TCP WITH SATELITE LINK

(TCP IN SPACE:-)

Philip Ginzboorg
Communication Systems Lab
Nokia Research Center
P.O. Box 422
FIN-00045, NOKIA GROUP, Finland
Email: philip.ginzboorg@research.nokia.com

March 19, 1999

Abstract

The satellite environment may severly limit TCP performance; achieving high data rates using TCP over satellite networks can be difficult. In this text we describe why TCP underperforms in satellite environment and the ways to deal with those difficilties.

1 Introduction

Compared to terrestrial links satellite links may be characterized by

- Long propagation delays. For example, the round trip time through Low Earth Orbit (LEO) satellite is 0.25s and through Geosynchronous Earth Orbit (GEO) satellite is about 0.5s. This is 20 and 100 times more than 0.005s, which is the round trip time in a LAN environment [1].
- Higher error rates, which could be as high as 10^{-5} [1, 2].
- Frequent link outage, which results from a satellite temporary passing out of view of the ground station, or other short term interruptions.
- Asymmetric forward and return channels. One example of an asymmetric channels are satellite down-link and terrestrial up-plink, e.g. a telephone line. Another example, is the communication with a space probe with a high bandwidth channel for transferring data from the probe to the ground and a low bandwidth channel for controlling the probe.
- High cost.

The satellite environment may severely limit TCP performance; achieving high data rates using TCP over satellite networks can be difficult. In this text we describe why TCP under-performs in satellite environment and the ways to deal with those difficulties. Satellites have a natural advantage in point-to-multipoint communication. For example, an MBone data stream could be transmitted up to a satellite and then relayed down to a large geographical area. Any ground station in that area could pick up the signal if tuned to the right MBone channel. We will not discuss multicast farther in this paper because we are concerned with TCP which is a point-to-point protocol.
The rest of this text is organized as follows. Section 2 contains a literature survey. In section 3 we provide a brief background on the congestion avoidance and slow start features of TCP. Section 4 describes the problems that occur in the operation of TCP over satellite links as well as the enhancements that mitigate those problems. In section 5 we describe two main ways to improve system performance: a special purpose application that opens multiple TCP channels and a relay station that splits a long control loop into two short ones. The paper ends with Conclusion section.

2 Literature survey

Transmission Control Protocol (TCP) is defined in RFC 793 [3]. In Chapter 24 of [4] R. Stevens dicusses some proposed modifications to TCP that allow it to obtain better throughput at higher speeds. Those include the window scale option, the timestamp option and the PAWS (Protection Against Wrapped Sequence numbers) algorithm that are defined in RFC 1323 [5]. RFC 1323 is a descendant of RFC 1072 [6]. The earlier RFC 1072 contained a selective acknowledgement option for TCP which was later removed from RFC 1323 because it was incompatible with the window scaling option. Later, another ver-

sion of selective acknowledgement mechanism which corrects the shortcomings of RFC 1072 was defined by Mathis et al in RFC 2018 [7].
Compressing the segment header can improve TCP performance when the forward and backward channels are asymmetrical. A differential compression scheme for TCP headers is defined in RFC 1144 [8].
R. Durst et al discuss TCP extensions for space communications in [2]. They criticize both RFC 2018 and RFC 1144 and propose alternative implementations. Selective acknowledgement of RFC 2018 is criticized because the feature consumes too much space in the TCP header and the header compression scheme of RFC 1144 is criticized because differential compression is not suitable for environments in which synchronization is expensive. In [9] Lakshman et al present a simple model of TCP behavior over asymmetric forward and backward channels. They show that TCP performance deteriorates rapidly once the product of the loss probability, the asymmetry ratio and the square of the bandwidth delay product exceeds a threshold.
A paper by C. Partridge and T. J. Shepard surveys the issues concerning TCP/IP performance over satellite link [1]. They conclude that the first step to achieve high performance is making sure that the sending and receiving TCP implementations contain all the modern features: large windows, PAWS and selective acknowledgements and that the TCP window space is larger than the delay bandwidth product of the path. A paper by M. Allman et al describes experimental results of running a TCP with modern features over a satellite link [10]. They also present XFTP, an application for bulk data transfer that improves transmission by opening several parallel TCP channels between sender and receiver.
The reports by C. Barakat and E. Altman [11] and N. Chaher et al [12] are concerned with the problems that occur when the buffer of the receiving TCP is much smaller than the bandwdth delay product of the network.
RFC 2488 [13], RFC 2414 [13] and the Internet Draft "Ongoing TCP Research Related to Satellites" [14] represent the work of IETF concerned with TCP over satellites. RFC 2488 is a summary of IETF standardized mechanizms, e. g. selective acknowledgements, that enable TCP to more effectively utilize the available capacity of the network path. The Internet Draft [14] is a systematic survey of new modifications that are proposed by the research community.
A discussion on how to improve TCP performance over wireless links can be found in the articles of H. Balakrishnan et al [15, 16].

