Internet Engineering Task Force
INTERNET DRAFT
File: draft-ietf-tcpsat-stand-mech-06.txt

Mark Allman
NASA Lewis/Sterling Software
Daniel R. Glover
NASA Lewis
Luis A. Sanchez
BBN

September, 1998
Expires: March, 1999

Enhancing TCP Over Satellite Channels using Standard Mechanisms

Status of this Memo

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 IETF standardized mechanisms that enable TCP to more effectively utilize the available capacity of the network path. This document outlines some of these TCP mitigations. At this time, all mitigations discussed in this document are IETF standards track mechanisms (or are compliant with IETF standards).

1. Introduction

Satellite channel characteristics may 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 is important, it is not the only consideration when constructing a network containing satellite links. For example,

data link protocol, application protocol, router buffer size, queueing discipline and proxy location are some of the considerations that must be taken into account. However, this

document focuses on improving TCP in the satellite environment and non-TCP considerations are left for another document. Finally, there have been many satellite mitigations proposed and studied by the research community. While these mitigations may prove useful and safe for shared networks in the future, this document only considers TCP mechanisms which are currently well understood and on the IETF standards track (or are compliant with IETF standards).

This document 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 may 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 the corresponding reply (one round-trip time or RTT) could be at least 558 ms. The RTT is not based solely on satellite propagation time. The RTT will be increased by other factors in the network, such as the transmission time and propagation time of other links in the network path and queueing delay in gateways. Furthermore, the satellite propagation delay will be 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) [Stu95] [Mon98] 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 (BER) for a satellite link today are on the order of 1 error per 10 million bits (1 x 10^-7) or less frequent. 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 approaching fiber will become more common as advanced error control coding is used in new systems. However, many legacy satellite systems will continue to exhibit higher BER than newer satellite systems and terrestrial channels.

BANDWIDTH - The radio spectrum is a limited natural resource, hence there is a restricted amount of bandwidth available to satellite systems which is typically controlled by licenses. This scarcity makes it difficult to trade bandwidth to solve other design problems. 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 (e.g., cannot be easily repaired, rain fades, etc.), they also have certain advantages over terrestrial links. First, satellites have a natural broadcast capability. This gives satellites an 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 to reach mobile users.

Satellite channels have several characteristics that differ from

most terrestrial channels. These characteristics may 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 may take a long 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 and the bandwidth is the capacity of the bottleneck link in the network path. Because the delay in some satellite environments is large, TCP will need to keep a large number of packets ``in flight'' (that is, sent but not yet acknowledged) .

Transmission errors

Satellite channels exhibit a higher bit-error rate (BER) than typical terrestrial networks. TCP uses all packet drops as signals of network congestion and reduces its window size in an attempt to alleviate the congestion. In the absence of knowledge about why a packet was dropped (congestion or corruption), TCP must assume the drop was due to network congestion to avoid congestion collapse [Jac88] [FF98]. Therefore, packets dropped due to corruption cause TCP to reduce the size of 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 a satellite 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 due to the expense of the transmitter required to

provide a high bandwidth back channel. This asymmetry may have an 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. Whether or not this will have an impact on TCP performance is currently an open question.

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 may cause packet loss if not properly performed.

Most satellite channels only exhibit a subset of the above characteristics. Furthermore, satellite networks are not the only environments where the above characteristics are found. However, satellite networks do tend to exhibit more of the above problems or the above problems are aggravated in the satellite environment. The mechanisms outlined in this document should benefit most networks, especially those with one or more of the above characteristics (e.g., gigabit networks have large delay*bandwidth products).

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.

The data link layer protocol employed over a satellite channel can have a large impact on performance of higher layer protocols. While beyond the scope of this document, those constructing satellite networks should tune these protocols in an appropriate manner to ensure that the data link protocol does not limit TCP performance. In particular, data link layer protocols often implement a flow control window and retransmission mechanisms. When the link level window size is too small, performance will suffer just as when the TCP window size is too small (see section 4.3 for a discussion of appropriate window sizes). The impact that link level retransmissions have on TCP transfers is not currently well understood. The interaction between TCP retransmissions and link level retransmissions is a subject for further research.

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 fragmentation. The sender transmits a packet that

is the appropriate size for the local network to 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 to be forwarded without being fragmented to a given channel along the network path,

the gateway that would normally fragment the packet and forward the fragments will instead 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 size of the largest packet that can be forwarded by the gateway. Additional information from the IESG regarding Path MTU discovery is available in [Kno93].

Path MTU Discovery 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 and therefore, larger segments enable TCP senders to increase the congestion window more rapidly, in terms of bytes, than smaller segments.

