Seminar ``Broadband Networking Technology''

TCP over Satellite Links

Andreas Schilke
TU Berlin
June 5, 1997

Under the guidance of: Morten Schläger

Abstract

If standard TCP is used over satellite links, some effects should be taken into consideration. First, satellites often provide connections with a high bandwidth, possibly leading to a wrap around of sequence numbers. Second, links over geostationary satellites have a large delay. And third, TCP can get in trouble with the TCP assumption that every loss of segments is caused by congestion. These problems will be discussed and possible solutions are shown. At the end some results of measurements, carried out at Ohio University, are given.

Contents

1. Overview

In this paper TCP and the effects if used over satellite links are examined. Section 2 summarizes what TCP is used for and what problems occur with satellite links. In section 3 the possibility of wrap around will be discussed. This is followed by problems caused by a large delay in section 4

and possible solutions in Section 5. Then, the loss assumption of TCP, related problems and solutions are mentioned in section 6. In section 7 some experiments, carried out at Ohio University, are shown. Section 8 provides conclusions and closing remarks.

2. Standard TCP

The TCP protocol is a general-purpose, reliable stream protocol. It works in different network environments, independent of medium, transmission rate, delay, and error rate. That is why it is used for many applications, e.g.:

- file transfer (FTP)
- remote login (TELNET)
- email (SMTP)
- news (NNTP)
- WWW (HTTP)

It also works correctly over satellite links, yet there are **problems** regarding throughput.

First, there are **large delays** over satellite links. For example, a delay of approximately 280 ms was measured over the NASA Advanced Communications Technology Satellite (ACTS) [7]. In section 4 and 5 this will be explained more detailed.

The second problem is the **assumption** of the TCP protocol that every **loss is caused by router congestion**, also in case of link errors. As reaction TCP slows down. How to handle this problem will be the object of section 6.

Both cases limit the maximum throughput. But there are also effects on reliability if satellite links with a high bandwidth are used.

3. Wrap-Around and Timestamps

In the context of satellite links **high transfer rates** may lead to a **reuse of sequence numbers**. This might happen by:

- an **earlier incarnation** of the connection (this is avoided by enforcing the maximum segment lifetime (MSL) which is fixed to 2 mins in TCP)
- a **sequence number wrap-around**

Two examples for $T_{wrap} = 2^{31}/\text{bandwidth}$ (cf. [4]) will give an idea of the problem:

- Ethernet (10Mbps): T_{wrap} = 1700 secs (approx. 30 mins)
- Gigabit (1 Gbps): T_{wrap} = 17 secs

A value of 30 mins should not be a problem but 17 secs are by far less than the MSL. To work reliable, the MSL would have to be bound to the transfer rate, which results in a new mechanism.

Because this would be a deep cut into the existing protocol, another solution is used: **timestamps** and **protect against wrapped sequence numbers** (PAWS) mechanism [4].

The timestamps are defined as TCP option and allow the measurement of RTT and PAWS. They use a granularity of approx. 1 ms to 1 sec. With the 32-bit timestamp the sign bit will wrap around in 24.8 days (granularity of 1 ms), which is assumed to be save.

4. Delay and Window Size

As mentioned above, one of the limiting factors for throughput is the delay of a connection. This section will give the reasons.

TCP is designed as a sliding window protocol. Hence, a fixed maximum amount of data (``receive window size") is on the link between sender and receiver as shown in figure 1.

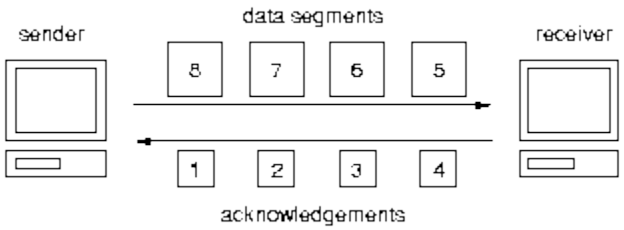

Figure 1: Data flow between sender and receiver

As a consequence the maximum throughput is limited by the Round Trip Time (RTT) of this link:

$$\text{throughput}_{max} = \frac{\text{receive buffer size}}{\text{round trip time}}$$

If this equation is rearranged to

$$\text{round trip time} \times \text{throughput} <= \text{receive buffer size}$$

one will get the condition for the ``delay bandwidth product" (DBP) that is often used to characterize connections.

