ON THE PERFORMANCE OF TCP-BASED DATA TRANSFERS ON A FADED KA-BAND SATELLITE LINK*

Hans Kruse
Ohio University, Rm 197, 9 S. College Str., Athens, OH 45701, USA
740-593-4891 voice, 740-593-4889 Fax, kruse@ohiou.edu

Shawn Ostermann
Ohio University
740-593-1234 voice, ostermann@cs.ohiou.edu

Mark Allman
NASA Glenn Research Center/BBN Technologies
216-433-6586 voice, mallman@grc.nasa.gov

INTRODUCTION

Satellites using the Ka band are capable of providing high capacity data communications links. It is very likely that much of this data traffic will use the TCP/IP suite of protocols. This paper focuses on the performance of the TCP protocol [Pos81], which is used for reliable, connection-oriented transfer of information. Since TCP will interpret a bit error introduced during a link fade as an indication of congestion, it is important to understand the performance implications of allowing the satellite link to operate at a non-zero bit error rate. Mitigation of fade-induced bit errors carries a cost in the form of increased power requirements or the need for error correction, which requires additional bandwidth and circuitry. Allowing the bit error rate on the link to rise too much will leave the satellite user with poor performance, and use link capacity for retransmission of damaged packets. It is therefore important to understand the maximum error rate at which TCP can operate with acceptable performance. This paper reports on an experimental investigation of this question using the NASA ACTS satellite during user-induced fades. The focus of this study is twofold. First, we examine TCP configurations that are likely to be found on typical users' desktop hosts. Second, we investigate TCP implementations that include more up-to-date TCP mechanisms. We are not concerned with the overall utilization of the satellite link, but rather the rate at which TCP performance declines as the bit error rate increases. The data presented in this paper are in fact the result of preliminary experiments which will lead up to the study of many simultaneous TCP flows using both faded and un-faded links.

TCP CONGESTION CONTROL

TCP uses a set of congestion control algorithms that were first outlined in [Jac88,Jac90] and standardized in [APS99]. These algorithms assume that segment drops indicate network congestion. In other words, the algorithms assume that all packet loss is caused by a router's queue being overrun by excessive traffic (TCP packets are normally referred to as segments). TCP's steady-state sending pattern is one of additive-increase and multiplicative decrease. In the absence of segment loss, TCP's congestion window (or the amount of data that can be transmitted before an acknowledgment is received) is increased linearly. When TCP detects a segment loss, it retransmits the lost data and halves the congestion window.

* To be presented at the 6th Ka-Band Utilization Conference, Cleveland, May 31 – June 2, 2000.

We used two variants of TCP for this study. The first variant is the version outlined in [APS99], called TCP Reno, which includes four basic congestion control algorithms: slow start, congestion avoidance, fast retransmit and fast recovery. TCP Reno is able to effectively recover if one segment is lost per congestion window of data. If multiple segments are lost in a given window of data, TCP Reno must wait for the retransmission timer to expire to retransmit the lost segments. This hurts performance for two reasons. First, the retransmission timer is typically quite long (usually a minimum of 1 second) and second the congestion window is reduced to one segment, rather than being halved, when the retransmission timer expires. The second version of TCP used in our study, called TCP with Rate-Halving [MSML99], uses TCP's selective acknowledgment (SACK) option [MMFR96] to make loss recovery more robust in the face of multiple lost segments from the same window. With SACK, the receiver informs the sender exactly which segments have arrived. Using this information, the TCP sender can quickly and efficiently recover from multiple lost segments from a window of data, while only reducing the congestion window by half [FF96].

In the experiments presented in this paper, not all of the losses are caused by network congestion; we altered the link characteristics to introduce data loss due to corruption that varied from negligible data loss to nearly total data loss. Therefore, TCP often makes the wrong inference by assuming network congestion and reducing the congestion window in our experiments. The research community is currently working on mechanisms to distinguish between congestion-based and corruption-based loss. However, no mechanism has yet been standardized to do so. Our goal is to quantify TCP performance over a real satellite link without the use of such mechanisms to gain an understanding into whether such mechanisms would be useful. See [ADG+00] for an overview of the work in this area.

