TCP Behavior in Networks with Dynamic Propagation Delay

Mark Allman, Jim Griner, Alan Richard[1 2]

Abstract—**This paper provides a preliminary investigation into the impact a link with changing propagation delay has on the performance of TCP file transfers. We investigate over a dozen different variable delay patterns, based on spacecraft movement. We highlight the performance impact of such variability, paying close attention to TCP's retransmission timer, which is based on the observed round-trip time of the network path. In addition, we explore one scenario in which the round-trip time across a network path suddenly changes due to a large change in the path between the two end-points. We conclude that the variable delay network paths studied in this paper do not drastically impact TCP performance.**

Keywords—**TCP, variable RTT, retransmission, satellite.**

I. Introduction

In this paper we investigate the effects of network links with propagation delays that change over time on the performance of the Transmission Control Protocol (TCP) [1]. NASA would like to use commercial protocols to communicate with its assets in space and therefore, this preliminary investigation involves communication between various spacecraft and the Earth. However, the results are applicable in any situation where the propagation delay of a link changes modestly over time. The round-trip time (RTT) of the network path used in our experiments is not only changing, but also quite large. Therefore, our environment shares a number of challenges with the more static long-delay environments that have been researched by the community recently (see [2], [3] for an overview). In this paper we will investigate not only the performance TCP is able to attain over links with variable delay, but also the impact of the variable propagation delay on TCP's *retransmission timeout* (RTO).

TCP provides reliable data transfer to higher-layer applications. In order to do this, TCP must detect when a segment has been lost and then retransmit the segment. A number of ways for detecting lost segments have been developed (see [4] for a discussion). The most basic mechanism is for TCP senders to track the round-trip time (RTT) of the network path. If an acknowledgment has not been received in the expected amount of time (based on the RTT), the segment is retransmitted. If the RTT changes rapidly the RTO may not be able to adapt and the TCP sender may end up retransmitting segments that would have not required retransmission had the sender simply waited longer for the acknowledgment [5].

This paper is a preliminary investigation and is organized as follows. Section II outlines the methodology and simulation environment used in our experiments. Section III outlines simple tests that verify that TCP's RTO estimator correctly handles the variable delay scenarios investigated in this paper. Section IV outlines simple investigations of various TCP transfer sizes over variable delay network paths. Section V discusses simple experiments involving handoffs between two significantly different network paths. Finally, section VI gives our conclusions and outlines future work in this area.

II. Simulation Setup

The variable delay scenarios used in our investigation were developed by choosing several satellite orbits within the low-Earth orbit (LEO; 500 km – 900 km altitude), mid-Earth orbit (MEO; 5,000 km – 12,000 km altitude), and geosynchronous (GEO; 36,000 km altitude) orbital bands. We used the Satellite Tool Kit (STK), version 4.0, to determine the line-of-sight distance from a GEO satellite (NASA's Tracking and Data Relay Satellite System or TDRS) to each of the LEO and MEO satellites considered. The orbital data was taken for the time period July 1–2, 1999. During this time period some satellites have as many as fifteen contact periods with the TDRS 5 satellite, while others have as few as one contact period. Two contact periods were selected for each satellite. The first was the minimum line-of-sight distance variability period, and the second was the maximum variability period. The STK distance data was divided by the speed of light, to obtain time delays, and processed in Matlab to obtain equations for each of the scenarios. We implemented a new type of link delay in the *ns* network simulator that used the developed equations to set the link delay as a function of time.

The topology of the network used in our investigation is shown in figure 1. The characteristics of the chosen variable delay scenarios are outlined in table I. Finally, the RTTs of the various link delay scenarios as a function of time during the contact period are given in figure 2.

