Abstract

The Transmission Control Protocol (TCP) provides reliable delivery of data across any network path, including network paths containing satellite channels. While TCP works over satellite channels there are several mechanisms that enable TCP to more effectively utilize the available capacity of the network path. This draft outlines some of these TCP mitigations. At this time, all mitigations discussed in this draft are IETF standards track mechanisms.

1. Introduction

Satellite channel characteristics have an effect on the way transport protocols, such as the Transmission Control Protocol (TCP) [Pos81], behave. When protocols such as TCP perform poorly, channel utilization is low. While the performance of a transport protocol, such as TCP, is important, it is not the only consideration when constructing a network containing satellite links. However, this document focuses on improving TCP in the satellite environment.

This draft is divided up as follows: Section 2 provides a brief outline of the characteristics of satellite networks. Section 3 outlines two non-TCP mechanisms that enable TCP to more effectively...
utilize the available bandwidth. Section 4 outlines the TCP mechanisms defined by the IETF that benefit satellite networks. Finally, Section 5 provides a summary of what modern TCP implementations should include to be considered "satellite friendly".

2. Satellite Characteristics

There is an inherent delay in the delivery of a message over a satellite link due to the finite speed of light and the altitude of communications satellites.

Many communications satellites are located at Geostationary Orbit (GSO) with an altitude of approximately 36,000 km [Sta94]. At this altitude the orbit period is the same as the Earth’s rotation period. Therefore, each ground station is always able to "see" the orbiting satellite at the same position in the sky. The propagation time for a radio signal to travel twice that distance (corresponding to a ground station directly below the satellite) is 239.6 milliseconds (ms) [Mar78]. For ground stations at the edge of the view area of the satellite, the distance traveled is 2 x 41,756 km for a total propagation delay of 279.0 ms [Mar78]. These delays are for one ground station-to-satellite-to-ground station route (or "hop"). Therefore, the propagation delay for a message and its reply (one round-trip time or RTT) would be no more than 558 ms. The delay will be proportionately longer if the link includes multiple hops or if intersatellite links are used. As satellites become more complex and include on-board processing of signals, additional delay may be added.

Other orbits are possible for use by communications satellites including Low Earth Orbit (LEO) and Medium Earth Orbit (MEO) [Mar78]. The lower orbits require the use of constellations of satellites for constant coverage. In other words, as one satellite leaves the ground station’s sight, another satellite appears on the horizon and the channel is switched to it. The propagation delay to a LEO orbit ranges from several milliseconds when communicating with a satellite directly overhead, to as much as 80 ms when the satellite is on the horizon. These systems are more likely to use intersatellite links and have variable path delay depending on routing through the network.

Satellite channels are dominated by two fundamental characteristics, as described below:

**NOISE** - The strength of a radio signal falls in proportion to the square of the distance traveled. For a satellite link the distance is large and so the signal becomes weak before reaching its destination. This results in a low signal-to-noise ratio. Some frequencies are particularly susceptible to atmospheric effects such as rain attenuation. For mobile applications, satellite channels are especially susceptible to multi-path distortion and shadowing (e.g., blockage by buildings). Typical bit error rates for a satellite link today are on the order of 1
error per 10 million bits (1 x 10^-7) or better. Advanced error control coding (e.g., Reed Solomon) can be added to existing satellite services and is currently being used by many services. Satellite error performance equivalent to fiber will become common as advanced error control coding is used in new systems. However, many current satellite systems do not provide error free service.

BANDWIDTH - The radio spectrum is a limited natural resource, hence the bandwidth available to satellite systems is limited. Typical carrier frequencies for current, point-to-point, commercial, satellite services are 6 GHz (uplink) and 4 GHz (downlink), also known as C band, and 14/12 GHz (Ku band). A new service at 30/20 GHz (Ka band) will be emerging over the next few years. Satellite-based radio repeaters are known as transponders. Traditional C band transponder bandwidth is typically 36 MHz to accommodate one color television channel (or 1200 voice channels). Ku band transponders are typically around 50 MHz. Furthermore, one satellite may carry a few dozen transponders.

Not only is bandwidth limited by nature, but the allocations for commercial communications are limited by international agreements so that this scarce resource can be used fairly by many different applications.