EXPERIMENT CONFIGURATION

For this experiment, workstations using the NetBSD 1.3 Unix operating system are located on the Ohio University campus in Athens, OH, as well as at the NASA Glenn Research Center (GRC) in Cleveland, OH. A block diagram for the experiment is shown in figure 1.

Two sets of workstations are connected using an ACTS T1 VSAT terminal (using a 1.2m reflector) at Ohio University, and the Master Ground Station (MCS) at GRC. The circuit speed is standard T1 rate (1.536Mbps). Link fades primarily effect the T1 VSAT transmit side, so bit errors appear almost exclusively in packets traveling from Ohio University to the GRC. The workstation at OU is the TCP sender, so bit errors will mostly effect data packets, while acknowledgements being returned in response to these data will experience a much lower loss probability.

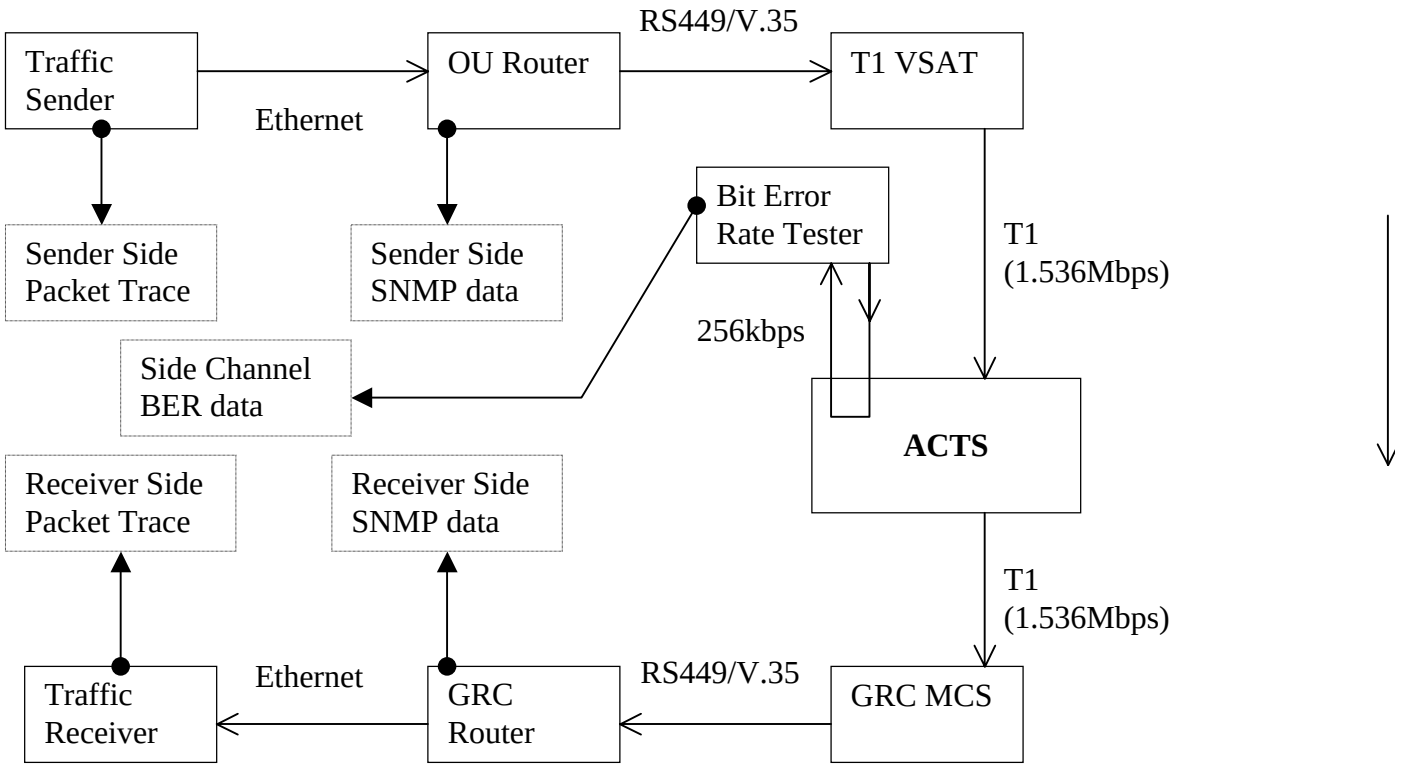

Figure 1 – Experiment Block Diagram

The data reported in this paper were collected during fade events which were artificially induced at the T1 VSAT terminal. Since the ACTS system is in inclined orbit operation, the T1 VSAT is equipped with an auto-tracking positioning system. During our runs, we disable the auto-track function and manually point the dish at a point along the satellite's path somewhat ahead of the actual satellite position. The amount of mis-pointing is adjusted to produce a bit error rate around 10^{-5}. During the experiment, the satellite will move towards the antenna pointing position, thereby gradually lowering the bit error rate. In some of our runs we have been able to let the satellite pass through the antenna position, thus generating a slowly increasing bit error rate towards the end of the run. Typical runs lasted 3 hours. It would of course be preferable to collect data during natural rain fades. However, we have access to the ACTS system for limited blocks of time. Moreover, the link margin of the ACTS system is quite high, and rain fades generating the bit error rates we wish to observe happen too infrequently to allow us to collect statistically significant data.

Each of the routers in the experiment are queried periodically using the Simple Network Management Protocol (SNMP). Typically these queries took place every 10 seconds. We monitor the RS449/V.35 interface inputs for packet errors on the satellite link. The packet drop counts for the Ethernet input and the RS449/V.35 output on the OU router are used to measure congestion loss.

We also collect "out-of-band" bit error rate data on a 256kbps loop-back channel between the T1 VSAT and the ACTS satellite, using an HP Data Communications Tester. Total bit errors and total errored seconds are recorded every 15 seconds.

