TCP Performance over Satellite Links*

Mark Allman, Chris Hayes, Hans Kruse, Shawn Ostermann
Ohio University
{mallman,chayes,ostermann}@cs.ohiou.edu
hkruse1@ohiou.edu

Abstract

Over the past few years, we have reported on the performance issues faced by TCP/IP based applications on satellite links. Performance is limited by the delay inherent in geosynchronous systems and the probability of bit errors found in any wireless system, including satellite systems. These limitations are becoming more important as new satellite systems offer much higher data transmission rates than those available in the past.

While high efficiency on fast satellite links may eventually require new protocol modifications, our previous studies indicate that full link utilization may be achievable using the TCP performance enhancements that have already been approved or that are currently in the standards process. Of particular interest in the satellite environment are performance enhancements for scaled windows and timestamps (RFC 1323), fast retransmit and recovery (RFC 2001), and selective acknowledgement (RFC 2018). The paper describes each of these performance enhancements and explains how they can increase TCP's performance over satellite links.

We present the results of a comprehensive performance study of these TCP protocol enhancements in the satellite environment over error free links. We show results over networks both with and without congestion loss. Finally, we examine TCP's performance on realistic satellite links with a non-zero probability of bit errors, where the effectiveness of selective acknowledgments is examined and compared to the performance of traditional TCP implementations.

1 Introduction

In several experiments using NASA's Advanced Communications Technology Satellite (ACTS), investigators have reported[Kru95] disappointing throughput using the TCP/IP[Com95, Pos81, Ste94] protocol suite over 1.536 Mbit/second (T1) satellite circuits.

*This work is sponsored in part by a grant from the NASA Lewis Research Center

From the 5th Int'l. Conference on Telecommunication Systems – 1997

A detailed analysis of FTP file transfers reveals that both the TCP receiver window size, and the TCP *slow start* and *congestion control* algorithms contribute to the observed limits in throughput. Furthermore, in the face of loss, TCP's data recovery (positive acknowledgment) mechanism works poorly over long-delay channels.

Several mechanisms have been introduced and standardized that will aid TCP performance in long-delay environments. The window size limitation in TCP was addressed in RFC 1323[JBB92]. RFC 2001[Ste97] formalizes the slow start, congestion control, fast retransmit and fast recovery algorithms previously widely implemented. A selective acknowledgement mechanism was defined in RFC 2018[MMFR96].

Section 2 summarizes the TCP variants that we tested. Section 3 describes the testing environment that we used to conduct the experiments. Section 4 shows the results of our experiments, and section 5 provides some closing remarks and directions for future work.

2 TCP Protocol Variants

All of the experiments reported in this paper were conducted using FTP client and server applications running on top of variants of TCP. The computer endpoints used in these tests were two Intel PC's running NetBSD 1.1 (based on 4.4 BSD Unix). Versions of TCP are typically referred to in the literature by the names given to their releases as part of BSD Unix; for the experiments discussed in this paper, we used *TCP Reno* with slow start, congestion avoidance, fast retransmit, fast recovery, and large windows, as described below.

Slow Start and Congestion Avoidance

The slow start and congestion avoidance mechanisms were introduced in 1988 in Jacobson[Jac88] and added as a requirement for TCP implementations in 1989[Bra89]. Slow start is used to gradually increase the rate at which the sender injects data into the network. Slow start begins by sending one segment and waiting for an acknowledgement. For each acknowledgement the sender receives, it injects two segments into the network; leading to an exponential increase in the amount of data being sent. Slow start ends when the receiver's advertised window is reached or when loss is detected. Because the amount of time required for slow start to achieve full bandwidth is a function of round trip time, satellite links are particularly sensitive to the limited throughput available during slow start.

Congestion avoidance is used to probe the network for available bandwidth by sending one additional segment for each round trip time (up to the receivers advertised window). In the original slow start/congestion avoidance scheme, when the sending TCP detects segment loss (indicating congestion), it drops back into slow start until the packet sending rate is one half the rate at which the loss was detected and then begins the congestion avoidance phase.

Fast Retransmit and Fast Recovery

TCP Reno also incorporates *Fast Retransmit and Recovery*[Ste97]. Although these

mechanisms have been found in many Unix variants of TCP for several years, they weren't documented as a *Standards Track* RFC until Stevens[Ste97] in 1997. Fast retransmit reduces the time it takes a TCP sender to detect a single dropped segment. Rather than waiting for the retransmit timeout (RTO), the TCP sender can retransmit a segment if it receives three duplicate ACKs for the segment sent immediately before the lost segment.

Fast recovery works hand in hand with fast retransmit. As mentioned above, when a sender retransmits a segment, it normally recovers by moving first into a slow start phase followed by a congestion avoidance phase. If the sending TCP detects the segment loss using fast retransmit, however, *fast recovery* is used instead. Fast recovery halves the segment sending rate and begins congestion avoidance immediately, without falling back to slow start.