In addition to investigating different variable delay scenarios the transfer size varies between 4 segments and 10,000 segments. The SACK-based TCP variant included in *ns* is used in all experiments. The segment size is 1500 bytes in all simulations. The advertised window modeled is large enough (500 segments) to never be a limiting factor in the transfers (emulating hosts with autotuned socket buffers [6]). Finally, the clock granularity, G, used to measure RTTs and set the retransmission timeouts is varied. We use $G = 500\ ms$ to model many current, widely used implementations of TCP. In addi-

[1] Mark Allman is with NASA GRC/BBN Technologies; Jim Griner is with NASA GRC; Alan Richard is with NASA GRC/Analex Corporation.

[2] This paper appears in the proceedings of IEEE Globecom 2000.

Scenario Number	Scenario Name	Orbit Band	Apogee[3] (km)	Perigee[4] (km)
1	Minimum variability from Sputnik-40[5]to TDRS 5.	LEO	293	286
2	Maximum variability from Sputnik-40 to TDRS 5.	LEO	293	286
3	Minimum variability from MIR Space Station to TDRS 5.	LEO	351	356
4	Maximum variability from MIR Space Station to TDRS 5.	LEO	351	356
5	Minimum variability from International Space Station to TDRS 5.	LEO	400	384
6	Maximum variability from International Space Station to TDRS 5.	LEO	400	384
7	Minimum variability from COBE[6]to TDRS 5.	LEO	898	890
8	Maximum variability from COBE to TDRS 5.	LEO	898	890
9	Minimum variability from RADCAL[7]to TDRS 5.	LEO	891	770
10	Maximum variability from RADCAL to TDRS 5.	LEO	891	770
11	Minimum variability from LAGEOS-2[8]to TDRS 5.	MEO	5951	5619
12	Maximum variability from LAGEOS-2 to TDRS 5.	MEO	5651	5619
13	From NAVSTAR-01[9]to TDRS 5 (one continuous contact period).	MEO	20559	20254

TABLE I
CHARACTERISTICS OF CHOSEN VARIABLE DELAY SCENARIOS.

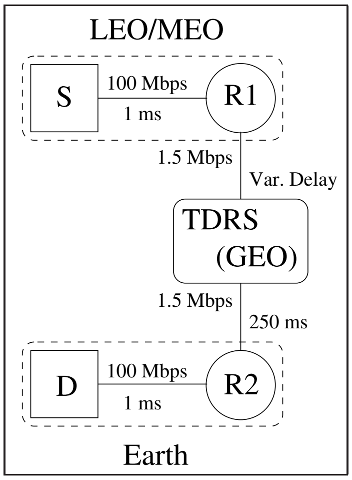

Fig. 1. Network topology.

tion, we use $G = 1\ ms$ to investigate whether performance could be increased by using a more accurate timer. Finally, we measure performance over the entire curve (shown in figure 2) by varying the start time of the connections. Specifically, our start times were roughly every 60 seconds (the start time is randomly selected to be within ± 0.5 seconds of each 60 second interval).

[3]Apogee is the point in an orbit most distant from the body being orbited.

[4]Perigee in the point in an orbit nearest to the body being orbited.

[5]Sputnik-40 represents the 1/3 scale Sputnik launched from MIR on October 4, 1997.

[6]COBE represents the Cosmic Background Explorer launched November 18, 1989.

[7]RADCAL is the Radar Calibration Satellite launched June 25, 1993.

[8]LAGEOS-2 is the Laser Geodynamics Satellite launched October 22, 1992.

[9]NAVSTAR-01 is the first GPS prototype satellite launched February 22, 1978.

III. RTO ESTIMATOR VALIDATION

The first test conducted over the network paths outlined above was to verify that the RTO estimator was able to adapt to slowly changing propagation delays. The retransmission timeout (RTO) is the method of last resort for repairing lost segments and providing reliable data delivery to applications using TCP. The estimator works by tracking the round-trip time (RTT). If a TCP segment is transmitted, but not acknowledged within the expected amount of time the segment is retransmitted. In addition, TCP assumes the dropped segment is due to network congestion (see [7] for the original discussion of TCP's congestion control algorithms and [4] for the specification of the algorithms). Therefore, the *congestion window* (*cwnd*) and *slow start threshold* (*ssthresh*) are reduced to 1 segment and half the previous *cwnd* respectively. The *cwnd* specifies the amount of data the TCP sender can transmit before receiving an acknowledgment (ACK). Hence, the *cwnd* reduction effectively reduces TCP's sending rate. Therefore, the goal of a good estimator should be to minimize the number of unnecessary retransmissions triggered.