Although satellites have certain disadvantages when compared to fiber channels, they also have certain advantages over terrestrial links. First, satellites have a natural broadcast capability. This gives satellites a natural advantage for multicast applications. Next, satellites can reach geographically remote areas or countries that have little terrestrial infrastructure. A related advantage is the ability of satellite links is to reach mobile users.

Satellite channels have several characteristics that differ from most terrestrial channels. These characteristics can degrade the performance of TCP. These characteristics include:

Long feedback loop

Due to the propagation delay of some satellite channels (e.g., approximately 250 ms over a geosynchronous satellite) it takes a large amount of time for a TCP sender to determine whether or not a packet has been successfully received at the final destination. This delay hurts interactive applications such as telnet, as well as some of the TCP congestion control algorithms (see section 4).

Large delay*bandwidth product

The delay*bandwidth product (DBP) defines the amount of data a protocol should have "in flight" (data that has been transmitted, but not yet acknowledged) at any one time to fully utilize the available channel capacity. The delay used in this
equation is the RTT, in practice. Because the delay in some satellite environments is large, TCP will need to keep a large amount of data "in flight".

Transmission errors

Some satellite channels exhibit a higher bit-error rate (BER) than typical terrestrial networks. TCP uses all packet drops as signals of congestion and reduces the sending rate, because TCP cannot figure out why a packet was dropped. Therefore, packets dropped due to corruption cause TCP to reduce the size of the its sliding window, even though these packet drops do not signal congestion in the network.

Asymmetric use

Due to the expense of the equipment used to send data to satellites, asymmetric satellite networks are often constructed. For example, a host connected to such a network will send all outgoing traffic over a slow terrestrial link (such as a dialup modem channel) and receive incoming traffic via the satellite channel. Another common situation arises when both the incoming and outgoing traffic are sent using a satellite link, but the uplink has less available capacity than the downlink. This asymmetry can have a large impact on TCP performance.

Variable Round Trip Times

In some satellite environments, such as low-Earth orbit (LEO) constellations, the propagation delay to and from the satellite varies over time. This can have a negative impact on TCP’s ability to accurately set retransmission timeouts and determine the appropriate window size.

Intermittent connectivity

In non-GSO satellite orbit configurations, TCP connections must be transferred from one satellite to another or from one ground station to another from time to time. This handoff can cause packet loss.

Most satellite channels only exhibit a subset of the above characteristics. In addition, some terrestrial networks exhibit some of the above characteristics, as well. The mechanisms outlined in this document should benefit most networks, especially those with one of the above characteristics.

3. Lower Level Mitigations

It is recommended that those utilizing satellite channels in their networks should use the following two non-TCP mechanisms which can increase TCP performance. These mechanisms are Path MTU Discovery and forward error correction (FEC) and are outlined in the following two sections.
3.1 Path MTU Discovery

Path MTU discovery [MD90] is used to determine the maximum packet size a connection can use on a given network path without being subjected to IP packet fragmentation. The sender transmits a packet that is the appropriate size for the local network with which it is connected (e.g., 1500 bytes on an Ethernet) and sets the IP "don't fragment" (DF) bit. If the packet is too large for a channel along the network path, the gateway that would normally fragment the packet and forward the fragments will return an ICMP message to the originator of the packet. The ICMP message will indicate that the original segment could not be transmitted without being fragmented and will also contain the maximum size that can be forwarded by the gateway. Additional information from the IESG on Path MTU discovery is available in [Kno93].

This allows TCP to use the largest possible packet size, without incurring the cost of fragmentation and reassembly. Large packets reduce the packet overhead by sending more data bytes per overhead byte. As outlined in section 4, increasing TCP’s congestion window is segment based, rather than byte based. Therefore, larger segments enable TCP senders to increase the congestion window more rapidly than smaller segments.

The disadvantage of Path MTU Discovery is that it may cause a long pause before TCP is able to start sending data. For example, assume a packet is sent with the DF bit set and one of the intervening gateways (G1) returns an ICMP message indicating that it cannot forward the segment. At this point, the sending host reduces the packet size to the size returned by G1 and sends another packet with the DF bit set. The packet will be forwarded by G1, however this does not ensure all subsequent gateways in the network path will be able to forward the segment. If a second gateway (G2) cannot forward the segment it will return an ICMP message to the transmitting host and the process will be repeated. Therefore, path MTU discovery can waste a large amount of time determining the maximum allowable packet size on the network path between the sender and receiver. Satellite delays can aggravate this problem (consider the case when the channel between G1 and G2 is a satellite link). However, in practice, Path MTU Discovery is not that time consuming due to wide support of common MTU values.