Large Windows

The original TCP standard limits the TCP receive window to 65535 bytes. TCP's receiving window size is particularly important in a satellite environment because the maximum throughput of a TCP connection is bounded by the round trip time[Pos81], as seen in the formula:

$$throughput_{max} = \frac{receive\ buffer\ size}{round\ trip\ time} \tag{1}$$

Without large windows, then, a TCP connection over a typical geosynchronous satellite is limited to throughput:

$$\begin{aligned} throughput_{max(satellite)} &= \frac{64Kbytes}{585ms} \\ &\approx 112,000\frac{bytes}{second} \approx 896,000\frac{bits}{second} \end{aligned} \tag{2}$$

Note that this upper bound on TCP throughput is independent of the bandwidth of the channel. A TCP connection running over a full T1 channel (1,536,000 bits/second) could still only achieve a maximum throughput of approximately 896,000 bits/second with a 65536 byte receive window. As specified in RFC 1323[JBB92], large windows (window scaling) can allow TCP to fully utilize higher bandwidth links over long-delay channels such as those found in satellite links.

Selective Acknowledgments

The cumulative positive acknowledgments employed by TCP are not particularly well suited to the long-delay satellite environment due to the time it takes to obtain information about segment loss. A selective acknowledgement (SACK) mechanism is defined in RFC 2018[MMFR96]. SACKs generated at the receiver explicitly inform

TCP Modification	Documented In	Standardization Status	Widely Available
Slow Start and Congestion Avoidance	RFC 2001[Ste97]	Required	Yes
Fast Retransmit and Fast Recovery	RFC 2001[Ste97]	Standards Track	Yes
Large Windows	RFC 1323[JBB92]	Proposed Standard	No
Selective ACKs	RFC 2018[MMFR96]	Standards Track	No

Table 1: Summary of TCP Performance Enhancements

the sender about which segments have arrived and which may have been lost, giving the sender more information about which segments might need to be retransmitted.

As a summary of the above information, the current status, documentation, and availability of the performance enhancements described above is shown in table 1.

3 Experimental Environment

All of the experiments reported in this paper measured the user-level data throughput over TCP between a pair of Intel 80486 computers running NetBSD 1.1 (which is derived from the BSD 4.4 source code). These two computers were each connected via a 10Mbps Ethernet to a Cisco 2514 router. Depending on which experiment was being run, the two routers were connected to each other via either a circuit over the NASA ACTS satellite or through a hardware satellite emulator.

3.1 NASA ACTS Satellite

Figure 1 shows the experimental setup used for the ACTS experiments. We connected a dual-interface, fractional T1 DSU/CSU to the T1 interface of the VSAT earth station. The first 12 channels were connected to one router, while the next 12 channels were connected to a second router. Connections were made via RS449 between the DSU and the routers. The earth station was then instructed to establish a satellite link from the first block of channels (1-12) to the cross-connect system in the satellite, and from the satellite back to the earth station (channels 13-24). Each channel was configured as a 64 kbps circuit. The result of this configuration was a 768 kbps full-duplex satellite link between the routers. An additional two 64 kbps channels were connected between the earth station and the Master Control station at NASA Lewis in Cleveland. These channels were instrumented for continuous, end-to-end bit error rate monitoring.

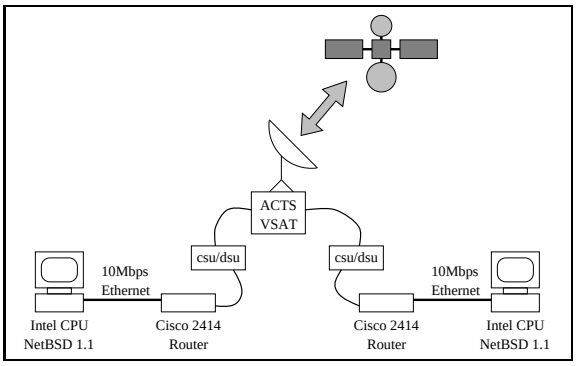

Figure 1: ACTS Experiment Setup
This figure shows the experimental setup used for the ACTS experiments reported in this paper.

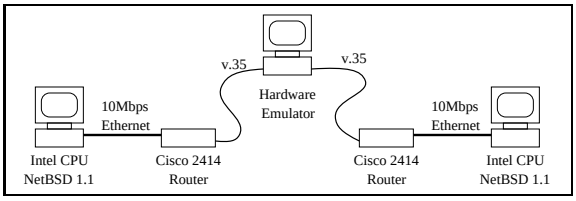

Figure 2: Hardware Emulator Setup
This figure shows the experimental setup used for the hardware emulation experiments reported in this paper.