Examples:

For a **terrestrial connection** with 8 KBytes window size and 10 ms RTT this leads to

$$\text{throughput}_{max} = 8 \text{ KBytes} \sim 800{,}000 \text{ Bytes}$$

$$\frac{}{10\ \text{ms}} \qquad \frac{}{\text{s}}$$

and for a **satellite connection** with 64 KBytes window size (maximum in standard TCP) and 560 ms RTT follows:

$$\text{throughput}_{max} = \frac{64\ \text{KBytes}}{560\ \text{ms}} \sim 115{,}000\ \frac{\text{Bytes}}{\text{s}}$$

As a result, at least a 100 KByte window is needed to fully utilize a T1-link (1.544 Mbps) with 560 ms RTT.

5. Solutions to the RTT Problem

To overcome the RTT problem the following solutions are possible:

- The straightforward solution is to **reduce the RTT** itself. This would mean to lower the satellites, e.g. by using low-Earth-orbit (LEO) satellites in an altitude of approx. 750 km instead of geostationary-Earth-orbit (GEO) satellites at approx. 36,000 km (for example see [10]).
- Another possibility are **multiple connections in parallel**, e.g. XFTP. This is a modified version of FTP developed at Ohio University (for details see [2]). It works with a user defined number of parallel connections and uses an extension to FTP to request multiple connections.
- The last given solution, discussed in the next section, is to **increase the window size** which results in changes to the TCP protocol.

5.1 Scaled Windows

The scaled window option of TCP is defined in RFC 1323 [3] (there is also a new internet draft [4] in progress). It **expands** the definition of the **TCP window** from 16 **to 32 bits**. Therefore the 16-bit window size field of the TCP header and a **scale factor** are used. This scale factor is carried in a new TCP option. It is limited to a power of two. Because it is sent in a SYN segment only, the window scale is fixed when the connection is opened. Using this extension, window sizes of up to 1 GByte are allowed. This is assumed to be big enough to fully utilize connections also for large delays and high bandwidth.

6. Solutions to the loss assumption

As mentioned in section 2, if a sender encounters an error in delivery, it assumes that it is caused by congestion and slows down. Usually this is true for copper or fiber based networks. But it need not to be true for radio connections as satellite, because of their higher error rate. In the following section it is shown why connections suffer from slow start and congestion avoidance algorithm.

6.1 Slow Start and Congestion Avoidance

The slow start and congestion avoidance algorithm is described in RFC 2001 [9]. Figure 2 gives an overview of its behaviour.

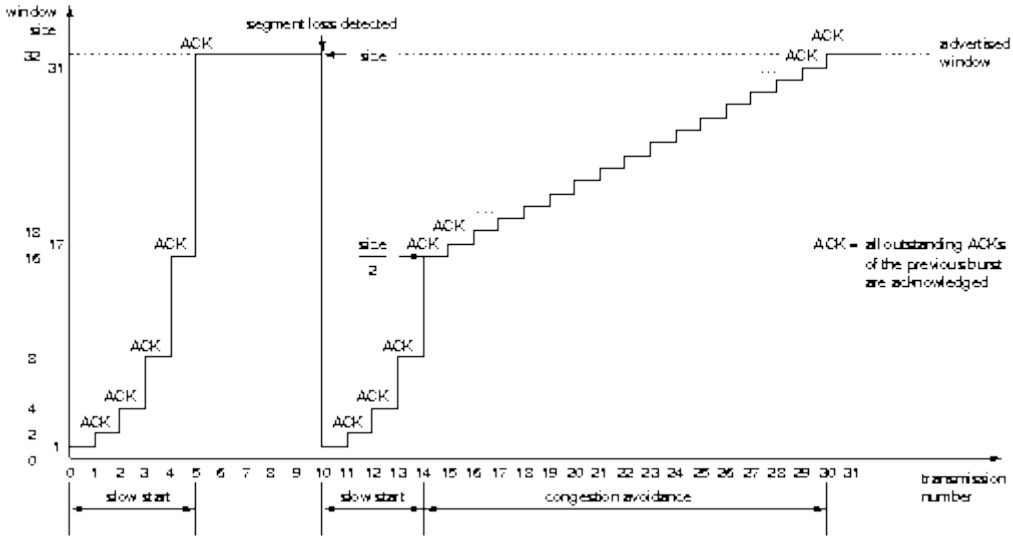

Figure 2: Slow Start and Congestion Avoidance (cf. [8], window size given in multiples of MSS)

Assumed sender and receiver advertised a window size of 32 segments. Now the sender starts transmission with the first segment. If the receiver's acknowledgement comes back a burst of two packets is sent. Starting from now, with the acknowledgement of all packets of a burst, the number of packets sent in a burst is doubled until the advertised window size is reached.

