TCP CONGESTION CONTROL IN SHARED SATELLITE ENVIRONMENTS

Keith Scott, Patrick Feighery, and Brian Crow
The MITRE Corporation
Reston, VA

and

Mark Jurik
U.S. Army Information Systems Engineering Command
Ft. Huachuca, AZ

ABSTRACT

This paper describes the use of a transparent TCP gateway to improve the performance of applications operating in a shared secure satellite environment. Typically, a satellite gateway is installed at each end of the satellite link, and the gateways process all traffic traversing the link. With the proliferation of virtual private network (VPN) technologies, multiple encrypted tunnels can be established over a satellite link. If transparent TCP gateways are used to optimize application performance, a pair of gateways are required at the egress points of each tunnel. Many TCP gateways perform poorly in this situation because they do not implement congestion control on their 'satellite' sides, instead relying mainly on rate-control to send data at or near the bandwidth capacity of the satellite link

Our approach is to use Space Communications Protocol Standards (SCPS) transparent transport layer gateways, which are capable of implementing a variety of congestion control schemes on their terrestrial and satellite sides. By using a variant of the TCP Vegas congestion control algorithm, the gateways can communicate indirectly (by detecting changes in packet round-trip times) to efficiently share the satellite bandwidth. Results show that this improves performance over end-to-end TCP without congesting the network between the gateways and the uplink as pure rate-control would do.

INTRODUCTION

A. TCP In Satellite Environments

Transmission Control Protocol (TCP) [1] is the Internet transport protocol that supports many common applications, including e-mail, ftp, http, and telnet. TCP provides reliable, in-order delivery of data from sending to receiving applications, without duplications. That is, TCP guarantees that all data sent will arrive at the destination, the data will be delivered to the destination application in the order in which it was sent, and that no duplicate data will be delivered to the receiver. TCP achieves reliability by having TCP receivers acknowledge data sent to them. If a TCP sender does not receive an acknowledgement for a particular piece of data, it will retransmit it until the data gets to the receiver.

In addition to the services described above, modern TCPs are also required to implement congestion control mechanisms [2, 3]. Congestion control prevents senders from flooding the network with data to the point that intermediate routers have no buffers left and have to discard data, which TCP will then have to retransmit. For example, when a TCP session begins, the sender first sends one packet to the destination and waits for a response. TCP then sends two packets before waiting for a response, and so on. This exponential growth phase is called *slow-start*, and its purpose is to allow TCP to quickly determine the available bandwidth on the path to the destination. At some point, the sending TCP reaches the flow control limit imposed by the receiver or it causes a router buffer to overflow, resulting in data loss. If the flow control limit imposed by the receiver is reached, transmission continues at that limit until a loss occurs. When a loss occurs, the receiving TCP will indicate it to the sender, who then retransmits the lost segment(s). In addition to retransmitting the lost segments, the sending TCP cuts its transmission rate in half, after which it increases it again linearly, at the rate of one extra TCP segment per round trip. Thus TCP's transmission rate grows linearly in this *congestion avoidance* phase, as opposed to exponentially as in the slow-start phase. It is worth noting here that the description above is for the Reno implementation of TCP/IP, which is the most widely used and implemented version of TCP/IP. The congestion control algorithm is known as Van Jacobson congestion control.

Unfortunately, TCP Reno does not perform well on high-delay, lossy satellite links. The default TCP window size on most common operating systems varies between 8 and 32 kilobytes (KB). Ideally the window size should be set to the bandwidth*delay product (BDP) of the channel, where the delay is the round-trip time. For geosynchronous satellite links, the round trip time (including processing time) can exceed 600 milliseconds. Depending on the bandwidth of the channel, it is common for terminals to become window limited (BDP exceeds configured TCP window size), resulting in a sending terminal having to stop transmission until acknowledgements (ACKs) are received so the sliding window can be advanced. Thus the sending terminal operates in a "stop and go" manner, which results in an under utilized transmission link.

If a datagram is corrupt, TCP assumes the link is congested, retransmits the lost datagram, and backs off its transmission rate. Unfortunately, reducing the transmission rate on an uncongested satellite link results in degraded performance. The congestion avoidance algorithm allows the sending terminal to grow linearly to its pre-loss data rate. Unfortunately, the rate of growth is paced by the ACK traffic, which must traverse a high delay satellite link before reaching the sender. Thus the recovery rate for satellite links is much slower than terrestrial links, resulting in reduced throughput.