3.2 Hardware Emulator

Figure 2 shows the experimental setup used for the experiments that used the hardware emulator. The emulator we used for these experiments is the "Data Link Simulator" made by Testlink Corporation. The hardware emulator can be configured to limit the amount of bandwidth between the routers. In the following experiments, the nominal bit rate was set to 1,536,000 bps (full T1) or 768,000 bps (half T1), depending on the test. In addition, the emulator can model a given amount of propagation delay. The propagation delay was set at 290 ms in each direction for the following tests, providing round trip times equivalent to those observed in the ACTS experiments.

4 Experimental Results

This section reports our results comparing the various TCP performance enhancements over both actual satellite links and emulated links. Each of the data points reported represents the average of at least 30 measurements obtained by timing a file transfer of a given size

using either FTP or XFTP. Experiments were run on isolated networks to eliminate the possibility of competing network traffic distorting the measurements. For the satellite tests, link errors rates were monitored before, during, and after each test to verify that link errors weren't distorting the data. Packets carrying TCP segments are limited to 512 bytes over the experimental network.

As part of our early research into TCP over satellite links, we developed and tested a modified version of FTP[PR85] called XFTP[AO96]. Using XFTP, the user can specify the number of parallel TCP connections that XFTP should use to transfer each file and the receive window size to use for each TCP connection. The XFTP client and server are based on the 4.4BSD Unix source code and have been compiled and run on various computer platforms supporting the Unix operating system. Several of the experiments below compared TCP variants against XFTP using an equivalent effective window size.

In each of the graphs below, in addition to plotting the experimental data, we also plot both the link speed and the maximum theoretical TCP data throughput rate for comparison. The maximum theoretical TCP data throughput rate takes into account the extra bytes lost to framing and header overhead and provides a better comparison against optimal throughput values than the link speed.

By analyzing the experimental data, we determined that two issues continue to limit TCP's large data file transfer efficiency over satellite links. The first problem is slow start. Assuming a 112 Kbyte window, used in some of the experiments below, TCP requires 11 round trip times to begin sending data at maximum speed. With long satellite round trip times, this speed increase requires approximately 6.5 seconds. For file transfers of relatively small files, the amount of time spent in slow start is shown to have a serious negative impact on overall throughput. For larger file sizes, the slow start penalty is amortized over a longer period of time and is less noticeable.

The second problem observed in our experiments is congestion avoidance. When a TCP that is sending at the full bandwidth of a T1 link, for example, detects a lost segment, it slows its sending rate from approximately 400 segments per second to approximately 200 segments per second (depending on how the loss was detected, see above). With the additive increase growth behavior of congestion avoidance, TCP then requires 200 round trip times (or almost 2 minutes) to return to its peak rate of 400 segments per second.

4.1 Results Over the ACTS Satellite - No Loss

Our first set of experiments tested the TCP variants over the ACTS experimental network described above in section 3.1. For this experiment, we tested both the standard TCP Reno (with a 56 Kbyte window) and XFTP using 4 connections yielding the same effective window size (4 connections, each with a 14 Kbyte window). We varied the size of the file being transferred and measured the throughput achieved. For each of these experiments, the queues in the routers were set above the *Delay Bandwidth Product* [PD96], meaning that there was enough buffering in the routers that no TCP segments were lost due to congestion.