The sender transmits now with the maximum throughput under these circumstances.

If a segment loss is detected the slow start algorithm is started again but with a small change. If half of the former window size is reached, congestion avoidance starts. This means that with every acknowledged burst the window size is increased only by 1 segment instead of doubling it. That is continued this way until the advertised window size is reached.

As an example one gets the following: at 500 ms RTT and 512 Byte maximum segment size (MSS) it takes

- approx. 4 s to slow start up to a 100 KByte window
- approx. 50 s to go from a 50 KByte to a 100 KByte window

This **seriously effects the throughput on short connections** (e.g. HTTP) because only the beginning of the slow start phase is passed. For this a possible solution is in discussion: the initial window size could be increased (e.g. to 4 or 8 segments [6]).

6.2 Fast Retransmit and Fast Recovery

Fast retransmit and fast recovery algorithms as described in RFC 2001 [9] give a higher throughput for long connections in case of errors. The mechanism is shown in figure 3.

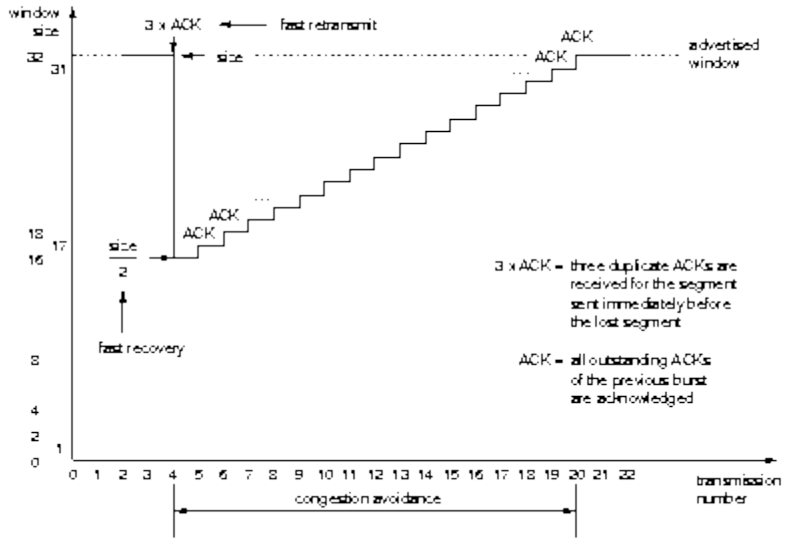

Figure 3: Fast Retransmit and Fast Recovery (window size given in multiples of MSS)

With fast retransmit, the time it takes a sender to recognize a lost segment, is reduced. Rather than waiting for a retransmit timeout, the sender can retransmit if three duplicate acknowledgements of the last passed segment are received.

If these three duplicate acknowledgements are received fast recovery starts. Instead of falling back to slow start, the window size is halved and the congestion avoidance phase begins immediately.

Using this algorithm, long connections get a higher throughput, hence there is an active loss detection and the slow start phase can be left out.

6.3 Selective Acknowledgement

The description of this algorithm can be found in RFC 2018 [5]. It is introduced for a better performance when multiple packets are lost from one window of data. If it is used only lost segments need to be retransmitted. It uses two TCP options:

- enabling option (SACK-permitted), which may be sent by the sender in a SYN segment to indicate that the SACK option can be used for this connection
- the SACK option itself, which may be sent by the receiver

With a SACK the sender is informed of non-contiguous blocks that have been received and queued. Therefore the left and right edge of a contiguous block are announced by their 32 bit sequence number. Here a maximum of 4 blocks can be specified (40 Bytes available for TCP options, 2 Bytes used for kind and length of option). It is stipulated that the SACK option always reports the block containing the most recently received segment. Say a (new) ``SACKed" flag bit

for each segment in the retransmission queue is created at the sender. If a SACK arrives, the accompanying bits are set and the sender will skip these segments. If a retransmit timeout occurs this knowledge is lost, hence the sender should turn off all of the SACKed bits in this case. This is important, since the timeout might indicate that the data receiver has reneged.