The disadvantage of Path MTU Discovery is that it may cause a delay 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 per the ICMP message 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 spend 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 does not consume a large amount of time due to wide support of common MTU values. Additionally, caching MTU values may be able to eliminate discovery time in many instances, although the exact implementation of this and the aging of cached values remains an open problem.

The relationship between BER and segment size is likely to vary depending on the error characteristics of the given channel. This relationship deserves further study, however with the use of good forward error correction (see section 3.2) larger segments should provide better performance, as with any network [MSMO97]. While the exact method for choosing the best MTU for a satellite link is outside the scope of this document, the use of Path MTU Discovery is recommended to allow TCP to use the largest possible MTU over the satellite channel.

3.2 Forward Error Correction

A loss event in TCP is always interpreted as an indication of congestion and always causes TCP to reduce its congestion window size. Since the congestion window grows based on returning acknowledgments (see section 4), TCP spends a long time recovering

from loss when operating in satellite networks. When packet loss is due to corruption, rather than congestion, TCP does not need to reduce its congestion window size. However, at the present time detecting corruption loss is a research issue.

Therefore, 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 improve the bit-error rate (BER) of the satellite channel. Reducing the BER is not always possible in satellite environments. However, since TCP takes a long time to recover from lost packets because the long propagation delay imposed by a satellite link delays feedback from the receiver [PS97], the link should be made as clean as possible to prevent TCP connections from receiving false congestion signals. This document does not make a specific BER recommendation for TCP other than it should be as low as possible.

FEC should not be expected to fix all problems associated with noisy satellite links. There are some situations where FEC cannot be expected to solve the noise problem (such as military jamming, deep space missions, noise caused by rain fade, etc.). In addition, link outages can also cause problems in satellite systems that do not occur as frequently in terrestrial networks. Finally, FEC is not without cost. FEC requires additional hardware and uses some of the available bandwidth. It can add delay and timing jitter due to the processing time of the coder/decoder.

Further research is needed into mechanisms that allow TCP to differentiate between congestion induced drops and those caused by corruption. Such a mechanism would allow TCP to respond to congestion in an appropriate manner, as well as repairing corruption induced loss without reducing the transmission rate. However, in the absence of such a mechanism packet loss must be assumed to indicate congestion to preserve network stability. Incorrectly interpreting loss as caused by corruption and not reducing the transmission rate accordingly can lead to congestive collapse [Jac88] [FF98].

4. Standard TCP Mechanisms

This section outlines TCP 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, during a connection TCP employs four congestion control mechanisms [Jac88] [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 senders use two state 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 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 (or just greater than in some TCP implementations) 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 set 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 [Jac88] [Bra89] [Ste97]. Slow start begins by initializing cwnd to 1 segment (although an IETF experimental mechanism would increase the size of the initial window to roughly 4 Kbytes [AFP98]) and ssthresh to the receiver's advertised window. This forces TCP to transmit one segment and wait for the corresponding ACK. For each ACK that is received during slow start, 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, in some implementations when cwnd equals ssthresh), or loss is detected.

When the value of cwnd is greater than or equal to (or equal to in certain implementations) ssthresh the congestion avoidance algorithm is used to increase cwnd [Jac88] [Bra89] [Ste97]. This algorithm increases the size of cwnd more slowly than does slow start. Congestion avoidance is used to slowly probe the network for

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 roughly 1 segment per round-trip time (RTT).

The slow start and congestion control algorithms can force poor utilization of the available channel bandwidth when using long-delay satellite networks [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).

Generally, 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 usually 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 is 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.

Note that the fast retransmit/fast recovery algorithms, as discussed above can lead to a phenomenon that allows multiple fast retransmits per window of data [Flo94]. This can reduce the size of the congestion window multiple times in response to a single ``loss event''. The problem is particularly noticeable in connections that utilize large congestion windows, since these connections are able to inject enough new segments into the network during recovery to trigger the multiple fast retransmits. Reducing cwnd multiple times for a single loss event may hurt performance [GJKFV98].

The best way to improve the fast retransmit/fast recovery algorithms is to use a selective acknowledgment (SACK) based algorithm for loss recovery. As discussed below, these algorithms are generally able to quickly recover from multiple lost segments without needlessly reducing the value of cwnd. In the absence of SACKs, the fast retransmit and fast recovery algorithms should be used. Fixing these algorithms to achieve better performance in the face of multiple fast retransmissions is beyond the scope of this document. Therefore, TCP implementers are advised to implement the current version of fast retransmit/fast recovery outlined in RFC 2001 [Ste97] or subsequent versions of RFC 2001.

4.1.3 Congestion Control in Satellite Environment

The above algorithms have a negative impact on the performance of individual TCP connection's performance because the algorithms

slowly probe the network for additional capacity, which in turn wastes bandwidth. This is especially true over long-delay satellite channels because of the large amount of time required for the sender to obtain feedback from the receiver [All97] [AHKO97]. However, the algorithms are necessary to prevent congestive collapse in a shared

network [Jac88]. 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 maximum 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]:

throughput = window size / 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:

throughput = 65,535 bytes / 560 ms = 117,027 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.