3 Background on TCP

Design principles. TCP is designed to fill an otherwise idle link and if the link is not idle to share the available bandwidth with other users. The protocol design follows the end-to-end philosophy: it requires nothing from the network except for existence of data link layer.
Reliable transmission. Transmission is made reliable via the use of sequence numbers and acknowledgments. Conceptually, each octet of data is assigned a sequence number. The sequence number of the first octet of data in a segment (TCP packet) is transmitted with that segment and is called the segment sequence number. Segments also carry and acknowledgment number which is the sequence number of the next expected data segment in the reverse direction. When TCP transmits a segment containing data, it puts a copy on a retransmission queue and starts a timer; when the acknowledgment for that data is received, the segment is deleted from the queue. If the acknowledgment is not received before the timer has run out, the segment is retransmitted.
TCP is a "closed loop" protocol because it relies on a stream of positive acknowledgments to clock out data continuously from the sender. A stream of acknowledgments is essential to the operation of TCP.
Flow control. The receiving TCP reports a window to the sending TCP. This window specifies the number of octets, starting with the acknowledgment number that the receiving TCP is currently prepared to receive [3].
TCP uses two algorithms, "slow start" and "congestion avoidance" to adapt its window to the network state. In the following we assume that the window N is measured in segments.
Slow start. Starting at $N = 1$, slow-start is used to increase N by one segment for every incoming ACK until N reaches a threshold considered as an estimation of the network capacity. This exponential growth reduces the burstiness of the network traffic. The threshold window size for the first slow start is given a default value at the beginning of the connection. For subsequent slow-starts it is determined dynamically as a function of network parameters.
Congestion avoidance. After the threshold is reached, the source moves to congestion avoidance where N is increased by 1 segment for every window worth of acknowledged packets, hence every round trip time T. This slower growth aims to probe the network for extra bandwidth and it continues until a segment is lost.
Fast retransmit and fast recovery. The loss is detected either by timeout or by two or more consecutive ACKs carrying the same sequence number. Berkeley-derived implementations count the number

of duplicate ACKs received and when the third one is received, assume that the segment has been lost and retransmit only one segment starting with the sequence number in the ACK [4].
When detecting a segment loss the source assumes that the network is congested and resets the window size to $N/2$; it considers half the window at which the loss is detected as a good estimate of available capacity. What happens next depends on the version of TCP. TCP Tahoe assumes that the network is congested, resets N to 1 and resorts to slow start. TCP Reno version will resort to slow start if the loss is indicated by timeout; if the loss is detected by multiple acknowledgements it starts a new congestion avoidance phase after recovering from the losses. TCP Reno considers multiple acknowledgements as an indication of light congestion [4, 11].
TCP uses slow start in three different ways: to start a new connection, to restart a transmission after a retransmission timeout and to restart a transmission after a long idle period.

4 Using TCP over satellite links

4.1 Theoretical performance limits

In what follows we shall use a formula from queuing theory known as Little's result [17, 18]. Consider a general queuing system on figure 1. If a system is in a steady state, Little's result relates the average number of customers in a system N to the average arrival rate λ and the average time spent in that system T, namely,

$$N = \lambda T. \tag{1}$$

Suppose that T is fixed. Two consequences of (1) are immediate in a network environment. First,

$$N \leq \max(\lambda)T. \tag{2}$$

In a point-to-point communication $\max(\lambda)$ is the bandwidth B of the communication link and T is the round trip time on that link. The bandwidth delay product BT is an important characteristic of network environment.
Second,