A number of TCP options have recently been developed and widely deployed to increase performance over long fat networks [4, 5], and research in the area of satellite networks is ongoing [6].

B. Split-TCP Gateways

Transport layer gateways can improve TCP performance across stressed environments by breaking the end-to-end TCP connection into multiple transport layer connections. Figure 1 shows an illustration of this concept, where the transport connections are split into three pieces. In Figure 1, each gateway translates between two transport layer protocols, using one to communicate with end hosts and another to communicate with the peer gateway. To see how this can improve performance, recall that the receiver's receive window needs to be at least the BDP of the network in order for the sender to fully utilize the network resources. Since most end host TCPs have their window sizes set for the BDP of terrestrial connections, they can significantly under-utilize the communications resources when communicating through a geosynchronous satellite. A pair of TCP gateways at the ground stations could terminate the terrestrial communications and implement the large windows TCP option [4] to increase the window size over the satellite hop.

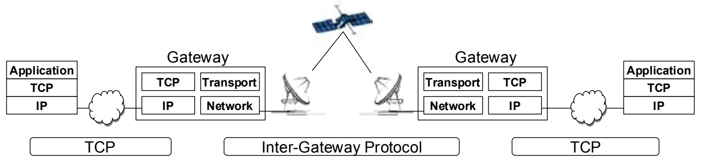

Figure 1. A Split-TCP Connection over a Satellite Channel

Most inter-gateway protocols are a modified TCP stack running on top of IP at the network layer. Typical gateway implementations use a pure rate-control strategy to keep the satellite channel full. Pure rate-control introduces traffic onto the satellite link at a specified rate (commonly the rate of the satellite link). The BDP on the satellite side of the gateways is tuned for the satellite link. It is also assumed that a single pair of gateways at either end of the satellite link will process all the traffic. Generally, all loss is assumed due to corruption and not congestion, therefore gateways will not back off their rate of transmission when loss is encountered. Gateways are usually designed to be transparent to the host systems. Normal default window sizes, etc., on the host systems are sufficient for proper gateway functioning. For networks where a single pair of gateways can process all traffic at the egress points of the satellite link, current gateway implementations will provide significant TCP throughput enhancement over a standard end-to-end TCP connection.

THE PROBLEM

Satellite gateways may not work well when they don't process all traffic entering and leaving a satellite link. One example is when traffic is segregated using virtual private networks (VPNs). VPNs are typically an incorporation of tunneling, authentication, and encryption technologies. TCP gateways will interoperate with VPN devices, but they must be installed on the link prior to entering an encrypted tunnel. There are two reasons for this:

- The gateways must see the original IP and transport headers in order to terminate a TCP connection with a local host.
- The devices installed on the tunnel-side of the VPN device are normally prevented from communicating with devices on the non-tunneled side of the VPN device.

Satellite Internet service providers and very small aperture terminal (VSAT) IP data service providers have developed a number of gateway architectures and inter-gateway protocols. Many of the inter-gateway protocols rely heavily on pure rate control to send data across the satellite hop; they do not implement congestion control on their 'satellite' sides. This can pose difficulties if multiple independent gateways are connected to a single satellite uplink ground station, say through a common router. In these cases either:

1. The bandwidth of the satellite link can be partitioned *a priori* between the various gateways so that even if all of them are transmitting at their full rates, they do not overrun the uplink
2. The various gateways need to communicate amongst one another to coordinate their transmission rates so as not to overrun the uplink
3. If each gateway pair is configured to rate-control traffic onto the channel at the full bandwidth of the satellite link, network congestion will likely occur if more than one gateway pair is simultaneously active.

The first option does not allow for the most efficient sharing of the scarce satellite resource. Under this option, a single source would be unable to use more bandwidth than was allocated to its gateway, even if other users did not need the bandwidth. The second option is often infeasible from a policy standpoint, as it would require coordination between gateways in different privacy areas. The third option, where each gateway sets its maximum transmission rate to that of the satellite link, allows a single user to take the entire bandwidth when needed, but makes no provision for effective sharing of that bandwidth when there is contention. Specifically, if multiple users transmit at the same time, data may be dropped at the satellite uplink due to buffer overflows (see Figure 2).