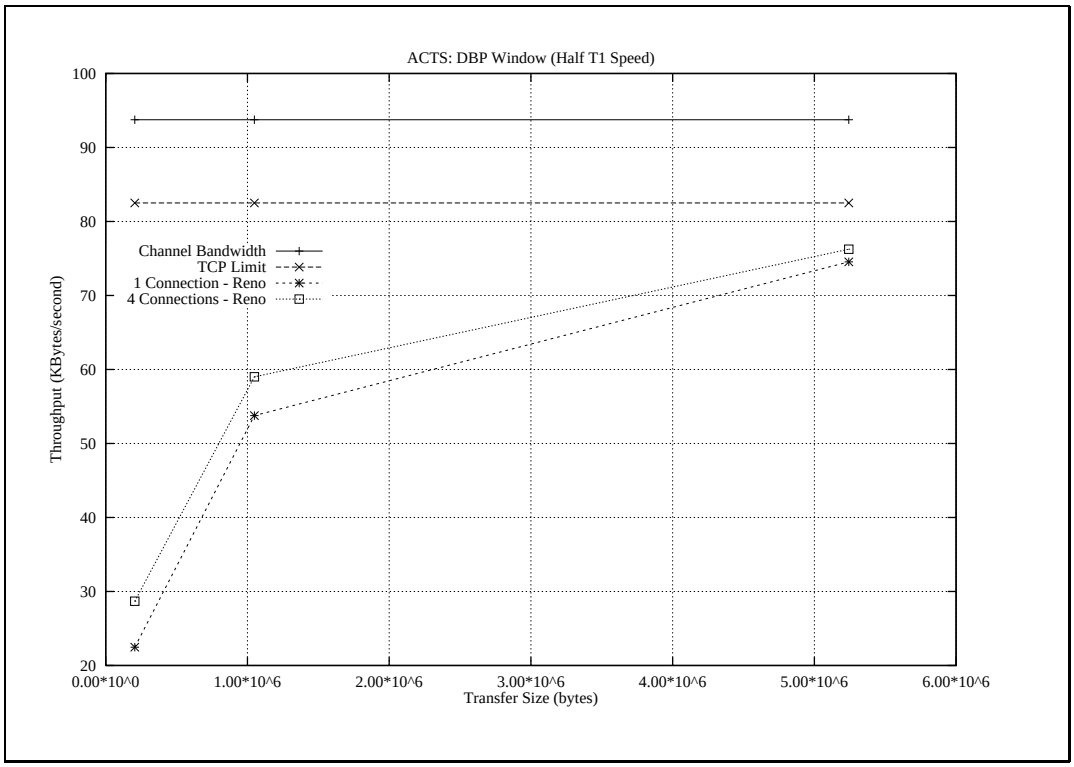

Figure 3: Tests over ACTS Links with Adequate Buffering
This graph compares the throughput achieved with standard TCP Reno with large windows against XFTP with the same effective window size. The link tested was one half T1 speed. Throughput values for transfers of files of size 200 Kbytes, 1 Mbyte, and 5 Mbytes are plotted.
[The figure included in the conference proceedings was incorrect. The figure above represents correct data]

Because selective acknowledgments are only useful in the presence of lost segments, we didn't include TCP SACK in these tests. Figure 3 summarizes the results.

In the absence of segment loss, TCP can achieve good throughput for long data transfers over satellite channels. As discussed above, the primary factor limiting TCP's performance (in the absence of loss) is slow start. Larger file transfers spend relatively less time in slow start resulting in better average throughput. Limited testing time with the ACTS satellite only allowed us to test files of size 5 Mbytes; even for 5 Mbyte files, TCP Reno achieved approximately 90% of optimal throughput.

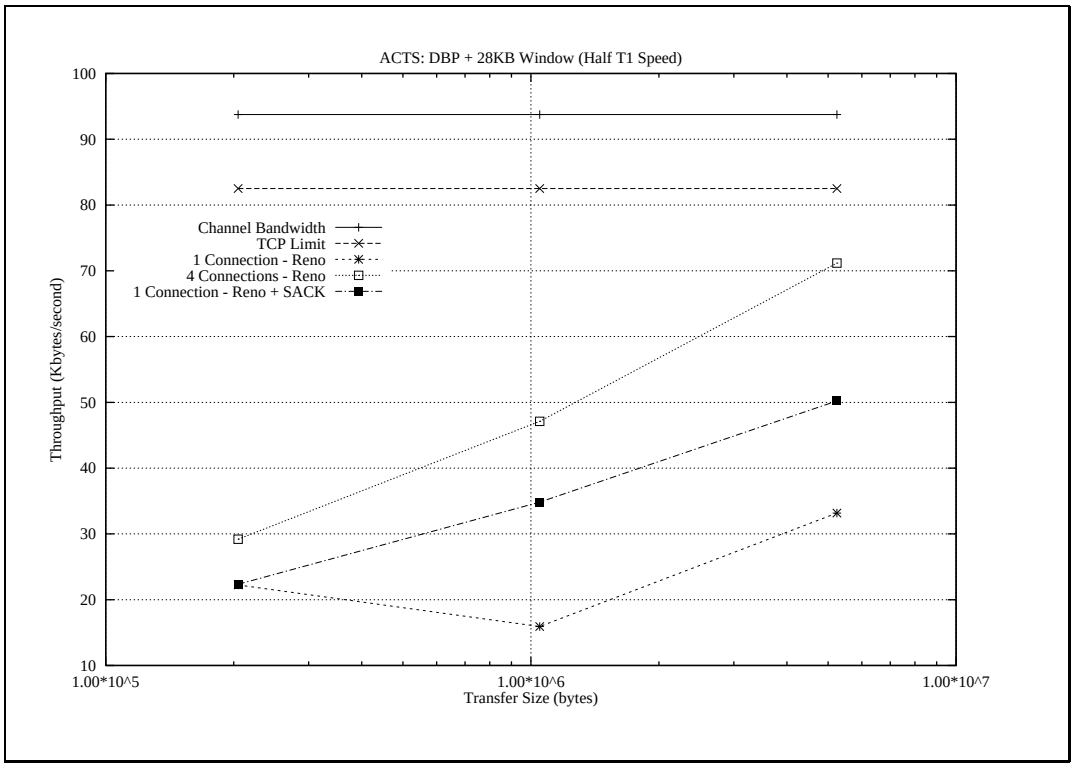

Figure 4: Tests over ACTS Links with Insufficient Buffering
This graph compares the throughput achieved with standard TCP Reno with large windows, XFTP with the same effective window size, and TCP SACK. The effective window size in each test was slightly larger than the buffering available in the routers to force segment loss due to congestion. The link tested was one half T1 speed. Throughput values for transfers of files of size 200 Kbytes, 1 Mbyte, and 5 Mbytes are plotted.