The RTO estimator used in most TCP implementations (and this study) is based on the estimator presented in [7] and specified in [8]. The algorithm calls for TCP to track a smoothed average of the RTT, $SRTT$, as well as an estimate of the variance in the samples, $RTTVAR$. The RTO is then calculated as $SRTT + K \cdot RTTVAR$ where $K = 4$ (per most TCP implementations). (See [8] for the exact details.)

For each of the variable delay scenarios outlined above (S_1 – S_{13}) we started a TCP sender at time 0 seconds and allowed the transfer to continue for the entire length of the scenario. The maximum TCP window size was set to 1 segment such

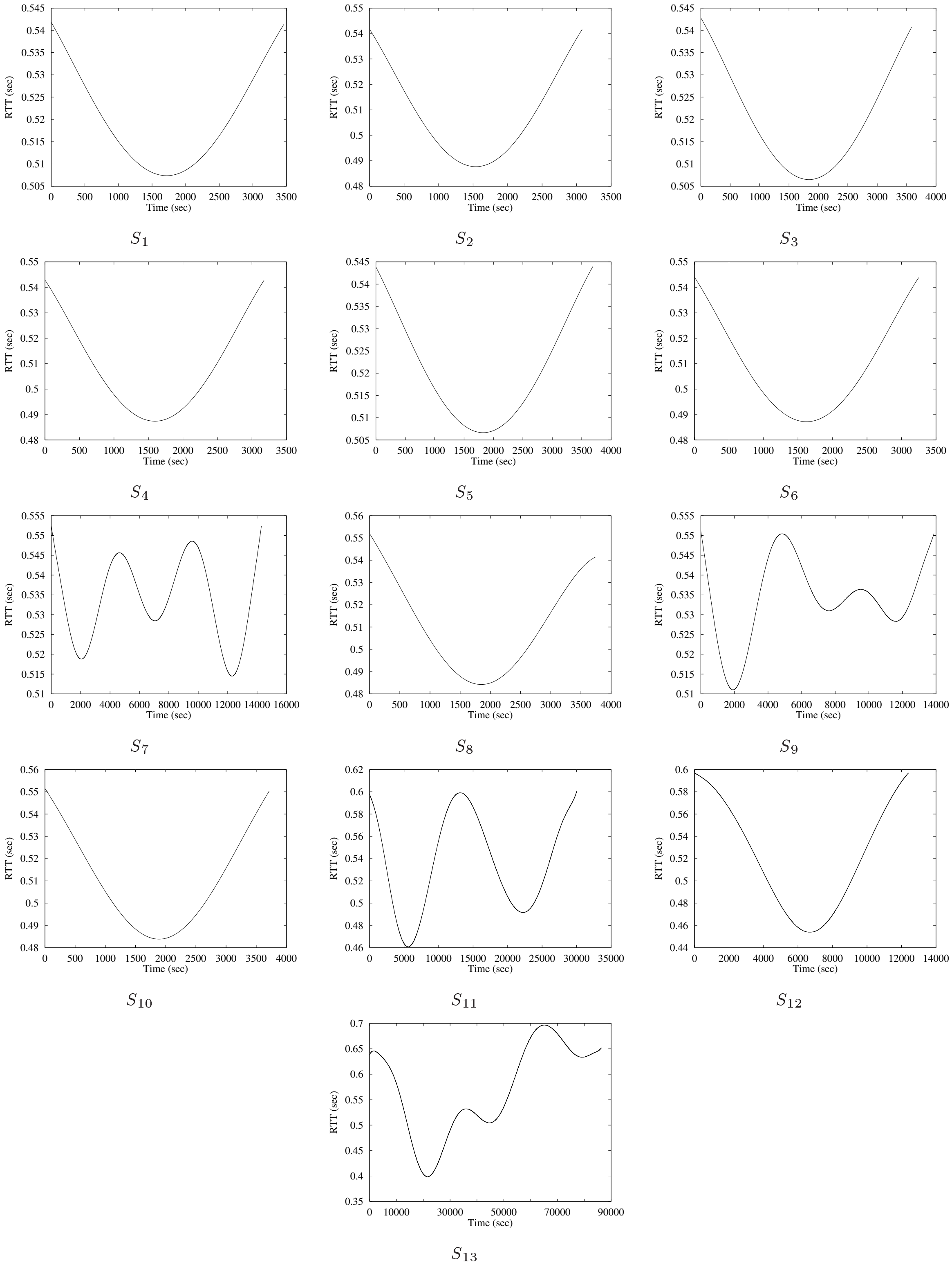

Fig. 2. RTTs of the variable delay scenarios investigated.

that queueing delays were not an issue. In this experiment, 1 segment was transmitted at a time. After the acknowledgment (ACK) for the segment is received a RTT sample is taken, the RTO is updated and another segment is sent. If the RTO estimator was not able to accurately adapt to the changing RTT of the network path we would have observed the TCP sender retransmit segments needlessly. However, for all variable delay scenarios and for $G = 1\ ms$ and $G = 500\ ms$ we observed no retransmissions in this simple set of experiments.

These experiments indicate that the RTO estimator is able to cope with the changing RTT of the link scenarios studied in this paper. As outlined in [5] spikes in the RTT can cause needless retransmissions. Such network characteristics did not come into play in these experiments.

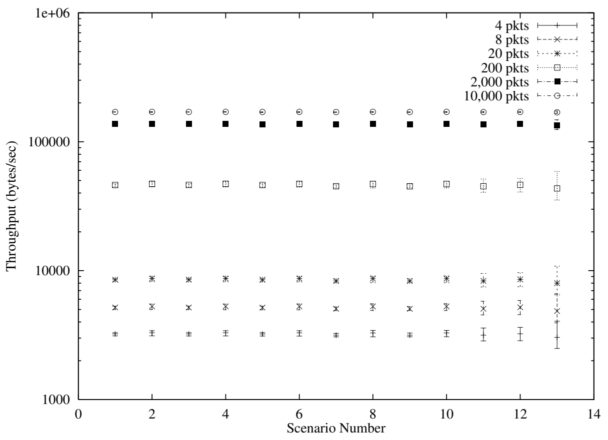

Fig. 3. Throughput as a function of variable delay scenario.

IV. Single Flow TCP Tests