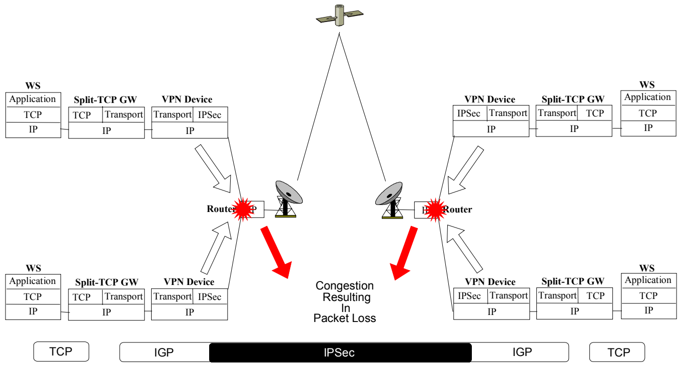

Figure 2. Congestion using Shared Gateways

SCPS-TP GATEWAYS

Our approach is to use split-TCP gateways based on the SCPS Transport Protocol (SCPS-TP[7]) to reliably send data across the satellite hop. The SCPS specification includes sender-side rate control, as well as admitting a number of congestion control strategies. Thus by using a SCPS gateway the inter-gateway traffic in our system can be subjected to congestion control. The main advantage of this method is more efficient sharing of the satellite resource. For example, with a SCPS gateway a single TCP session can use the full satellite bandwidth if no other user contends for it. If multiple users vie for the channel, the congestion control scheme detects the contention and backs each gateway's transmission rate down so that the channel is shared efficiently. The sender-side rate control ensures that the congestion control algorithm does not over-saturate the link, as could be the case if Van Jacobson congestion control were used between the gateways.

The SCPS-TP protocol is completely interoperable with TCP. SCPS includes a set of IANA registered options that negotiate the various capabilities and extensions during the connection establishment phase. These include: Selective Negative ACKnowledgment (SNACK); a partial reliability transport mechanism; a loss-tolerant header compression algorithm; record boundaries markings; and corruption experienced. SCPS-TP also implements other techniques to perform "better" in various environments. These include alternate congestion control strategies; rate control mechanisms; ability to differentiate and respond differently to loss caused by corruption, congestion, and link outages; reduced acknowledgement modes for highly asymmetric channels; and methods for dealing with extremely lossy channels.

The following list contains the SCPS extensions that are used within the SCPS-TP gateway. The SNACK option, sent by the receiver when loss has been detected, provides an immediate indication of a lost segment, which must be retransmitted. SNACKs can be more bandwidth-efficient than SACKs, as SNACKs do not have to appear on every acknowledgement when there is an out-of-sequence queue at the receiver. A combination of SNACK and SACK, however, might provide the best performance, and this is an area of ongoing work. In addition to using SNACK, the following four congestion control strategies can be applied: rate control, Van Jacobson congestion control, TCP Vegas (assume congestion), and TCP Vegas (assume corruption).

As noted above, the SCPS-TP gateways can implement a variant of the TCP Vegas implementation. The Vegas congestion control strategy can proactively detect congestion by measuring network queueing delay. This is accomplished by measuring the round-trip times using return ACK packets. If the round-trip time starts to increase, TCP Vegas assumes traffic is starting to queue in network device buffers. TCP Vegas will then reduce its rate of transmission until round-trip times decrease, at which point more traffic is introduced onto the network. By relying on queueing delays, gateway pairs operating on independent VPNs do not need to be aware of other gateways or traffic sources in the network that jointly vie for the satellite link.

In our implementation, TCP Vegas is implemented jointly with rate-control. Since the speed of the link is known by the gateways, it can accurately set buffer sizes and the TCP window size to ensure the satellite channel can be fully utilized.

In addition to the queueing delay-based congestion control of TCP Vegas, we can also make assumptions about traffic loss. We can treat loss as congestion and reduce the transmission rate, using the Van Jacobson implementation. We can also make the assumption that loss is due to corruption, which does not result in dropping the transmission rate. When we assume loss is due to corruption, we rely exclusively on queueing delay to signal congestion. In our testing scenarios, we found that Vegas assume corrupt works the best. One of the key advantages of the SCPS-TP gateways is that they can be setup behind independent VPNs and dynamically adjust their transmission rates to keep the satellite channel full without congesting network devices.