$$\lambda \leq \max(N)/T, \tag{3}$$

which means that if the time in the system T is fixed, the maximum throughput is determined by the maximum window size.
Equations (1-3) govern the behavior of transmission protocols. For example, the window field in TCP header is 16 bits, which means that the maximum window size that can be advertized by the receiver is 2^{16} bytes. By Eq. (3) $\lambda \leq 2^{16}/T$. On a GEO satellite link $T = 0.5$s. It follows that the maximum throughput that can be achieved on such link is 1 MBytes/s. To mitigate this problem RFC 1323 [5] defines a window scale option which increases the definition of TCP window from 16 to 32 bits. The header still holds a 16 bit value, and an option is defined that scales the 16 bit value. The maximum window size is 2^{30}, which gives maximum throughput of 2 GBytes/s over GEO satellite link.
As another example, consider a situation in which the lastly received segment is lost, or corrupted. It takes one round trip time T for the receiver to get a copy of a lost segment. During that time at most BT new bits may arrive through communication link. Before the lost segment arrives those bits have to be queued because TCP delivers data in the sent order. It follows that to prevent data loss at the receiving TCP its buffer should be larger than the bandwidth delay product BT.

4.2 Difficulties caused by TCP design

Slow start. There are several problems with slow start algorithm on high-speed networks. First, it can take quite a long time to get up to speed. For example, on a Gb/s GEO satellite link with $T = 0.5$s, it takes 29 round-trip times to or 14.5s to finish start up [1]. If the link is otherwise idle, during that period most of the bandwidth will be unused. The entire transfer may finish before the full link speed is reached. The user will never experience the full link's bandwidth. All the transfer time will be spent in slow start.
Double slow start. Second, during slow start TCP sends larger and larger bursts of data into the network. If the threshold window size at which TCP should change from slow start to congestion avoidance is too large, the buffers in the receiver or the network will overflow before getting in congestion avoidance. Multiple losses force TCP to reduce its window and to begin a new slow-start with threshold that is less than the network capacity estimate. This results in low average throughput. One possible solution to this problem is to probe the network in order to obtain a better initial value of the threshold.
Data loss. Third, TCP assumes that virtually all packet loss is caused by network congestion. Consequently, TCP invokes congestion control and reduces its transmission rate as a result of any packet loss. This response is inappropriate when loss is due to corruption rather than congestion. In case of corruption you should transmit more, not less. However, in case of congestion transmitting more is like pouring gasoline on the fire. The fast retransmit algorithm in TCP Reno detects loss, and thus avoids slow start,

by means of duplicate acknowledgments. I think that negative acknowledgments would be a better solution to the problem.
With cumulative ACKs and fast retransmit TCP can recover efficiently from a single loss per window. However, because new data must be received for the receiver to advance the ACK number, TCP requires a minimum of one round trip time T to signal each additional hole in the out-of-sequence queue. To deal with this problem a Selective ACK (SACK) mechanism is defined in RFC 2018. SACKs generated at the receiver contain a list of correctly received data blocks. The sender gets explicit information on which segments have arrived and which may have been lost. Thus the sender has more information about which segments may need to be retransmitted.
Rollover of segment counter. On high speed connections the 32 bit sequence number may roll over from 2^{32} to 0 during TCP connection's life time. This may cause old segments to reappear and contain sequence numbers currently being transmitted. To prevent this problem RFC 1323 introduced the Protection Against Wrapped Sequence numbers (PAWS) algorithms. With PAWS the sender time-stamps each segment. The receiver uses segment number concatenated with the time stamp as the segment identifier. Even if the segment number rolls over, the old and the new segments will be distinguished by the segment's time stamp.
Asymmetric channels. As mentioned in the Introduction, the satellite and the space communication may be characterized by asymmetric up and down channels. The asymmetry may be in terms of bandwidth and error rate. The bandwidth ratio of the down to the up channel can be as high as 1000:1 [2]. A receiving TCP generally acknowledges every other segment, which dictates and ACKs channel capacity proportional to the data channel capacity and is a function of the segment size.
Here is a rough calculation: Assume segment size of 10^3 bytes. The size of the TCP header is 20 bytes and each header contains a single ACK;

$$10^3 : 20 = 50 : 1.$$

It follows that with a 10^3 bytes segment size, TCP throughput is not affected by the bandwidth ratio of the forward and the return channels when that ratio is less than $50 : 1$. At higher ratios throughput is limited by the ACKs channel capacity.
One way to deal with this problem is reduce the amount of acknowledgments generated by the receiving TCP. However, acknowledgments are so central to the operation of the TCP that a big change in the ACKs mechanism will result in a different protocol. Another way to cope with the limited link capacity is to compress TCP header. A differential compression scheme for TCP headers is defined in RFC 1144 [8]. The compression is done on IP level, hop-by-hop. Initially, the header is saved. Subsequently, only the changes to the previous header are sent to the other end. If the segment is lost or corrupted, the invalid TCP header will be detected when the TCP checksum fails. Retransmissions are sent uncompressed to facilitate synchronization. The header compression scheme of RFC 1144 is criticized by R. Durst et al in [2] because differential compression is not suitable for environments in which synchronization is expensive.