We conducted simulations taking into account all combinations of the variables outlined in section II. Figure 3 shows the average throughput as a function of the variable delay scenario for each file size tested and $G = 500\ ms$. The results are expected and conform to prior results obtained in static long-delay networks [9]. As the file size grows the throughput increases. The shorter transfers spend most (if not all) of their time using the slow start algorithm and therefore obtain lower throughput. The figure includes error bars that indicate the minimum and maximum throughput observed. Unlike in static delay environments the throughput a given transfer receives is modestly dependent on the delay pattern when the transfer is being conducted. As the figure illustrates, short transfers are more susceptible to performance variation than long transfers (where the error bars are not even noticeable). Scenarios S_{11} – S_{13} show the most amount of throughput variation. This is expected as the RTT varies more during these scenarios than the others (by at least a factor of 2). Finally, note that the largest transfer (10,000 packets) essentially fully utilizes the available bandwidth (1.5 MBps) of the network path in all scenarios.

Next, we investigated whether the clock granularity, G, TCP uses to measure RTTs and schedule retransmissions has pronounced effects in this environment. For the smaller transfer sizes (4–200 segments) the value of G did not have an impact on the transfer. However, for the two largest transfer sizes (2,000 and 10,000 packets) a clock granularity of $G = 1\ ms$ did cause needless retransmissions, while when $G = 500\ ms$ the transfers experienced no bad retransmissions.