TCP packets are captured both at Ohio University and at the GRC. The packet trace data is analyzed for the observed throughput in defined time intervals, and for the number of retransmissions during those same intervals.

All data collection devices are time synchronized to allow the correlation of all data. All data presented in this paper was collected using a single, continuous TCP data flow.

EXPERIMENTAL RESULTS

We have collected data using both the Reno and the Rate-Halving (SACK) mechanisms described above. For each of these TCP mechanisms, we have conducted tests with TCP maximum window sizes of 16kBytes (a common default setting), 102kBytes (the setting which corresponds to a bandwidth-delay product), and 512kBytes (which causes performance to be determined by the network path characteristics in conjunction with the TCP congestion control described earlier).

We find that the 16kByte window restricts the maximum throughput that can be observed, as was expected. We also find that the Reno and SACK mechanisms – regardless of window size – yield roughly the same performance as measured by throughput. This result is still under investigation, and further analysis will be needed to produce a detailed explanation of the factors that produce this behavior. For this paper we show results using the SACK mechanism, with a TCP maximum window size of 512kBytes. These data represent the most aggressive form of error recovery and reveal the effect of network path behavior without TCP window limitations. As noted above, we have found TCP Reno to perform roughly at the same level if the same window size is used.

Figure 2 shows the relationship between the link fade and the bit error rate observed on the side channel. The fade level is reported by the T1 VSAT terminal to the MCS, where these values are archived. Only the fade level of the terminal receive signal is measured. Since – as we noted above – the terminal transmit side produces most of the errors, it is not given that the receive fade will track the error rate. We can see that the antenna mis-pointing resulted in a 5dB fade at the outset, with the fade diminishing to 0dB over the course of about 2 hours. The bit error rate does follow the receive fade over the course of these two hours, indicating that receive fade levels may be of use in setting adjustable link fade mitigation levels at least on this time frame. Our observation suggest that the absolute ratio of bit error rate and receive fade level does not remain constant over longer periods of time, but rather is subject to time of day and satellite orbit related factors.

