for IPv6. Bradner has been the prime author of the current Internet Standards Process documents. He has also been an instructor at ISOC's *Network Training Workshops for Developing Countries* for many years, and has been a catalyst for the development of operationally robust Internet services in many areas of the world.

The Award is named for Dr. Jonathan B. Postel, an Internet pioneer and head of the organization that administered and assigned Internet names, protocol parameters, and Internet Protocol (IP) addresses. He was the primary architect behind what has become the *Internet Corporation for Assigned Names and Numbers* (ICANN), the successor organization to his work. The Award is presented at the Internet Society's annual INET Conference. It consists of an engraved crystal globe and US $20,000.00. Scott Bradner becomes the second recipient of the award. The first was presented posthumously to Dr. Postel in 1999.

The Internet Society is a non-profit, non-governmental, open membership organization whose worldwide individual and organization members make up a veritable "who's who" of the Internet industry. It provides leadership in technical and operational standards, policy issues, and education. ISOC hosts two annual Internet conferences, trains people from all over the world in networking technologies, conducts workshops for educators, and publishes an award-winning magazine, *OnTheInternet*. ISOC provides an international forum to address the most important economic, political, social, ethical and legal initiatives influencing the evolution of the Internet. This includes facilitating discussions on key policy decisions such as taxation, copyright protection, privacy and confidentiality, and initiatives towards self-governance of the Internet. ISOC created the Internet Societal Task Force as an ongoing forum for discussion, debate, and development of position papers, white papers, and statements on Internet related societal issues.

ISOC is the organizational home of the IETF, the Internet Architecture Board, the IESG, and the Internet Research Task Force—the standards setting and research arms of the Internet community. These organizations operate in an environment of bottom-up consensus building made possible through the participation of thousands of people from throughout the world. For more information, see **http://www.isoc.org/**

APNIC Policy Meeting

The *Asia Pacific Network Information Centre* (APNIC) will host an Open Policy Meeting October 25–27, 2000 in Brisbane, Australia. The meeting is open to anyone with an interest in Internet addressing issues. For more information see: **http://apnic.org**

APRICOT 2001

The *Asia Pacific Regional Internet Conference on Operational Technologies* (APRICOT) will be held in Kuala Lumpur, Malaysia, February 26 to March 2, 2001. APRICOT is a forum that facilitates knowledge sharing among key Internet builders in the region, with peers and leaders from the Internet community worldwide. Since 1996, APRICOT has established itself as Asia Pacific's premier regional Internet Summit where related organisations converge and host their annual general meetings and other special events. The week-long summit comprises seminars, workshops, tutorials, conference sessions, Birds of a Feather (BOFs), and other forums, all geared towards spreading and sharing the knowledge required to operate the Internet within the Asia Pacific region. For more information see: **http://www.apricot2001.net**

The Internet Protocol Journal

Ole J. Jacobsen, Editor and Publisher

The Internet Protocol Journal is published quarterly by the Chief Technology Office, Cisco Systems, Inc.
www.cisco.com
Tel: +1 408 526-4000
E-mail: ipj@cisco.com

The Internet Protocol Journal, Cisco Systems
170 West Tasman Drive, M/S SJ-10/5
San Jose, CA 95134-1706
USA

ADDRESS SERVICE REQUESTED

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