We found two reasons for the lack of bad retransmissions in short transfers. [5] suggests that bad timeouts are caused by spikes in the RTT. However, small transfers with no competing traffic do not build queues large enough or fast enough to create RTT spikes. Therefore, the RTO estimator is well behaved. The second reason TCP avoids bad timeouts in short transfers is $RTTVAR$ is initially set to $\frac{R}{2}$ where R is the first RTT measurement taken. Hence, when the RTT is large this adds a significant delay to the RTO calculation at the beginning of a connection (or for an entire short transfer), making the estimator less likely to trigger a bad retransmit.

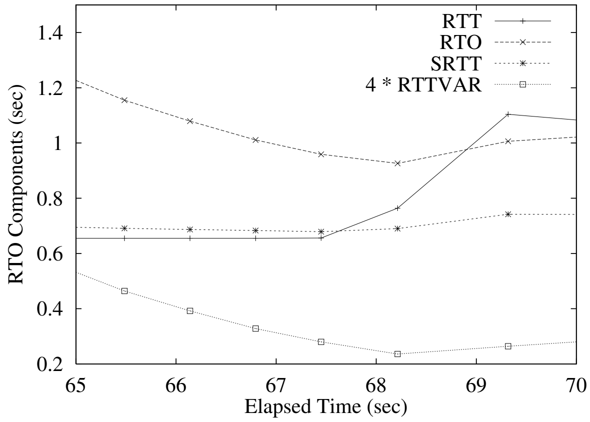

Fig. 4. RTO components as RTT exceeds RTO on the S_{13} network link.

When transferring large files with $G = 1\ ms$ we did observe the RTO timer firing prematurely and needlessly retransmitting data. These events were caused by RTT spikes due to rapid increases in queueing delays. TCP experienced needless retransmissions in every variable delay scenario except S_7. Figure 4 shows the RTO components for a 2,000 segment file transfer across the S_{13} network link. As shown in the plot, approximately 69 seconds into S_{13} (and ≈ 9 seconds into the TCP transfer) the RTT exceeds the RTO. This causes a needless retransmission. The root cause of this is a spike in the RTT caused by the queueing delay increasing as TCP quickly increases the load on the network during slow start. The RTO algorithm is not able to adapt to this RTT spike fast enough and therefore the TCP sender needlessly retransmits a segment roughly 23 ms before the ACK for the original transmission arrives at the sender. A TCP sender using a 500 ms granularity clock does not needlessly retransmit in this case. As a result, when $G = 1\ ms$ the transfer takes approximately 110 seconds, while when using $G = 500\ ms$ the transfer takes only about 82 seconds. Note that we ran an additional simulation with a static link delay of approximately the same delay as shown in figure 4 with $G = 1\ ms$ and the RTT spike caused a needless retransmission in that case, as well. Therefore, as argued in [5], RTT spikes are the main cause of bad RTOs, rather than the slowly changing link delay. As discussed in [5], there seems to be no easy fix to the RTO estimator to make it anticipate RTT spikes in general. The suggested strategy is to use a large minimum RTO (as $G = 500\ ms$ provides). Another mechanism that may help reduce spikes in the RTT is pacing TCP segments [10]. This mechanism attempts to smooth out the bursts caused by TCP's transmission pattern and avoid large queueing delays, thus helping to decrease RTT spikes.

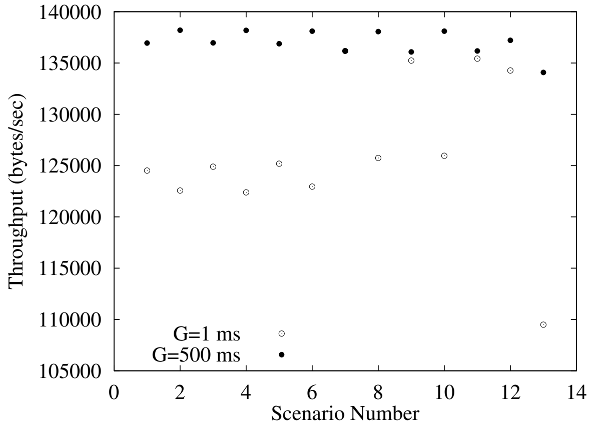

Fig. 5. Throughput comparison between $G = 1\ ms$ and $G = 500\ ms$ for a 2,000 segment transfer.