The use of large segments may pose a problem on links with a high BER. In such an environment, the probability that a segment will incur a bit error increases with the size of the segment. Therefore, using smaller segments may ensure that more segments are delivered and less data is retransmitted over high BER channels.

The relationship between BER and segment size is likely to vary depending on the error characteristics of the given channel. This
A loss event in TCP is always interpreted as an indication of congestion and always causes TCP to reduce the window size. When loss occurs during slow start, then slow start is terminated and TCP enters congestion avoidance. Premature termination of slow start and entry into congestion avoidance due to losses other than congestion losses will cause needless inefficiency in channel utilization. Furthermore, drops due to corruption causes TCP to needlessly reduce the amount of data being injected into the network.

For TCP to operate efficiently, the channel characteristics should be such that nearly all loss is due to network congestion. The use of forward error correction coding (FEC) on a satellite link should be used to bring the performance of the link to at least fiber quality. Because of the effect of long RTT, the time needed to recover from errors on a satellite link is longer than for a terrestrial network with shorter RTT [PS97]. There are some applications, such as military jamming, where FEC cannot be expected to solve the noise problem.

4. Standard TCP Mechanisms

This section includes an outline of the mechanisms that may be necessary in satellite or hybrid satellite/terrestrial networks to better utilize the available capacity of the link. These mechanisms may also be needed to fully utilize fast terrestrial channels. Furthermore, these mechanisms do not fundamentally hurt performance in a shared terrestrial network. Each of the following sections outlines one mechanism and why that mechanism may be needed.

4.1 Congestion Control

To avoid generating an inappropriate amount of network traffic for the current network conditions TCP employs four congestion control mechanisms [JK88] [Jac90] [Ste97]. These algorithms are slow start, congestion avoidance, fast retransmit and fast recovery. These algorithms are used to adjust the amount of unacknowledged data that can be injected into the network and to retransmit segments dropped by the network.

TCP uses two variables to accomplish congestion control. The first variable is the congestion window (cwnd). This is an upper bound on the amount of data the sender can inject into the network before receiving an acknowledgment (ACK). The value of cwnd is limited to the receiver's advertised window. The congestion window is increased or decreased during the transfer based on the inferred amount of congestion present in the network. The second variable is
the slow start threshold (ssthresh). This variable determines which algorithm is being used to increase the value of cwnd. If cwnd is less than ssthresh the slow start algorithm is used to increase the value of cwnd. However, if cwnd is greater than or equal to ssthresh the congestion avoidance algorithm is used. The initial value of ssthresh is the receiver’s advertised window size. Furthermore, the value of ssthresh is reduced when congestion is detected.

The four congestion control algorithms are outlined below, followed by a brief discussion of the impact of satellite environments on these algorithms.

4.1.1 Slow Start and Congestion Avoidance

When a host begins sending data on a TCP connection the host has no knowledge of the current state of the network between itself and the data receiver. In order to avoid transmitting an inappropriately large burst of traffic, the data sender is required to use the slow start algorithm at the beginning of a transfer [JK88] [Bra89] [Ste97]. Slow start begins by initializing cwnd to 1 segment. This forces TCP to transmit one segment and wait for the corresponding ACK. For each ACK that is received, the value of cwnd is increased by 1 segment. For example, after the first ACK is received cwnd will be 2 segments and the sender will be allowed to transmit 2 data packets. This continues until cwnd meets or exceeds ssthresh, or loss is detected.

When the value of cwnd is greater than or equal to ssthresh the congestion avoidance algorithm is used to increase cwnd [JK88] [Bra89] [Ste97]. This algorithm increases the size of cwnd more slowly than does slow start. Congestion avoidance is used to probe the network for any additional capacity. During congestion avoidance, cwnd is increased by 1/cwnd for each incoming ACK. Therefore, if one ACK is received for every data segment, cwnd will increase by 1 segment per round-trip time (RTT).