7. Experimental Results

All the experiments described below were carried out at Ohio University using NASA's ACTS. The following setup was used:

- half T1 link
- window size of 56 KByte for TCP Reno (that is with slow start, congestion avoidance, fast retransmit, fast recovery and large windows) and 14 KByte for each of the 4 XFTP connections (giving the same effective window size)
- packet size of 512 Bytes

Tests were undertaken with 200 KBytes, 1 MBytes, and 5 MBytes file transfers. To force congestion the window size of the TCP connection was increased (so it is a little larger than buffering on the routers).

As a clue for the possible throughput the channel bandwidth and TCP limit (which includes calculation of the protocol overhead) are given.

First, the result of the experiment without congestion is shown.

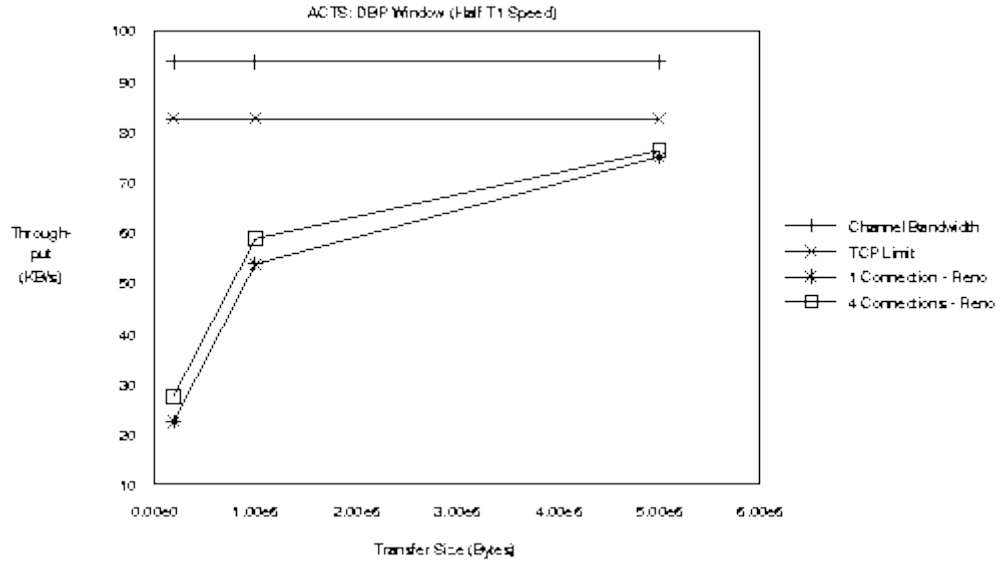

Figure 4: Tests over ACTS Links without congestion (cf. [1])

Here, the throughput for small files suffers from the influence of slow start. The performance of XFTP is slightly better than for standard TCP since the effective window size is bigger in the

slow start period. Larger files spend less time on slow start and so their average throughput is much better. In this case TCP Reno achieved approx. 90% of the optimum throughput.

The next figure shows the results with congestion.