Figure 5 compares the average performance of a 2,000 segment transfer as a function of the delay scenario. As noted above, transfers over the S_7 link do not experience unnecessary retransmissions, therefore the performance is the same for $G = 1\ ms$ and $G = 500\ ms$. In all other scenarios, the performance is hurt by using a fine-grained timer. In other words, the greater precision of the timer, which can save time when the RTO expires appropriately, is more than offset by the number of times the RTO fires prematurely when $G = 1\ ms$. The degree to which the average performance suffers is a function of the percentage of transfers that experienced unnecessary timeouts. For instance, delay scenario S_1 has a lower percentage of transfers experiencing bad timeouts than scenario S_{13}, but a higher percentage of transfers with bad timeouts than scenario S_{11}. Therefore, the average throughput attained under S_1 is better than under S_{13} and worse than under S_{11}.

V. Handoff Scenario

The last item we consider is a scenario where a handoff is required. The scenario involves two spacecraft (Mir and the International Space Station) in low-Earth orbit communicating through one or two GEO satellites (TDRS 4 and TDRS 5). When both LEO spacecraft can communicate with the same GEO satellite, the communication simply goes through this common spacecraft. However, if each LEO spacecraft is communicating with a different GEO spacecraft, the traffic between ISS and Mir must occur through the ground. In other words, Mir sends traffic to one of the GEO satellites, which sends it to the ground. The traffic is then routed to the other GEO satellite and finally to ISS, resulting in basically a double satellite hop.

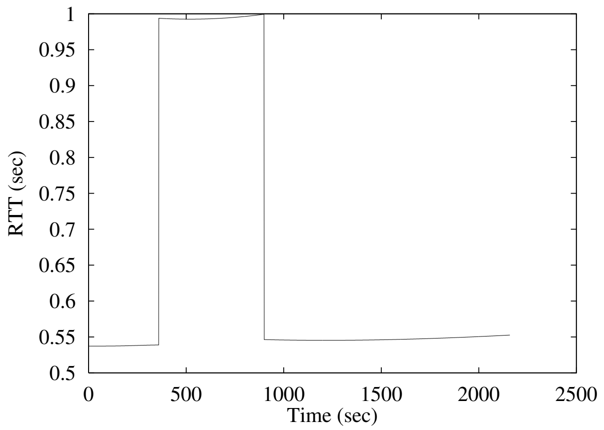

Fig. 6. RTT of unloaded network between two LEO spacecraft.

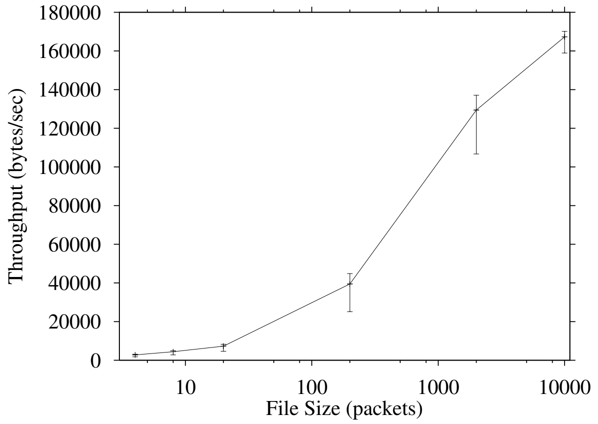

Fig. 7. Throughput as a function of file size for S_{14}.

We are only investigating the impact of varying RTTs on TCP performance in this paper and therefore, the handoffs between satellites are assumed to be perfect. Obviously, this is a large simplifying assumption in general and investigating handoffs is an area which deserves future attention.