4.2 Results Over the ACTS Satellite - Congestion Loss

The first set of ACTS experiments, above, tested the TCP variants over a network with sufficient router buffering that congestion loss did not occur. In normal use, however, file transfers often incur occasional congestion loss on the networks between the two hosts. In the experiments below, the window size of the TCP connections was increased so that it was a little larger than the buffering available on the routers. Because TCP was able to inject more data into the network than the routers could buffer, congestion loss occurs in these tests. As in the previous experiments, we tested standard TCP Reno and XFTP. To test the effectiveness of selective acknowledgments in overcoming congestion loss in this environment, we also included experiments with the TCP SACK kernel in these experiments. The results are shown in figure 4.

For 200 Kbyte files, the results are similar to those seen in the previous graph. XFTP

with 4 connections performed slightly better than standard TCP because XFTP effectively sends more data during the initial slow start period. Because 200 Kbyte files are not large enough to cause congestion loss, TCP Reno and TCP SACK show equal throughput.

For 1 Mbyte files, TCP is able to send data more quickly than the routers can buffer and loss occurs, although the congestion event only causes the loss of a few segments. Without selective acknowledgments, standard TCP Reno suffers relatively poor performance. TCP SACK and XFTP (which effectively offers selective acknowledgments by spreading the loss across multiple connections) both perform much better than TCP Reno for 1 Mbyte files.

For 5 Mbyte files, the results are similar. A relatively larger file size amortizes the slow start over a longer period of time, yielding improved throughput. TCP SACK and XFTP are both able to overcome the occasional segment losses to achieve good performance, with TCP SACK achieving 61% of the optimal throughput and XFTP achieving 86% of the optimal throughput.

4.3 Comparing the Experimental Environments

As mentioned above, limited testing time with the ACTS satellite and bandwidth limited to one half T1 speed didn't allow us to perform comprehensive testing. By repeating each of the ACTS satellite tests on the hardware emulator, described above, we were able to verify that the emulator provided experimental results that were quite close to those seen over ACTS. Figure 5 shows a comparison of the two testing environments. Most of the test points are sufficiently close together that they are difficult to distinguish from each other on the graph, leading us to conclude that additional tests using only the hardware emulator should be good approximations of the results that we should expect over the ACTS satellite.

4.4 Results Over the Hardware Emulator - No Loss

This section describes tests using the hardware satellite emulator with sufficient router queuing that congestion loss could not occur. These tests duplicate the ACTS experiments above except that we tested a link at full T1 speed and added tests for file sizes of 12 Mbytes and 30 Mbytes. Figure 6 summarizes the results for this test.

Results of this test are similar to those of the ACTS test. XFTP performed slightly better than TCP Reno. The addition of the larger file sizes to the test tends to indicate that, in the absence of loss, TCP can achieve near-optimal throughput levels in this environment. TCP Reno reaches an average throughput for 30 Mbyte files that is 93% of optimal; XFTP achieves 96% for the same size file.

4.5 Results Over the Hardware Emulator - Congestion Loss

As a final comparison with the ACTS results, we repeated the congested test on the hardware emulator. As in the previous experiment, we emulated a full T1 link. In addition to the file