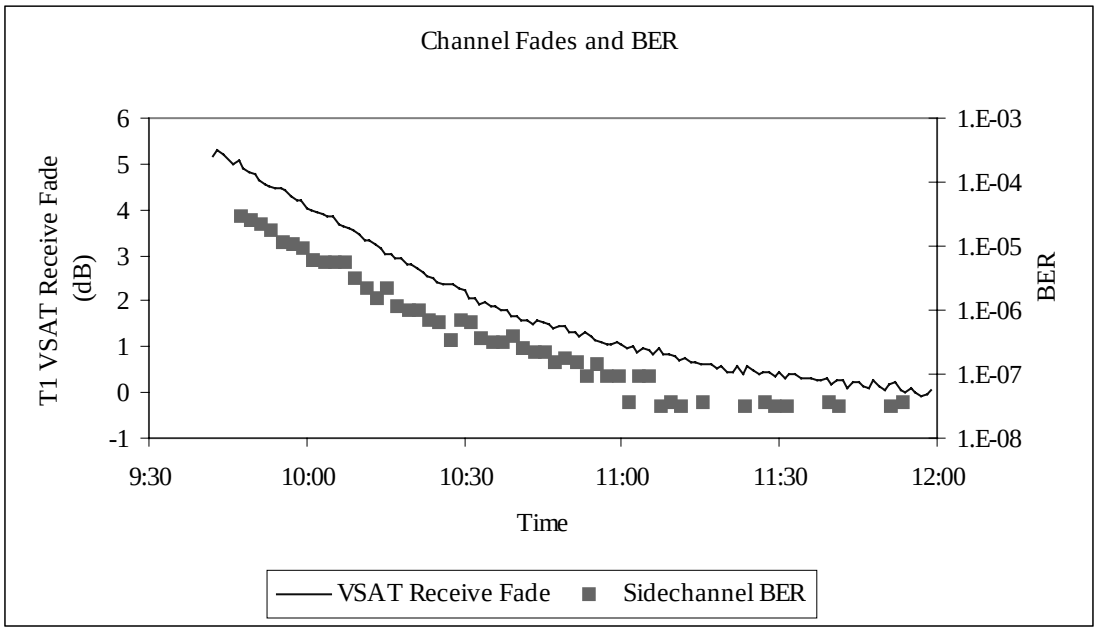

Figure 2
This figure shows the observed fade (in dB relative to the clear sky value) resulting from the antenna mis-pointing. Since the antenna points ahead of the satellite along its path, the fade decreases over time. Also shown is the Bit Error Rate (BER) measured during the experiment on the side channel.

Next we examine the relationship between bit error rate, the probability of segment loss, and the observed throughput. The data are shown in figure 3. To determine the segment loss rate, we divide the observed data into 2 minute intervals. We note the number of packet errors reported by the routers, and divide the number of segments sent during the time interval into the packet error count. The bit error rate is the same as in figure 2, i.e. collected on the side channel. We also use the total number of transmitted bytes of TCP user data to compute the throughput during each time interval.

We observe that the segment error rate and the bit error rate track each other very well over time. The two rates are separated by about 4 orders of magnitude, consistent with the fact that a full-sized segment carries about 12,000 bits. While our instrumentation is not designed to measure the distribution of bit errors, this result gives at least some indication that the fade-induced bit errors are random, and do not appear in large bursts.

The T1 channel permits a sustained throughput of about 185,000 bytes/sec of TCP user data when fully utilized. We note that the observed throughput remains at a fraction of this value until bit error rates drop well below 10^{-7}. To better quantify the relationship between throughput and bit error rates, we combine the results from figure 3 with the results from other runs using the same settings; this data summary is shown in figure 4.