EXPERIMENT SETUP

We are concerned with improving performance of a system such as that illustrated in Figure 3. Here data of different privacy areas shares a single satellite channel.

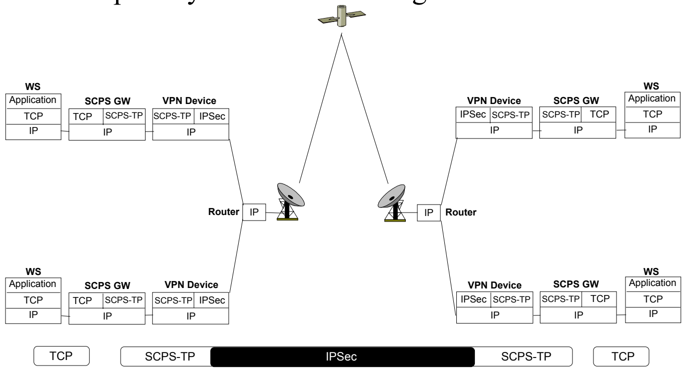

Figure 3. Emulated Test Configuration

Our lab setup emulates the configuration of Figure 3. To simulate the satellite channel, we use a MITRE developed data link simulator that is capable of delaying packets, imposing random errors, and restricting the bandwidth available to particular connections. We simulated a 512 kbps satellite link with 0.640 seconds of round-trip delay, and a random bit error rate (BER) of $1x10^{-6}$. The purpose of our testing is to observe the behavior of the channel when multiple test flows are initiated across the simulated satellite link. We implemented several different congestion control schemes in the gateways, and observed the behavior by monitoring the individual streams using the Tele Traffic Tapper (ttt) application. The plots shown in the next section are throughput plots from the ttt application.

The experiments consist of sample runs of the follow configurations:

- TCP running end-to-end without gateways.
- TCP flows gatewayed via the SCPS transport gateway, using only pure rate control on the satellite network.
- TCP flows gatewayed via the SCPS transport gateway, using the Vegas congestion control algorithm (with loss assumed to be corruption) along with rate control on the satellite network.

RESULTS

The plots below show the throughput versus time for two competing TCP traffic flows. The initial traffic flow is a 10 megabyte (MB) file, followed later by the initiation of a 5 MB file via a separate tunnel. The plots were obtained using the ttt application, which promiscuously monitored traffic on the sending side between the VPN devices and the router.

A. TCP Running End-to-End (no gateways)

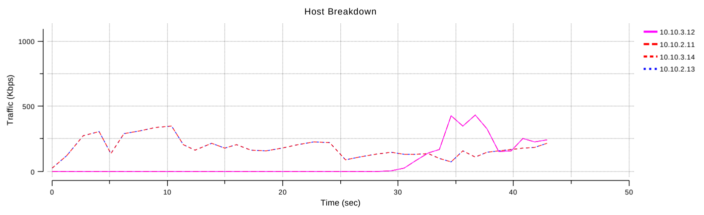

Figure 4. Standard TCP Reno Streams (no gateways)

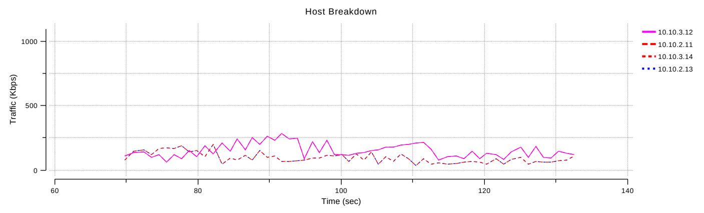

Figure 5. Standard TCP Reno Streams (no gateways)

The plots in figures 4 and 5 illustrate standard end-to-end TCP connections, showing the impact of the satellite channel delay and packet errors on the throughput of the streams. When only one stream is active (see figure 4), less than half of the available bandwidth is used by the active stream. This is common when the TCP windows are not increased to compensate for the delay in the channel (i.e., window limited). Due to the TCP window limitations of both connections, neither connection impacts the other from a performance standpoint. Additional connections would start to impact the throughput performance of the other connections. The jagged nature of the curves is an indication that packet errors are affecting the performance of each connection. TCP Reno uses Van Jacobson congestion control, resulting in the sending terminal scaling back its rate of transmission following loss.

