TCP in Networks with Abrupt Delay Variations and Random Loss

Alhussein A. Abouzeid, Sumit Roy
Department of Electrical Engineering
University of Washington, Box 352500
Seattle, WA 98195-2500
e-mail:{hussein,roy}@ee.washington.edu

Abstract— **This paper provides a preliminary investigation of the effect of abrupt round-trip variations on the performance of different versions of TCP. It is shown that end-to-end goodput of a bulk TCP transfer can be severely limited by variations in RTT primarily due to the non-adaptiveness of TCP's timeout estimation and fast recovery thresholds. The paper proposes a model for evaluating the performance of TCP in environments with abrupt RTT changes and random packet loss and outlines possible future TCP improvements.**

Keywords— **Transport Control Protocol, Wireless Networks, Mobile Ad-Hoc Networks, Satellite Networks, Internet, Performance Evaluation.**

I. Introduction

TCP, the Internet Transport Control Protocol, is currently the most widely used transport protocol in packet networks with primary responsibility for congestion control. In the absence of any explicit information about the network configuration, TCP achieves its congestion control objective by attempting to drive the network to the point of full utilization (by increasing the rate at which it releases packets to the network) while continuously monitoring the network for signs of congestion and lowering the rate if congestion is detected. Packet loss is traditionally [1] used as a signal of congestion; however, this information to the TCP sender is only implicitly provided by the network. The original intent behind TCP's congestion control law is to assume that the *only* cause of any inferred packet loss is congestion. Thus situations where the packet loss is *non congestion related* such as due to unreliable data links (as is characteristic for wireless channels) or due to packet re-ordering will needlessly trigger the congestion avoidance mechanisms in TCP and adversely affect end-to-end throughput. Consequently there is a growing literature on modeling and performance evaluation of TCP performance where congestion loss is not dominant (or at least, not the only source of packet loss).

A number of recent studies show that a significant amount of round-trip delay variation may also result in inference of packet loss by a TCP sender; in our work, we first focus exclusively on this problem assuming ideal (non-lossy) paths and then extend the observations to the lossy case. Paxson [1] shows that multi-path routing in the Internet can cause "route fluttering" (frequent route changes) which results in significant out-of-order packet delivery and/or frequent abrupt end-to-end RTT variations. Henderson et al. [2] and Allman et al. [3] show that packet reordering and abrupt delay variations might be caused by the lossless hand-off of satellite links in LEO as well as GEO satellite networks. In [3], the authors focus on the effect of *slowly varying* round-trip variations on TCP performance and conclude that the variable delay network paths considered do not drastically impact TCP performance.

This paper presents a study of TCP operation over an end-to-end path experiencing *abrupt* randomly varying RTT and packet loss. A model for abrupt RTT variations is proposed; initial insight is obtained by considering a single abrupt RTT change. We compare the performance of two TCP versions to highlight issues regarding TCP's packet loss detection algorithms.

II. RTT Model & Simulation Setup

We consider a TCP source/destination pair in which the source has an infinite number of equal-sized packets to send (e.g. a large 'ftp' transfer). The end-to-end path is modeled as an abstract link with bandwidth μ (packets/sec.) and a time-varying round-trip delay $\tau(t)$ that switches at random instants between two values τ_1 and τ_2. Then, the total RTT (inclusive of packet transmission time) $T(t) = \tau(t) + \frac{1}{\mu}$ alternates between two values $T_i = \tau_i + \frac{1}{\mu}$. Let V_i denote the duration of stay in state i. Since in practice, the duration between two consecutive route changes is usually greater than one RTT we assume that $V_i \geq T_i$ with probability one. Specifically, we set $V_i = T_i + X_i$ where X_i is exponentially distributed with mean $\frac{1}{\lambda_i}$. The above models the RTT process as an alternating Markov (or Cox) process; a direct generalization (not considered here) would be a semi-Markov process with a larger number of states and associated transition probability matrix. In the context of an *ns2* [4] simulation model, the total RTT can be applied to the forward link with zero-delay reverse link for simplicity [2].

[1] Proposals to use Explicit Congestion Notification (ECN) are expected to be standardized soon by the IETF. However, ECN is expected to be used in addition to the current TCP packet loss detection schemes; thus, the results of our study also applies if ECN is used.

[2] Although in reality, $T(t)$ is the sum of a forward $T_f(t)$ and a reverse delay $T_r(t)$, it is difficult to differentiate between them in practice; it suffices for purposes of investigating TCP behavior that the acknowledgment (ACK) for a packet transmitted at time t is received at time $T(t)$.

The above provides a pathway for modeling RTT variation for a mobile client (distance between the source/destination is proportional to the RTT) whose distance from source (fixed host in the network) may be discretized into several states (here only 2 is considered) comprised of possibly unequal length paths. We analyze two versions of TCP: the original (old) Tahoe version, and the Reno version. We assume the reader is familiar with the basic window adaptation algorithms of these two protocols and the packet loss inference modes of TCP - namely, timeout (TO) and the reception of multiple duplicate ACKs (TD), typically 3. OldTahoe relies solely on timeouts (TO's) for packet loss inference while Reno/NewReno relies on both TO's and TD's.