Long-delay satellite networks force poor utilization of the available channel bandwidth when using the slow start and congestion control algorithms [All97]. For example, transmission begins with the transmission of one segment. After the first segment is transmitted the data sender is forced to wait for the corresponding ACK. When using a GSO satellite this leads to an idle time of roughly 500 ms when no useful work is being accomplished. Therefore, slow start takes more real time over GSO satellites than on typical terrestrial channels. This holds for congestion avoidance, as well [All97]. This is precisely why Path MTU Discovery is an important algorithm. While the number of segments we transmit is determined by the congestion control algorithms, the size of these segments is not. Therefore, using larger packets will enable TCP to send more data per segment which yields better channel utilization.

4.1.2 Fast Retransmit and Fast Recovery
TCP’s default mechanism to detect dropped segments is a timeout [Pos81]. In other words, if the sender does not receive an ACK for a given packet within the expected amount of time the segment will be retransmitted. The retransmission timeout (RTO) is based on observations of the RTT. In addition to retransmitting a segment when the RTO expires, TCP also uses the lost segment as an indication of congestion in the network. In response to the congestion, the value of ssthresh is set to half of the cwnd and the value of cwnd is then reduced to 1 segment. This triggers the use of the slow start algorithm to increase cwnd until the value of cwnd reaches half of its value when congestion was detected. After the slow start phase, the congestion avoidance algorithm is used to probe the network for additional capacity.

TCP ACKs always acknowledge the highest in-order segment that has arrived. Therefore an ACK for segment X also effectively ACKs all segments < X. Furthermore, if a segment arrives out-of-order the ACK triggered will be for the highest in-order segment, rather than the segment that just arrived. For example, assume segment 11 has been dropped somewhere in the network and segment 12 arrives at the receiver. The receiver is going to send a duplicate ACK covering segment 10 (and all previous segments).

The fast retransmit algorithm uses these duplicate ACKs to detect lost segments. If 3 duplicate ACKs arrive at the data originator, TCP assumes that a segment has been lost and retransmits the missing segment without waiting for the RTO to expire. After a segment is resent using fast retransmit, the fast recovery algorithm is used to adjust the congestion window. First, the value of ssthresh is set to half of the value of cwnd. Next, the value of cwnd is halved. Finally, the value of cwnd is artificially increased by 1 segment for each duplicate ACK that has arrived. The artificial inflation can be done because each duplicate ACK represents 1 segment that has left the network. When the cwnd permits, TCP is able to transmit new data. This allows TCP to keep data flowing through the network at half the rate it was when loss was detected. When an ACK for the retransmitted packet arrives, the value of cwnd is reduced back to ssthresh (half the value of cwnd when the congestion was detected).

Fast retransmit can resend only one segment per window of data sent. When multiple segments are lost in a given window of data, one of the segments will be resent using fast retransmit and the rest of the dropped segments must wait for the RTO to expire, which causes TCP to revert to slow start.

TCP’s response to congestion differs based on the way the congestion was detected. If the retransmission timer causes a packet to be resent, TCP drops ssthresh to half the current cwnd and reduces the value of cwnd to 1 segment (thus triggering slow start). However, if a segment is resent via fast retransmit both ssthresh and cwnd are set to half the current value of cwnd and congestion avoidance
is used to send new data. The difference is that when retransmitting due to duplicate ACKs, TCP knows that packets are still flowing through the network and can therefore infer that the congestion is not that bad. However, when resending a packet due to the expiration of the retransmission timer, TCP cannot infer anything about the state of the network and therefore must proceed conservatively by sending new data using the slow start algorithm.

4.1.3 Congestion Control in Satellite Environment

The above algorithms have a negative impact on the performance of individual TCP connection’s performance, especially over long-delay satellite channels [All97] [AHKO97]. However, the algorithms are necessary to prevent congestive collapse in a shared network [JK88]. Therefore, the negative impact on a given connection is more than offset by the benefit to the entire network.

4.2 Large TCP Windows

The standard TCP window size (65,535 bytes) is not adequate to allow a single TCP connection to utilize the entire bandwidth available on some satellite channels. TCP throughput is limited by the following formula [Pos81]:

$$\text{throughput} = \frac{\text{window size}}{\text{RTT}}$$

Therefore, using the maximum window size of 65,535 bytes and a geosynchronous satellite channel RTT of 560 ms [Kru95] the maximum throughput is limited to:

$$\text{throughput} = \frac{65,535 \text{ bytes}}{560 \text{ ms}} = 117,027 \text{ bytes/second}$$

Therefore, a single standard TCP connection cannot fully utilize, for example, T1 rate (approximately 192,000 bytes/second) GSO satellite channels. However, TCP has been extended to support larger windows [JBB92]. The window scaling options outlined in [JBB92] should be used in satellite environments, as well as the companion algorithms PAWS (Protection Against Wrapped Sequence space) and RTTM (Round-Trip Time Measurements).