Using large windows often requires both client and server applications or TCP stacks to be hand tuned (usually by an expert) to utilize large windows. Research into operating system mechanisms that are able to adjust the buffer capacity as dictated by the current network conditions is currently underway [SMM98]. This will allow stock TCP implementations and applications to better utilize the capacity provided by the underlying network.

4.3 Acknowledgment Strategies

There are two standard methods that can be used by TCP receivers to generated acknowledgments. The method outlined in [Pos81] generates an ACK for each incoming segment. [Bra89] states that hosts SHOULD use ``delayed acknowledgments''. Using this algorithm, an ACK is generated for every second full-sized segment, or if a second full-size segment does not arrive within a given timeout (which must not exceed 500 ms). The congestion window is increased based on the number of incoming ACKs and delayed ACKs reduce the number of ACKs being sent by the receiver. Therefore, cwnd growth occurs much more

slowly when using delayed ACKs compared to the case when the receiver ACKs each incoming segment [All98].

A tempting ``fix'' to the problem caused by delayed ACKs is to simply turn the mechanism off and let the receiver ACK each incoming segment. However, this is not recommended. First, [Bra89] says that a TCP receiver SHOULD generate delayed ACKs. And, second, increasing the number of ACKs by a factor of two in a shared network may have consequences that are not yet understood. Therefore, disabling delayed ACKs is still a research issue and thus, at this time TCP receivers should continue to generate delayed ACKs, per [Bra89].

4.4 Selective Acknowledgments

Selective acknowledgments (SACKs) [MMFR96] allow TCP receivers to inform TCP senders exactly which packets have arrived. SACKs allow TCP to recover more quickly from lost segments, as well as avoiding needless retransmissions.

The fast retransmit algorithm can generally only repair one loss per window of data. When multiple losses occur, the sender generally must rely on a timeout to determine which segment needs to be retransmitted next. While waiting for a timeout, the data segments and their acknowledgments drain from the network. In the absence of incoming ACKs to clock new segments into the network, the sender must use the slow start algorithm to restart transmission. As discussed above, the slow start algorithm can be time consuming over satellite channels. When SACKs are employed, the sender is generally able to determine which segments need to be retransmitted in the first RTT following loss detection. This allows the sender to continue to transmit segments (retransmissions and new segments, if appropriate) at an appropriate rate and therefore sustain the ACK clock. This avoids a costly slow start period following multiple lost segments. Generally SACK is able to retransmit all dropped segments within the first RTT following the loss detection. [MM96] and [FF96] discuss specific congestion control algorithms that rely on SACK information to determine which segments need to be retransmitted and when it is appropriate to transmit those segments. Both these algorithms follow the basic principles of congestion control outlined in [Jac88] and reduce the window by half when congestion is detected.

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 ``L'' for the satellite link.

Mechanism	Use	Section	Where
Path-MTU Discovery	Recommended	3.1	S
FEC	Recommended	3.2	L
TCP Congestion Control			
Slow Start	Required	4.1.1	S
Congestion Avoidance	Required	4.1.1	S
Fast Retransmit	Recommended	4.1.2	S
Fast Recovery	Recommended	4.1.2	S
TCP Large Windows			
Window Scaling	Recommended	4.2	S,R
PAWS	Recommended	4.2	S,R
RTTM	Recommended	4.2	S,R
TCP SACKs	Recommended	4.4	S,R

Table 1

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. Alternatively, the Pittsburgh Supercomputer Center tracks TCP implementations and which extensions they support, as well as providing guidance on tuning various TCP implementations [PSC].

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

6. Security Considerations

The authors believe that the recommendations contained in this memo do not alter the security implications of TCP. However, when using a broadcast medium such as satellites links to transfer user data and/or network control traffic, one should be aware of the intrinsic security implications of such technology.

Eavesdropping on network links is a form of passive attack that, if performed successfully, could reveal critical traffic control information that would jeopardize the proper functioning of the network. These attacks could reduce the ability of the network to provide data transmission services efficiently. Eavesdroppers could also compromise the privacy of user data, especially if end-to-end security mechanisms are not in use. While passive monitoring can occur on any network, the wireless broadcast nature of satellite links allows reception of signals without physical connection to the network which enables monitoring to be conducted without detection. However, it should be noted that the resources needed to monitor a satellite link are non-trivial.

Data encryption at the physical and/or link layers can provide secure communication over satellite channels. However, this still leaves traffic vulnerable to eavesdropping on networks before and after traversing the satellite link. Therefore, end-to-end security mechanisms should be considered. This document does not make any

recommendations as to which security mechanisms should be employed. However, those operating and using satellite networks should survey the currently available network security mechanisms and choose those that meet their security requirements.