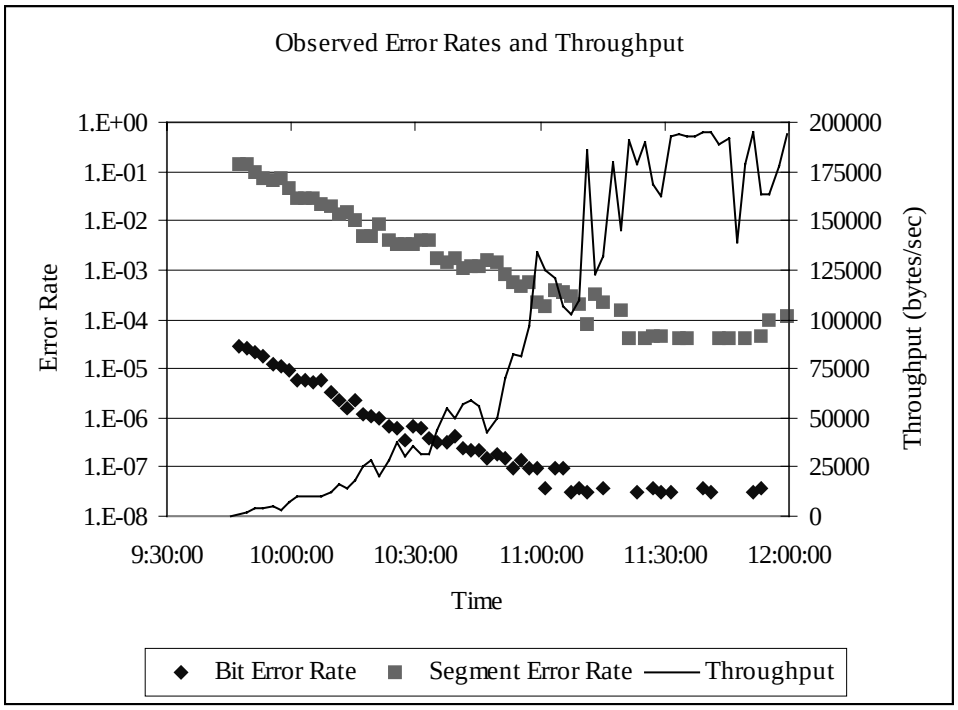

Figure 3

This figure shows the correlation of the bit error rate on the side channel, the segment error rate as reported by the routers, and the observed throughput. Data is shown for the same run as in figure 2. The TCP sender was using 512kByte windows and the SACK mechanism.

In figure 4 we combine data obtained during 4 separate runs with a maximum TCP window of 512kBytes, and 3 runs using a 16kByte TCP Window (we note again that SACK and Reno yield identical results for both window sizes; the 512kByte data were taken using SACK, and the 16kByte data using Reno). We plot the throughput observed in each time interval against the side channel bit error rate for that same time interval. For a 512kByte window, the observed throughput is at or near full channel utilization until the bit error rate reaches 10^{-8}.[1] The throughput at a bit error rate of 10^{-7} is less than 50% and degrades rapidly as bit error rates increase beyond 10^{-7}. It is worthwhile to again note the 512kByte window results in Figure 4 were obtained using the modern SACK mechanism. The results shown in figure 4 represent the best current TCP configuration with respect to handling link errors. The 16kByte window constrains the TCP sender to a sending rate below the link speed. Throughput on this constrained TCP flow remains constant until the bit error rate reaches $5x10^{-7}$, and then starts to degrade.

[1] Since we are using a log scale, a bit error rate of zero is plotted at 10^{-9}.