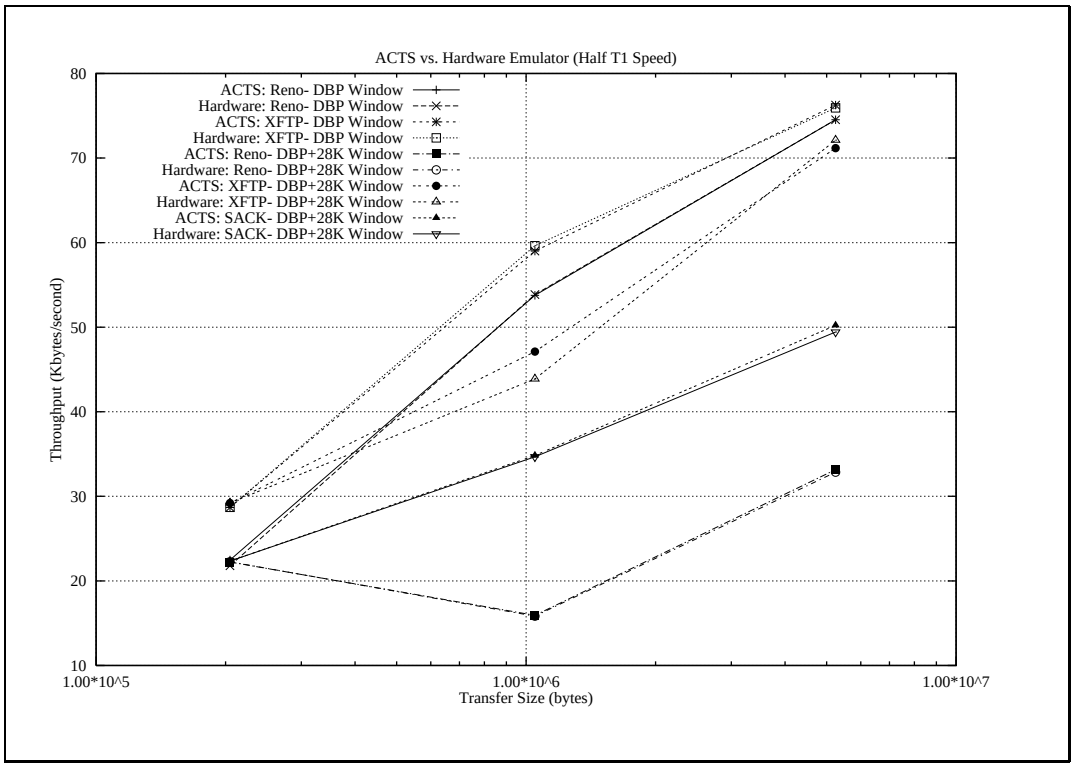

Figure 5: Comparing the Hardware Emulator to the ACTS Satellite
This graph shows the tests reported above together with the same tests conducted using the hardware emulator. For all of the tests except one, the results are nearly identical.

sizes used for the ACTS experiments, we added a test for files of size 12 Mbytes. Figure 7 shows our results.

The results from this experiment are again similar to the results of the ACTS experiment. TCP Reno did relatively poorly in the presence of congestion loss and only achieved throughput that was 32% of optimal for 12 Mbyte files. TCP SACK performed much better, achieving throughput that was 63% of optimal for 12 Mbyte files. As before, XFTP's ability to send more data during slow start and recover more quickly from congestion avoidance allowed it to achieve throughput that was 87% of optimal for 12 Mbyte files.

4.6 Throughput in the Presence of Link Errors

All of the tests above were conducted on links that were error free (to the best of our monitoring ability). In addition to their relatively long propagation delays, as compared to terrestrial links, satellite links are also believed to exhibit error characteristics that impede TCP's ability to achieve good throughput. To test TCP's ability to overcome bit errors, we transferred 5 Mbyte files over the hardware emulator at different bit error rates. The

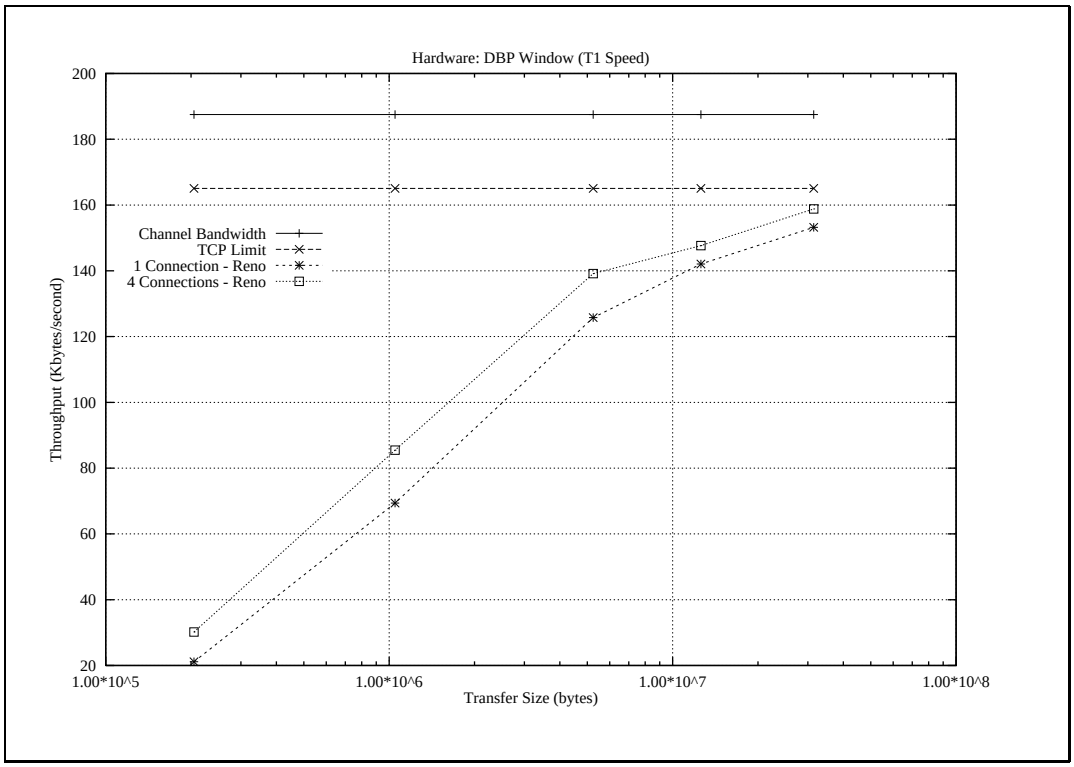

Figure 6: Tests over Hardware Emulator with Adequate Buffering
This graph compares the throughput achieved with standard TCP Reno with large windows against XFTP with the same effective window size. The link emulated was T1 speed. Throughput values for transfers of files of size 200 Kbytes, 1 Mbyte, 5 Mbytes, 12 Mbytes, and 30 Mbytes are plotted.

hardware emulator can be programmed to introduce a given number of bit errors at a deterministic interval. For these tests, the hardware emulator inserted 4000 consecutive bit errors into the data stream at a programmed interval. We chose 4000 bits (500 bytes) because that error burst was likely to affect 2 adjoining segments. Figure 8 shows the results.