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, Hans Kruse, Matt Mathis, Greg Nakanishi, Vern Paxson, Jeff Semke, Bill Sepmeier and Eric Travis for their useful comments about this document.

References

[AFP98] Mark Allman, Sally Floyd, Craig Partridge. Increasing TCP's Initial Window, September 1998. RFC 2414.

[AHKO97] Mark Allman, Chris Hayes, Hans Kruse, and Shawn Ostermann. TCP Performance Over Satellite Links. In Proceedings of the 5th International Conference on Telecommunication Systems, March 1997.

[All97] Mark Allman. Improving TCP Performance Over Satellite Channels. Master's thesis, Ohio University, June 1997.

[All98] Mark Allman. On the Generation and Use of TCP Acknowledgments. ACM Computer Communication Review, 28(5), October 1998.

[Bra89] Robert Braden. Requirements for Internet Hosts -- Communication Layers, October 1989. RFC 1122.

[FF96] Kevin Fall and Sally Floyd. Simulation-based Comparisons of Tahoe, Reno and SACK TCP. Computer Communication Review, July 1996.

[FF98] Sally Floyd, Kevin Fall. Promoting the Use of End-to-End Congestion Control in the Internet. Submitted to IEEE Transactions on Networking.

[Flo94] S. Floyd, TCP and Successive Fast Retransmits. Technical report, October 1994. ftp://ftp.ee.lbl.gov/papers/fastretrans.ps.

[GJKFV98] Rohit Goyal, Raj Jain, Shiv Kalyanaraman, Sonia Fahmy, Bobby Vandalore, Improving the Performance of TCP over the ATM-UBR service, 1998. Sumbitted to Computer Communications.

[Jac90] Van Jacobson. Modified TCP Congestion Avoidance Algorithm.

Technical Report, LBL, April 1990.

[JBB92] Van Jacobson, Robert Braden, and David Borman. TCP Extensions for High Performance, May 1992. RFC 1323.

[Jac88] Van Jacobson. Congestion Avoidance and Control. In ACM SIGCOMM, 1988.

[Kno93] Steve Knowles. IESG Advice from Experience with Path MTU Discovery, March 1993. RFC 1435.

[Mar78] James Martin. Communications Satellite Systems. Prentice Hall, 1978.

[MD90] Jeff Mogul and Steve Deering. Path MTU Discovery, November 1990. RFC 1191.

[MM96] Matt Mathis and Jamshid Mahdavi. Forward Acknowledgment: Refining TCP Congestion Control. In ACM SIGCOMM, 1996.

[MMFR96] Matt Mathis, Jamshid Mahdavi, Sally Floyd, and Allyn Romanow. TCP Selective Acknowledgment Options, October 1996. RFC 2018.

[Mon98] M. J. Montpetit. TELEDESIC: Enabling The Global Community Interaccess. In Proc. of the International Wireless Symposium, May 1998.

[MSMO97] M. Mathis, J. Semke, J. Mahdavi, T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", Computer Communication Review, volume 27, number3, July 1997. available from http://www.psc.edu/networking/papers/papers.html.

[Pos81] Jon Postel. Transmission Control Protocol, September 1981. RFC 793.

[PS97] Craig Partridge and Tim Shepard. TCP Performance Over Satellite Links. IEEE Network, 11(5), September/October 1997.

[PSC] Jamshid Mahdavi. Enabling High Performance Data Transfers on Hosts. http://www.psc.edu/networking/perf_tune.html.

[SMM98] Jeff Semke, Jamshid Mahdavi and Matt Mathis. Automatic TCP Buffer Tuning. In ACM SIGCOMM, August 1998. To appear.

[Sta94] William Stallings. Data and Computer Communications. MacMillian, 4th edition, 1994.

[Ste97] W. Richard Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, January 1997. RFC 2001.

[Stu95] M. A. Sturza. Architecture of the TELEDESIC Satellite System. In Proceedings of the International Mobile Satellite

Conference, 1995.

Author's Addresses:

Mark Allman
NASA Lewis Research Center/Sterling Software
21000 Brookpark Rd. MS 54-2
Cleveland, OH 44135
mallman@lerc.nasa.gov
+1 216 433 6586
http://gigahertz.lerc.nasa.gov/~mallman

Daniel R. Glover
NASA Lewis Research Center
21000 Brookpark Rd. MS 54-2
Cleveland, OH 44135
Daniel.R.Glover@lerc.nasa.gov
+1 216 433 2847

Luis A. Sanchez
BBN Technologies
GTE Internetworking
10 Moulton Street
Cambridge, MA 02140
USA
lsanchez@ir.bbn.com
+1 617 873 3351