5 Improving system performance

Custom applications. An application-level approach to improve system performance is shown on figure 4. The source and the destination applications establish n TCP connections which allows them to send n times more data during TCP slow start and recover more quickly from congestion avoidance. In [10] M. Allmann et al describe XFTP, an application for bulk data transfer that is based on FTP and that splits the data among multiple TCP connections. XFTP performed better than FTP over satellite links. The disadvantage of this approach is that one has to build custom application; the data must be split between the TCP channels at the source and merged at the destination. Another concern with allocating multiple TCP channels is that, similarly to increasing the initial window size, it will increase the overall burstiness of the network which will result in more packet losses [14].
Splitting the control loop. A well known way to get rid of problems concerning long delay in a control loop is to split it into several small loops. A similar idea has been proposed in the Internet Draft "Ongoing TCP Research Related to Satellites" [14]. The satellite link is isolated from the rest of the Internet using two routers: a virtual destination and a virtual source (see figure 2). The TCP sender first transmits to the virtual destination that will take care of acknowledgments and retransmissions. The router acknowledges the data to the source as if the TCP receiver has got the data; it sends then the data through the satellite link. At the other end the virtual source transmits to the receiver (it needs to queue the outstanding segments). The routers may use a custom protocol to communicate over the satellite link. The advantage of this configuration is that it is transparent to the sending and receiving TCP. The disadvantage is that it breaks the end-to-end semantics associated with TCP.
A variation on this idea is the Snoop protocol [16]

that provides performance improvements on wireless links by introducing an agent at the base station that monitors the passing TCP traffic and cashes packets until they are acknowledged. The agent is able to detect lost segments through the receipt of duplicate acks or by a local time-out. When the agent detects a packet loss in the wireless link, it retransmits the lost packet and intercepts the duplicate acknowledgments to prevent the TCP sender from invoking congestion control. In effect, the agent covers for losses on the wireless link transparently to the sending TCP.
I think that the problem of long delay in space communications will be solved similarly. The space probe which may be millions of kilometers away will communicate with Earth through a chain of relay stations (see figure 3).

6 Conclusion

There are many similarities between the space and the mobile communication environments when observed from the perspective of transport protocol. Also, in both environments, power, weight and physical volume of equipment are scarce resources. Similar problems lead to similar solutions.
Satellite links are characterized by a high bandwidth delay product. TCP under-performs in high bandwidth delay environments were end-to-end synchronization is expensive. The performance of the TCP can be improved substantially if the sending and the receiving TCP implementations contain the features described in RFC 1323 [5].
Since TCP assumes very little from the network it works (adequately) in many different environments. The design choices of a protocol made for for space communications would be different from those of TCP. For example, I'd use rate-based (token bucket) rather than window-based congestion control, explicit signaling of link outage and negative, rather than positive acknowledgments.

References

[1] Partridge and T. Shepard. Tcp/ip performance over satellite links. *IEEE Network*, 17(7):44–51, September 1997.

[2] R. C. Durst, G. J. Miller, and E. J. Travis. TCP extensions for space communications. In *Proceedings of ACM MOBICOM '96*, pages 15–26, 1996.

[3] Jon Postel, Editor. Transmission Control Protocol — DARPA Internet Program Protocol Specification. RFC 793, September 1981.

[4] W. Richard Stevens. *TCP/IP Illustrated*, volume 1. Addison Wesley, 1994.

[5] V. Jacobson, R. Braden, and D. Borman. RFC 1323: TCP extensions for high performance, May 1992. Obsoletes RFC1072, RFC1185 [6, 19]. Status: PROPOSED STANDARD.

[6] V. Jacobson and R. Braden. TCP extensions for long-delay paths; RFC 1072. *Internet Request for Comments*, (1072), October 1988.

[7] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. RFC 2018: TCP selective acknowledgment options, October 1996. Status: PROPOSED STANDARD.

[8] V. Jacobson. RFC 1144: Compressing TCP/IP headers for low-speed serial links, February 1990. Status: PROPOSED STANDARD.

[9] T. V. Lakshman, U. Mahdow, and Bernhard Suter. Window-based error recovery and flow control with a slow acknowledgement channel: a study of TCP/IP performance. In *Proceedings of INFOCOM 97, Kobe, Japan*, 1997.