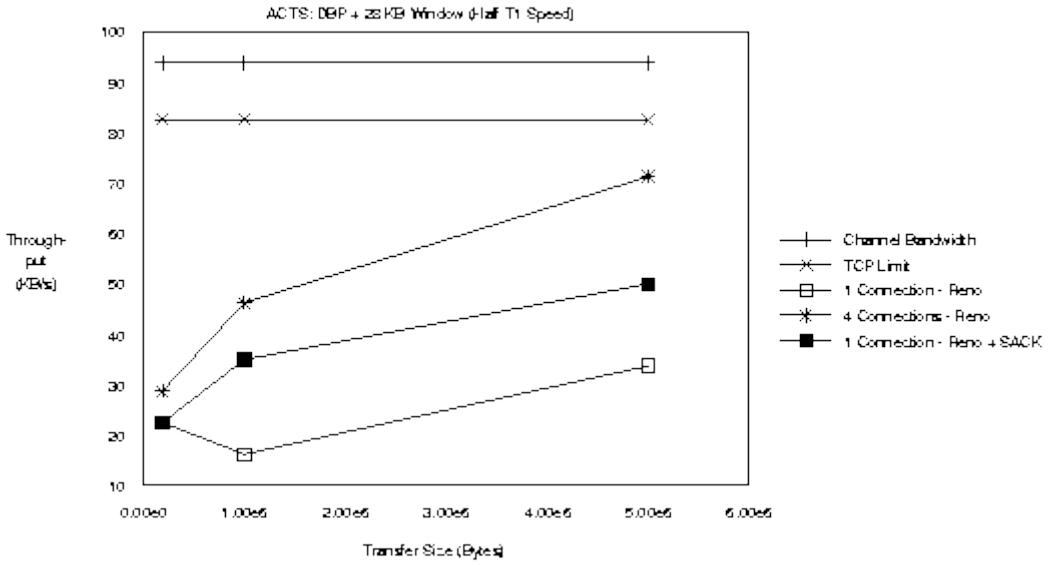

Figure 5: Tests over ACTS Links with congestion (cf. [1])

For the tests in figure 5 it can be seen that short files are not able to cause congestion (note that this is an isolated network!). So the results are the same as above. For larger files TCP Reno gives a relatively poor performance. But with TCP SACK (61% of opt.) and XFTP (86% of opt.) the files with a size of 5 MBytes achieve a good performance also in case of losses. It should be noted that XFTP does not use selective acknowledgements explicitely but by spreading the loss over multiple connections it has the same effect.

It should be mentioned that the measurements were limited by both testing time and bandwidth (half T1 link) which did not allow for comprehensive testing. Hence, the same tests were repeated with a hardware emulator to compare the results. They were quite close together, so additional tests were only made with the emulator.

8. Conclusion

Conclusively, there is a strongly limited throughput if only slow start and congestion avoidance, also in conjunction with fast retransmit and fast recovery, are used. Large windows are recommended to use the full channel bandwidth. If there are congestion and higher bit error rates, SACKs can improve performance, too. There is a limiting effect of slow start and congestion avoidance but changes must be taken with caution because it is considered to be essential to well-behaved TCP implementations.

References

[1]
M. Allman, C. Hayes, H. Kruse, S. Ostermann. TCP Performance over Satellite Links. Ohio University, 1997 - http://jarok.cs.ohiou.edu/papers/nash97.ps

[2]
M. Allman, S. Ostermann. Multiple Data Connection FTP Extensions. Ohio University TR-19971, 1997 - http://jarok.cs.ohiou.edu/papers/xftp.ps

[3]
V. Jacobson, R. Braden, D. Borman. TCP Extensions for High Performance. RFC 1323, 1992 - ftp://ds.internic.net/rfc/rfc1323.txt

[4]
V. Jacobson, R. Braden, D. Borman. TCP Extensions for High Performance. Internet-Draft, 1997 - ftp://ds.internic.net/internet-drafts/draft-ietf-tcplw-high-performance-00.txt

[5]
M. Mathis, J. Mahdavi, S. Floyd, A. Romanow. TCP Selective Acknowledgment Options. RFC 2018, 1996 - ftp://ds.internic.net/rfc/rfc2018.txt

[6]
S. Ostermann. TCP Over Satellite BOF, Briefing Two Phased Solutions. Presentation 38th IETF, 1997 - http://jarok.cs.ohiou.edu/papers/presentations/memphis-tcp.ps

[7]
S. Ostermann, M. Allman, and H. Kruse. An Application-Level Solution to TCP's Satellite Inefficiencies. Presentation WOSBIS-96, 1996 - http://jarok.cs.ohiou.edu/papers/presentations/wosbis_talk.ps

[8]
A. S. Tanenbaum. Computer Networks. 3rd ed., Prentice-Hall, pp. 536-539, 1996.

[9]
W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. RFC 2001, 1997 - ftp://ds.internic.net/rfc/rfc2001.txt

[10]
The Importance of Latency in Telecommunications Network. Teledesic Corp. - http://www.teledesic.com/overview/latency.html

For current development notes and a description of how to subscribe to the ``TCP-over-satellite'' mailing list, see the homepage of the ``Internet Protocols over Satellite (IPoS) Working Group'' at http://www.isr.umd.edu/CSHCN/Links/IPoS/homepage.html.