For bit error rates of 1×10^{-7} (the hardware emulator was programmed to insert 4000 bit errors every 10,000,000 bits), the addition of SACK to the standard TCP Reno code is shown to greatly improve throughput relative to TCP Reno without SACK. At this error rate, we're destroying approximately one or two segments out of 2400, or injecting 3 loss events into a 5 Mbyte file transfer. Optimal throughput for this line speed would be approximately 165 Kbytes/second. TCP Reno achieved approximately 22% of this rate, while TCP SACK was able to achieve approximately 41% efficiency.

For bit error rates of 1×10^{-6}, we're destroying approximately one or two segments out of 240, or injecting 30 loss events into a 5 Mbyte file transfer. TCP Reno only achieved approximately 4% of the optimal rate, while TCP SACK was able to achieve approximately

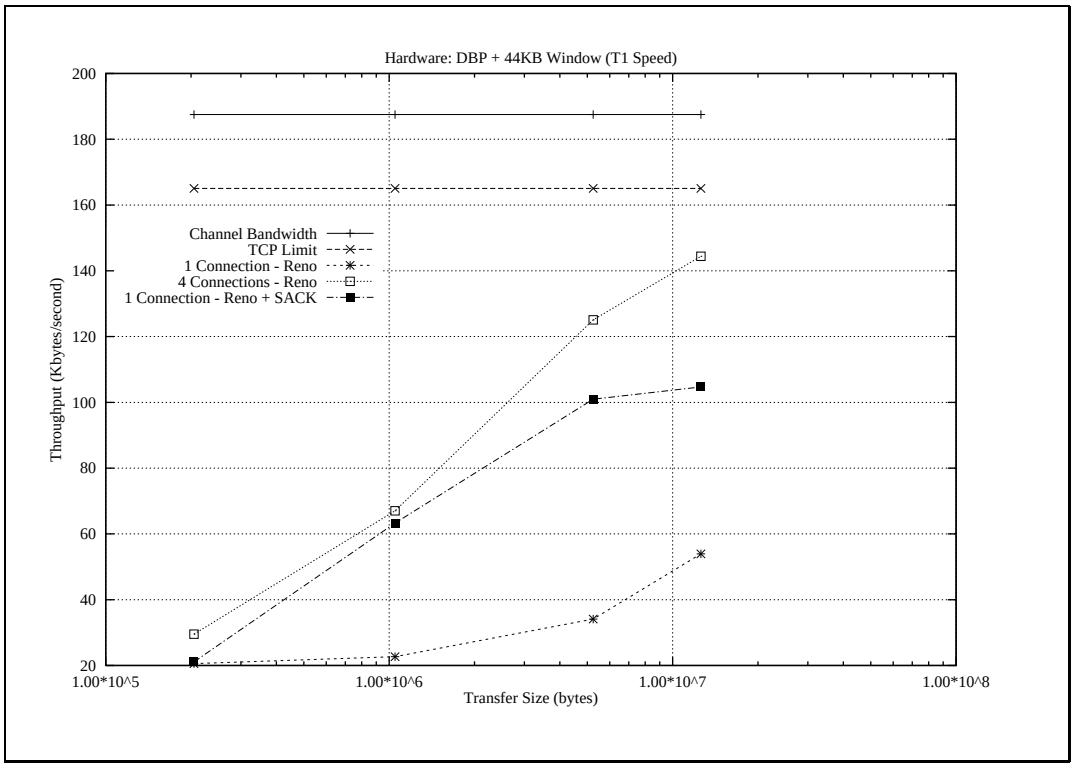

Figure 7: Tests over Hardware Emulator with Insufficient Buffering
This graph compares the throughput achieved with standard TCP Reno and TCP SACK, both with large windows, against XFTP with the same effective window size. The link emulated was T1 speed. Throughput values for transfers of files of size 200 Kbytes, 1 Mbyte, 5 Mbytes, and 12 Mbytes are plotted.

9% efficiency. At an error rate of 1×10^{-5}, affecting one or two segments out of 24, both of the protocols exhibits very poor performance.

5 Conclusions

The experiments presented in this paper indicate that a version of TCP using slow start and congestion avoidance, fast retransmit and fast recovery, and large (scaled) windows can allow an application that uses TCP to achieve high throughput over satellite links. When network congestion and higher bit error rates are present, the addition of selective acknowledgments (SACK) can also improve performance in some cases.