[10] M. Allman, C. Hayes, H. Kruse, and S. Ostermann. TCP performance over satellite links. In *Proceedings of the Fifth Inernational Conference on Telecommunications Systems, Nashville, TN, March 1997*, pages 456–462, 1997.

[11] Chadi Barakat and Eitan Altman. Analysis of TCP in networks with small buffering capacity and large bandwidth-delay product. Technical Report RR-3574, Inria, Institut National de Recherche en Informatique et en Automatique, 1998.

[12] Nesrine Chaher, Chadi Barakat, Walid Dabbous, and Eitan Altman. Improving TCP/IP over geostationary satellite links. Technical Report RR-3573, Inria, Institut National de Recherche en Informatique et en Automatique, 1998.

[13] M. Allman, D. Glover, and L. Sanchez. RFC 2488: Enhancing TCP over satellite channels using standard mechanisms, January 1999. See also BCP0028. Status: Best Current Practice.

[14] M. Allman (Editor). Ongoing TCP research related to satellites. Internet draft draft-ietf-tcpsat-res-issues-05.txt., November 1998.

[15] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. In *Proceedings of the ACM SIGCOMM*

Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, volume 26,4 of *ACM SIGCOMM Computer Communication Review*, pages 256–269, New York, August 26–30 1996. ACM Press.

[16] H. Balakrishnan, S. Seshan, and R. H. Katz. Improving reliable transport and handoff in cellular wireless networks. In *ACM Wireless Networks*, volume 1,4, pages 469–481, New York, December 1995. ACM Press.

[17] J. D. C. Little. A proof for the queueing formula $L = \lambda W$. *Oper. res.*, pages 383–387, 1961.

[18] L. Kleinrock. *Queueing Systems. Volume I: Theory.* John Wiley & Sons, New York, 1975.

[19] V. Jacobson, R. T. Braden, and L. Zhang. RFC 1185: TCP extension for high-speed paths, October 1990. Obsoleted by RFC1323 [5]. Status: EXPERIMENTAL.

7 Figures

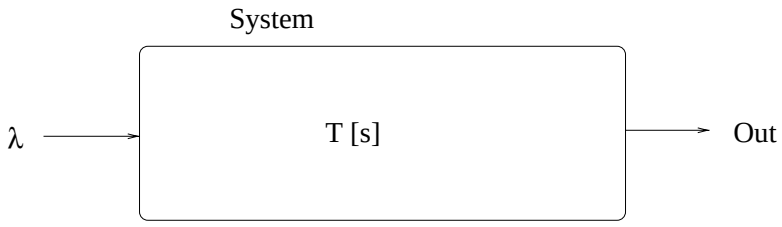

Figure 1: A general queueing system. Little's result states that in the steady state the average number of customers in the system N equals the arrival rate λ times the average time spent in the system T.

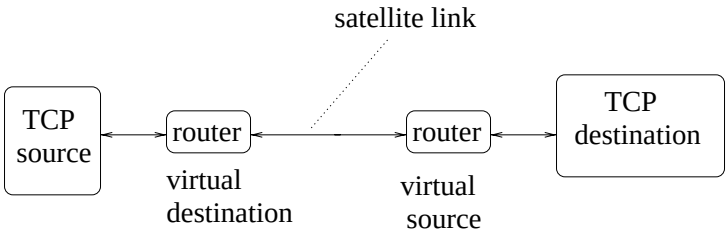

Figure 2: Isolating the satellite link by inserting virtual destination and virtual source.

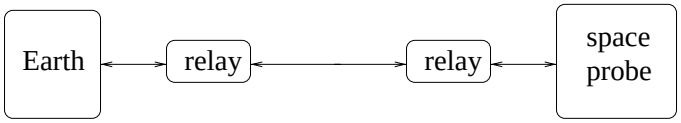

Figure 3: Shortening the control loop by inserting relay stations.

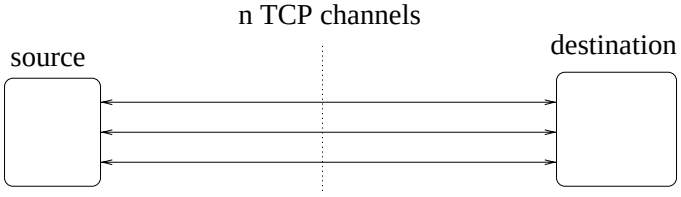

Figure 4: Application-level approach to improve system performance.