The TO mode of packet loss inference can be briefly summarized as follows. TCP computes a smoothed RTT estimate (denoted $SRTT$) and a smoothed variance estimate (denoted $SVAR$) using an exponentially weighted moving average. The RTO estimate is then set to

$$RTO = SRTT + k\,SVAR \tag{1}$$

where the value of k in (1) first proposed in [5] was 2 and later increased to 4 in [6]. TCP timers employ RTT measurements using a clock with a certain granularity G that ranges from very low values up to a coarse value of 500 msec. Also, in many implementations, a minimum value of the RTO, $RTO_{min} = 2G$ is also used. The number of RTT samples per window round may also vary. Two options are proposed - one measurement per window round [7] or using all available samples by making use of the time-stamps options [6]. In both cases, we assume that ACKs for retransmitted packets are not used for timeout estimation [8].

The essential difference between Reno and NewReno is that NewReno responds to multiple TD's within the same window round as if only one TD took place. This difference is irrelevant in the context of the current model since we only focus on RTT changes that are limited to a maximum rate of one change per RTT as outlined earlier.

We describe TCP's congestion avoidance in terms of rounds where a round w starts by the arrival of a burst of ACKs (of length $w-1$) from previous round hence triggering the release of w packets at the beginning of the current round (called the busy period of the round) followed by an idle duration until the beginning of the next round.

III. Analysis of Single Transitions

In this section, we consider the two possible cases of a single ideal RTT change - RTT increase and RTT decrease with no packet loss. We perform a number of simulations involving single step RTT changes in order to characterize the amount of RTT change required to trigger false inference of packet loss and hence spurious retransmissions. In simulations, we set the link delay to an initial value τ_1, which is changed to τ_2 at some instant. All simulations use a packet size of 1KByte. We plot the packet sequence number, congestion window size (in packets), RTO and the path RTT as a function of time.

A. A Single RTT Increase $T_2 > T_1$

Consider a TCP session operating in congestion avoidance phase for a sufficiently long duration. Let w_1 and Y_1 denote the values of the window size and RTO estimate just before an RTT increase. Assume that the RTT increase takes place just prior to the start of the window round. Since we assume here a lossless path, the ACKs for packets from the previous round will arrive and trigger new packet transmissions during the current round. However, these ACK arrivals have an RTT of $T_2 > T_1$ and a false TO will surely occur if $T_2 > T_1 + Y_1$. It can be shown that even for $T_1 < T_2 < T_1 + Y_1$, a TO *may* still take place if certain conditions are satisfied regarding the window size and the instant of RTT change within the window round. Notice that previous work such as [3] state that a false TO occurs only if a *large* amount of RTT increase takes place; our observation shows that the amount of step RTT necessary to trigger false TO's can be small.

In general, the amount of step RTT increase required to cause a TO will depend on the following: (i) Whether the RTT increase takes place during the idle or the busy period of the round. (ii) The RTO just before the time instant the RTT is increased. (iii) The RTT before and after the change.

These conditions may be analytically computed but are not included due to space considerations.

The simulations in Figures 1 and 2 highlight the above mentioned dependence of the conditions for a TO. In these simulations, $T_1 = 0.2$, $T_2 = 0.3$ and $Y_1 \approx 0.2$; thus $T_1 < T_2 < T_1 + Y_1$. The simulations used a fine grained timer and the time-stamps option. The results confirm the above observations for the conditions required to trigger a TO by showing that the same amount of RTT change may (Figure 2) or may not (Figure 1) cause a TO, depending on the window size at the instant of the RTT change. The results for the case without time-stamps is identical, except that the RTO curve shows a slower decay following the 'spike' at the instant TO takes place.

B. A Single RTT decrease $T_2 < T_1$

Again, assume a TCP session in congestion avoidance. TCP infers a packet loss if the source receives multiple duplicate ACKs (i.e. ACKs with the same next expected packet number) exceeding a threshold denoted by D ($D = 3$ typically), and responds by halving the window size and going into a (short) fast recovery phase followed by re-entry into congestion avoidance phase. This mode of packet loss detection is generally abbreviated as TD (i.e. triple duplicate).