The RTT of the network, denoted S_{14}, when unloaded is shown in figure 6. We repeated the stop-and-wait experiments outlined in section III on the S_{14} network. As one might expect, when $G = 1\ ms$ the RTO algorithm cannot cope with the large increase in the RTT at approximately 350 seconds into the period shown in figure 6. TCP sent a needless retransmission at this point. As expected, the RTO estimator had no problem with the drastic decrease in RTT that happens approximately 900 seconds into the simulation. When $G = 500\ ms$, effectively ensuring a large minimum RTO, no needless retransmissions were produced by the estimator. This illustrates the importance of a large minimum RTO.

Figure 7 shows the throughput as a function of file size for scenario S_{14}. The error bars on the plot illustrate the minimum and maximum throughput obtained for the given file size. As shown, the variability in the link delay causes the largest throughput variation in the medium file sizes (200 and 2,000 packets). This is caused because these files are more likely to be split across a handoff than the smaller files. When the transfer's RTT is suddenly increased, the *cwnd* must be increased accordingly, which takes some time (during which TCP underutilizes the network). The same thing happens to the 10,000 packet transfer. However, those transfers are long enough to better absorb the handoff performance degradation without taking a large overall performance hit.

VI. Conclusions and Future Work

This paper presents a preliminary evaluation of TCP over a network path that includes a link whose delay changes as a function of time. We illustrate that TCP is able to perform quite well in this environment. With the use of an RTO timer that has a large minimum value we have found that the changing delay does not trigger bad retransmissions. However, there can be quite large performance degradation if a fine-grained timer is used without imposing a minimum value on the RTO. There are several extensions to this work that we are planning on investigating.

- Section V only considers ideal handoffs. In practice, handoffs can cause packet drops, packet duplication and often times "dead" periods when no communication occurs. The impact these phenomena have on TCP performance should be studied.
- Often times when a satellite is moving the signal strength between it and the GEO satellite varies. With the varying signal strength comes the possibility of packet losses due to corruption. TCP interprets all packet losses as indications of network congestion and reduces the sending rate accordingly. Future work should investigate the extent to which this reduces TCP performance in general and what (if anything) can be done to combat it.
- Finally, future studies should take into account a more realistic traffic model. As more traffic is added to the network the RTT spikes will happen more randomly and this should shed some light on what the minimum RTO should be and what G is the most appropriate in the general case.

Acknowledgments

We thank Brian Kachmar and Jason Pugsley for their assistance in defining the variable delay models used in this paper.

References

[1] Jon Postel, "Transmission Control Protocol," Sept. 1981, RFC 793.

[2] Mark Allman, Dan Glover, and Luis Sanchez, "Enhancing TCP Over Satellite Channels Using Standard Mechanisms," Jan. 1999, RFC 2488, BCP 28.

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

[4] Mark Allman, Vern Paxson, and W. Richard Stevens, "TCP Congestion Control," Apr. 1999, RFC 2581.

[5] Mark Allman and Vern Paxson, "On Estimating End-to-End Network Path Properties," in *ACM SIGCOMM*, Sept. 1999.

[6] Jeff Semke, Jamshid Mahdavi, and Matt Mathis, "Automatic TCP Buffer Tuning," in *ACM SIGCOMM*, Sept. 1998.

[7] Van Jacobson, "Congestion Avoidance and Control," in *ACM SIGCOMM*, 1988.

[8] Vern Paxson and Mark Allman, "Computing TCP's Retransmission Timer," Apr. 2000, Internet-Draft draft-paxson-tcp-rto-01.txt (work in progress).

[9] Mark Allman, Chris Hayes, Hans Kruse, and Shawn Ostermann, "TCP Performance Over Satellite Links," in *Proceedings of the 5th International Conference on Telecommunication Systems*, Mar. 1997, pp. 456–469.

[10] Joanna Kulik, Robert Coulter, Dennis Rockwell, and Craig Partridge, "Paced TCP for High Delay-Bandwidth Networks," in *Proceedings of Globecom*, Dec. 1999.