It should be noted that for a satellite link shared among many flows, large windows may not be necessary. For instance, two long-lived TCP connections each using a window of 65,535 bytes, as in the above example, can fully utilize a T1 GSO satellite channel.

4.3 Selective Acknowledgments

Selective acknowledgments (SACKs) [MMFR96] allow TCP receivers to inform TCP senders exactly which packets have arrived. TCP senders that do not use SACKs must infer which segments have not arrived and retransmit accordingly. This can lead to needless retransmissions, in the case when the sender infers incorrectly. When utilizing SACKs, the sender does not need to guess which segments have not arrived.
Some satellite channels require the use of large TCP windows to fully utilize the available capacity, as discussed above. With the use of large windows, the likelihood of losing multiple segments in a given window of data increases. When multiple segments are lost, SACKs will ensure the data sender retransmits only those segments that were dropped and not those that safely arrived at the receiver.

5. Mitigation Summary

Table 1 summarizes the mechanisms that have been discussed in this document. Those mechanisms denoted "Recommended" are IETF standards track mechanisms that are recommended by the authors for use in networks containing satellite channels. Those mechanisms marked "Required" have been defined by the IETF as required for hosts using the shared Internet [Bra89]. Along with the section of this document containing the discussion of each mechanism, we note where the mechanism needs to be implemented. The codes listed in the last column are defined as follows: "S" for the data sender, "R" for the data receiver and "GS" for the satellite ground station.

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>Use</th>
<th>Section</th>
<th>Where</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path-MTU Discovery</td>
<td>Recommended</td>
<td>3.1</td>
<td>S</td>
</tr>
<tr>
<td>FEC</td>
<td>Recommended</td>
<td>3.2</td>
<td>GS</td>
</tr>
<tr>
<td>TCP Congestion Control</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slow Start</td>
<td>Required</td>
<td>4.1.1</td>
<td>S</td>
</tr>
<tr>
<td>Congestion Avoidance</td>
<td>Required</td>
<td>4.1.1</td>
<td>S</td>
</tr>
<tr>
<td>Fast Retransmit</td>
<td>Recommended</td>
<td>4.1.2</td>
<td>S</td>
</tr>
<tr>
<td>Fast Recovery</td>
<td>Recommended</td>
<td>4.1.2</td>
<td>S</td>
</tr>
<tr>
<td>TCP Large Windows</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Window Scaling</td>
<td>Recommended</td>
<td>4.2</td>
<td>S,R</td>
</tr>
<tr>
<td>PAWS</td>
<td>Recommended</td>
<td>4.2</td>
<td>S,R</td>
</tr>
<tr>
<td>RTTM</td>
<td>Recommended</td>
<td>4.2</td>
<td>S,R</td>
</tr>
<tr>
<td>TCP SACKs</td>
<td>Recommended</td>
<td>4.3</td>
<td>S,R</td>
</tr>
</tbody>
</table>

Satellite users should check with their TCP vendors (implementors) to ensure the recommended mechanisms are supported in their stack in current and/or future versions.

Work on improving the efficiency of TCP over satellite channels is ongoing and will be summarized in a planned memo along with other considerations, such as network architectures.

6. Security

The recommendations contained in this memo do not alter the security implications of TCP.
Acknowledgments

This document has benefited from comments from the members of the TCP Over Satellite Working Group. In particular, we would like to thank Aaron Falk, Matthew Halsey and Greg Nakanishi.

References


Author’s Addresses:

Mark Allman
NASA Lewis Research Center/Sterling Software
21000 Brookpark Rd. MS 54-2
Cleveland, OH 44135
mallman@lerc.nasa.gov
http://gigahertz.lerc.nasa.gov/~mallman

Dan Glover
NASA Lewis Research Center
21000 Brookpark Rd. MS 54-2
Cleveland, OH 44135
Daniel.R.Glover@lerc.nasa.gov