Two TCP mechanisms appear to have a limiting effect on TCP's satellite performance: slow start and congestion avoidance. Slow start over satellite links takes 6.5 seconds to reach maximum throughput. When lost segments trigger congestion avoidance, the resulting

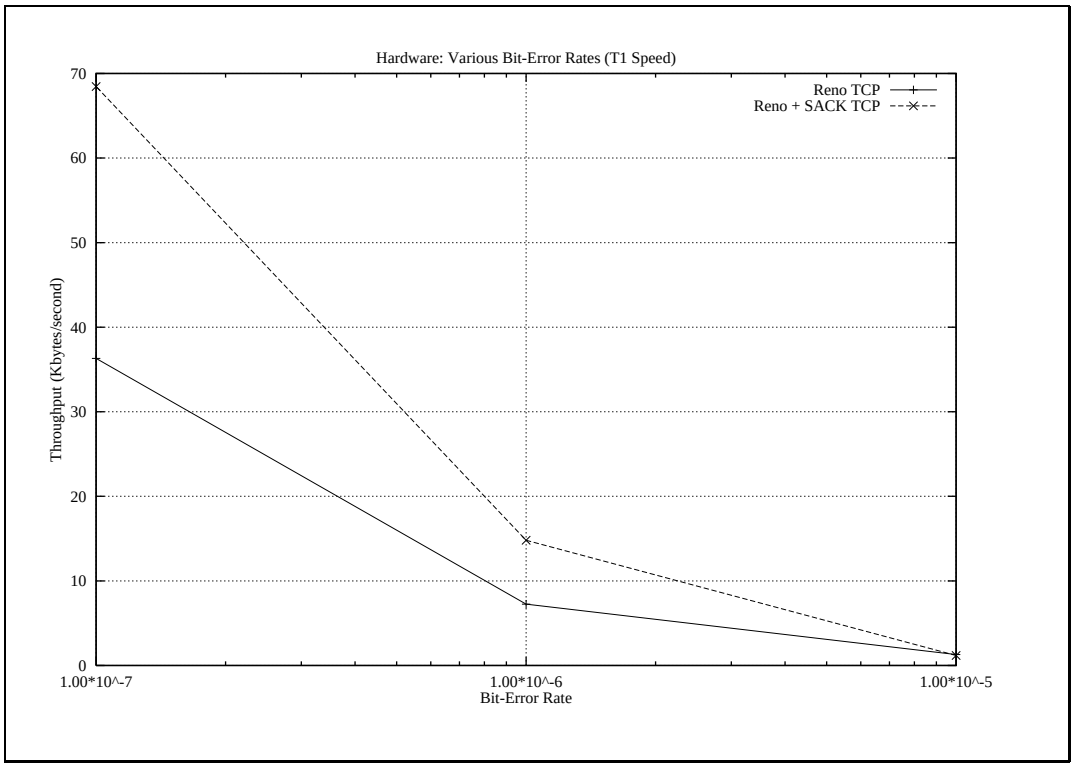

Figure 8: Tests over Hardware Emulator with Bit Errors
This graph compares the throughput achieved by TCP Reno and TCP SACK for different bit error rates.

throughput decrease can continue for as long as several minutes. Because the slow start and congestion avoidance mechanisms are generally considered to be essential to well-behaved TCP implementations on the Internet, however, suggesting changes to these mechanisms must be undertaken with extreme caution. Further research into these mechanisms is clearly required.

For further information regarding TCP work being done at Ohio University, software used for these experiments, or references to related work and publications, visit us on our web site at `http://jarok.cs.ohiou.edu/`.

References

[AO96] Mark Allman and Shawn Ostermann. Multiple Data Connection FTP Extensions. Technical report, Ohio University, 1996. (in preparation).

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

[Com95] Douglas E. Comer. *Internetworking with TCP/IP Volume I, Principles, Protocols, and Architecture.* Prentice-Hall, Englewood Cliffs, New Jersey, third edition, 1995.

[Jac88] V. Jacobson. Congestion Avoidance and Control. In *Proceedings ACM SIGCOMM '88*, pages 314–329. ACM, August 1988.

[JBB92] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance, May 1992. RFC 1323.

[Kru95] Hans Kruse. Performance Of Common Data Communications Protocols Over Long Delay Links: An Experimental Examination. In *3rd International Conference on Telecommunication Systems Modeling and Design*, 1995.

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

[PD96] Larry L. Peterson and Bruce S. Davie. *Computer Networks: A Systems Approach.* Morgan Kaufman, 1996.

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

[PR85] J. Postel and J. Reynolds. File Transfer Protocol (FTP), October 1985. RFC 959.

[Ste94] W. Richard Stevens. *TCP/IP Illustrated, Volume 1.* Addison-Wesley, Reading, Massachusetts, 1994.

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