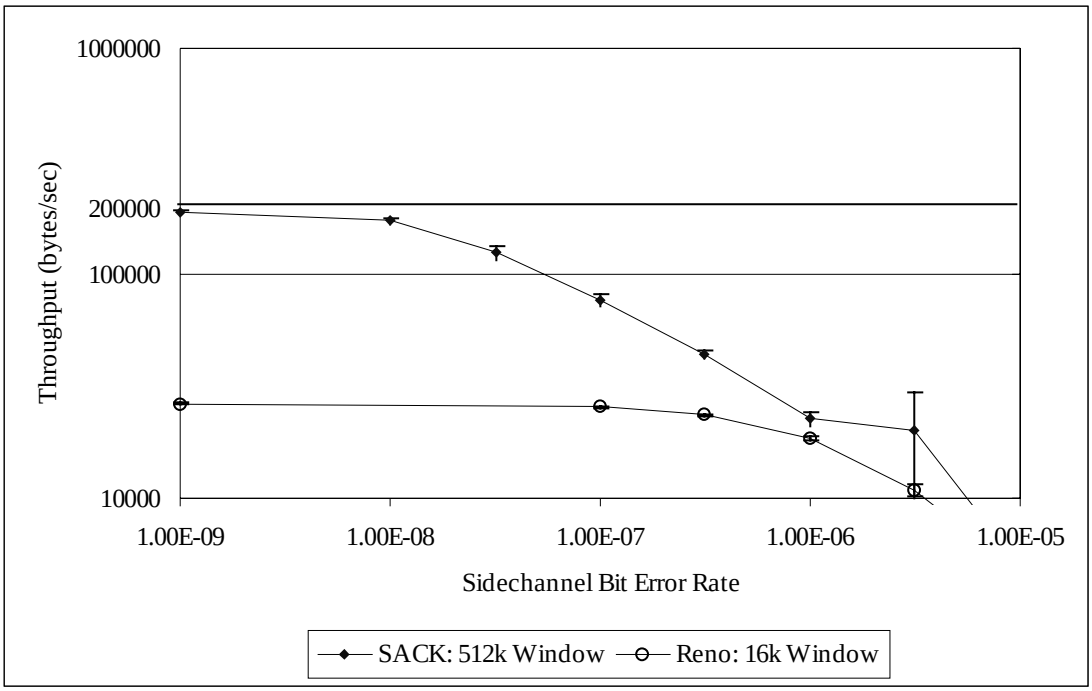

Figure 4
Measured TCP throughput as a function of the bit error rate measured on the side-channel. These data were obtained using the SACK implementation, with TCP window sizes of 16 kBytes and 512kBytes.

CONCLUSIONS

We have presented initial results from a series of TCP tests on faded Ka Band links. We have been able to measure TCP steady-state throughput during error rates ranging from 10^{-8} to 10^{-5}. Our measurements show that TCP performance begins to degrade noticeably for bit error rates above about $3x10^{-8}$ on a T1 link. The results show an exponential decline for error rates above this level, with throughput cut roughly in half by the time the bit error rate reaches 10^{-7}. At this loss rate, TCP receives what it sees as a steady flow of congestion indications that prevent it from ramping up to link speed. At these error rates there is a very low probability that more than one packet will be lost during one congestion window. Our results show that the SACK mechanism performs no better than the older Reno version in this case[2]. As we have noted before, the research community is actively looking into protocol enhancements that may permit TCP to distinguish corruption and congestion losses. However, given currently understood TCP mechanisms, geostationary satellite links need to maintain bit error rates well below 10^{-7} to permit TCP flows in excess of 1Mbit/sec.

The data presented in this paper are designed to measure the performance of a single, steady-state TCP flow in the presence of a slowly varying link fade. As a next step we are planning experiments with many smaller, concurrent TCP flows. While the current data address network utilization based on a single TCP flow, the multi-flow tests will address the perceived end-user

[2] Note, however, that SACK is expected to perform better than Reno in the general case where packets experience a more bursty mix of corruption and/or congestion losses.

performance. We also note that the bit error rate during a satellite link fade varies slowly compared to the time constants inherent in TCP. Other wireless links, especially those used for mobile terminals, may experience a much more rapid onset and cessation of fade conditions. In these cases SACK may perform differently than Reno.

BIBLIOGRAPHY

[ADG+00] Mark Allman, Spencer Dawkins, Dan Glover, Jim Griner, Diepchi Tran, Tom Henderson, John Heidemann, Joe Touch, Hans Kruse, Shawn Ostermann, Keith Scott, Jeff Semke. Ongoing TCP Research Related to Satellites, February 2000. RFC 2760. `

[APS99] Mark Allman, Vern Paxson, W. Richard Stevens. TCP Congestion Control, April 1999. RFC 2581.

[FF96] Kevin Fall, Sally Floyd. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. ACM Computer Communications Review, 26(3), July 1996.

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

[Jac90] Van Jacobson. Modified TCP Congestion Avoidance Algorithm. Email to the end2end-interest mailing list. Available from: ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt

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

[MSML99] Matt Mathis, Jeff Semke, Jamshid Mahdavi, Kevin Lahey. The Rate-Halving Algorithm for TCP Congestion Control. Internet-Draft draft-mathis-tcp-ratehalving-00.txt. August 1999. (Work in progress).

[Pos81] Postel, J., "Transmission Control Protocol", September 1981, STD 7, RFC 793.