B. Gateways Using Rate Control

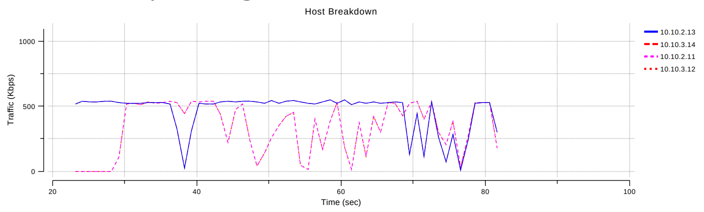

Figure 6. Gateways Using Rate Control

Figure 6 illustrates two TCP streams, each independently gatewayed using pure rate control. The gateway pairs for each stream push data onto the network at the full capacity of the satellite channel. The traffic offered to the network is twice the capacity of the satellite channel, resulting in buffer overflow and lost traffic. With pure rate control, the gateways do not backoff their transmission rate when loss is encountered, which results in significant network congestion and poor application throughput performance.

C. Gateways Using TCP Vegas Congestion Control

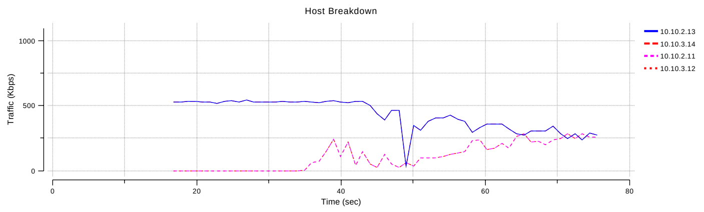

Figure 7. Gateways with TCP Vegas CC

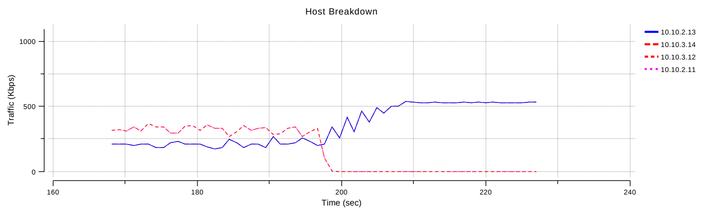

Figure 8. Gateways with TCP Vegas CC

Figures 7 and 8 illustrate two independent TCP streams gatewayed using a combination of rate control and TCP Vegas congestion control. In our TCP Vegas variant, we make the assumption that all loss is due to corruption. Therefore the gateways only reduce their transmission rate when queueing delay is experienced. Figure 7 shows a TCP session that is already in progress. The session grabs all available bandwidth of the satellite link. A second ftp session is started using a separate gateway pair approximately 35 seconds after the first session is started. The two ftp sessions converge to equally share the satellite bandwidth. Figure 8 shows the two ftp sessions later in time. When one ftp session finishes, the other ramps back up to fully utilize the channel.

CONCLUSIONS

TCP satellite gateways are proven to improve TCP throughput across high delay, lossy, satellite links. Satellite gateways typically use pure rate control to keep the channel full. This is ideal when a single gateway pair processes all traffic entering and leaving a satellite link. However, when multiple gateway pairs share a satellite link, congestion will likely occur.

When multiple gateway pairs are required or when a single satellite gateway pair does not process all traffic traversing the satellite link, we have shown that a SCPS-TP gateway implementation using a combination of rate control and a variant of the TCP Vegas congestion control scheme works very well. Our gateways dynamically share the available bandwidth and promote fairness across the competing TCP streams.

REFERENCES

[1] W. R. Stevens, *TCP/IP Illustrated,* Volume 1, Addison-Wesley, Reading, MA 1994.

[2] V. Jacobson, “Congestion Avoidance and Control,” *Proceedings of ACM SIGCOMM ’88*, pp. 314-329, Stanford, CA, August 1988.

[3] M. Allman, V. Paxson, and W. Stevens, “TCP Congestion Control”, RFC 2581, April 1999.

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

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

[6] M. Allman et. al., " Ongoing TCP Research Related to Satellites", RFC 2760, February 2000.

[7] R. Durst, G. Miller, and E. Travis, “TCP Extensions for Space Communications,” *Wireless Networks*, vol. 3, no. 5, pp 389-403, 1997.