In the context of the proposed RTT model of Section II, an RTT decrease may cause a TD if the change occurs during the busy period of the round such that all of the following conditions are satisfied:
(i) there exists an idle period in the round of duration greater than the duration of D packet transmission;
(ii) at least D packets are transmitted after the RTT decrease;

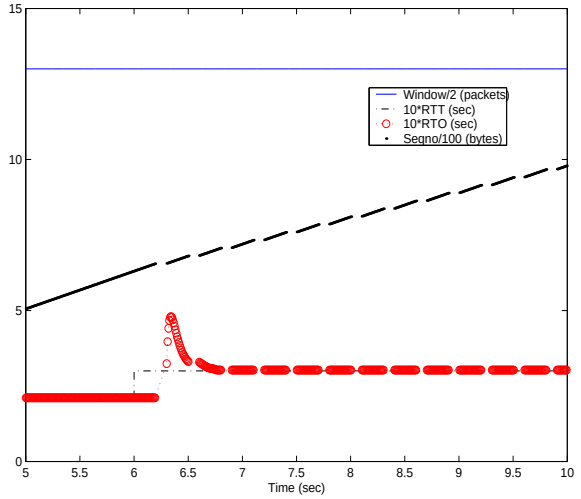

Fig. 1. Link Speed is 1Mbps (125 packets/sec), packet size is 1KBytes and $w_m = 26$ packets. Initially, $RTT = T_1 = 0.2$ sec. At $t = 6.0$ sec., $RTT = T_2 = 0.3$ sec. Observe that there is no TO.

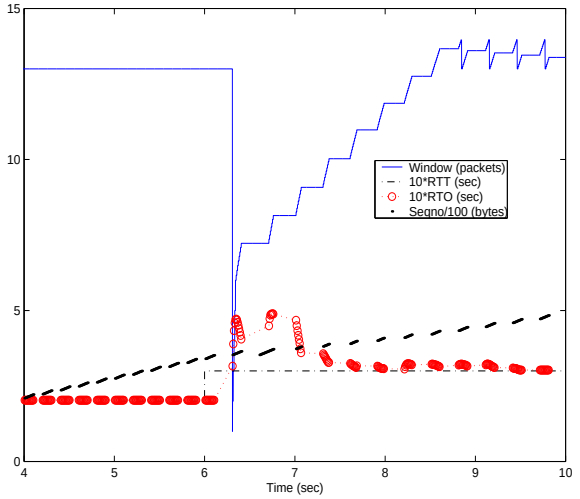

Fig. 2. Same scenario as in Fig.1 but with $w_m = 13$ instead of 26. In this case, a TO takes place.

and (iii) the RTT decrease is such that the ACKs for these packets arrive during the idle period of this round.

The above conditions guarantee that at least D packets arrive out of order. These conditions can be analytically stated and analyzed, and are confirmed in the simulations of Figures 3 and 4. The only difference between the simulations in the two scenarios is the instant at which RTT decrease takes place. In Figure 3, the RTT changes during the idle period of the round and hence, as discussed above, no TD takes place while in Figure 4 a TD takes place since the RTT decrease occurs during the busy period of the round and satisfies the above mentioned conditions for a false TD.

It is noteworthy that the amount of RTT decrease required to trigger false TD is very low for typical links, specifically for cases where $RTT > \frac{1}{\mu}$ (i.e. high bandwidth-delay product links), Since OldTahoe does not implement the TD option and relies only on TO's to detect packet losses, one would expect that OldTahoe is more robust than Reno/NewReno in the sense that the amount of RTT change required to trigger false retransmissions are higher. The amount of performance improvement is analyzed in the next section.

IV. Analysis of OldTahoe and Reno

In this section, we use the RTT alternating process model proposed in Section II to evaluate and compare between the performance of the OldTahoe and Reno versions of TCP.

A. Reference (Ideal) Goodput

In the following experiments, we measure the goodput defined as the number of non-repeated packets received at the destination per unit time. Goodput, and not throughput, is the appropriate figure of merit since in such environments, significant packet retransmissions takes place.

Now consider a (genie-aided) TCP session that somehow can identify timeouts and multiple duplicate ACK events that are an artifact of the high variability of the end-to-end RTT and hence does not respond by initiating spurious retransmissions. For an ideal (lossless) path, such a TCP session will never decrease its window size below w_m. Thus, the goodput of such a session for the two RTT values will be $\frac{w_m}{T_1}$ and $\frac{w_m}{T_2}$. In steady state, the proportion of time spent in state i is given by $\frac{E[V_i]}{E[V_1]+E[V_2]}$ where E denotes the expectation. Let R_I denote the ideal (steady state) goodput, then,

$$R_I = \frac{(\frac{E[V_1]}{T_1} + \frac{E[V_2]}{T_2})w_m}{E[V_1] + E[V_2]} \tag{2}$$

B. Effect of transition rate

In this section, we perform a number of simulations of TCP Reno and OldTahoe using a lossless path in which the *proportion* of time spent in each state is held constant while varying the duration of time spent in each state. Since the proportion of time spent in a state i is $\frac{E[V_i]}{E[V_1]+E[V_2]}$, R_I is independent of the rate of state transitions.

From Section II, we have assumed that $V_i = T_i + X_i$. Let $E[X_i] = hT_i, k > 0$, then $E[V_i] = (1+h)\,T_i$. Thus, h can be defined as a 'holding time factor'. Increasing h increases the average holding time in each state and hence decreases the transition frequency. Substituting in (2),

$$R_I = \frac{2w_m}{T_1 + T_2} \tag{3}$$

Figure 5 shows that while R_I is not affected by changes in h, the actual measured goodput seriously deteriorates with small h (i.e. high transition rates). We also notice that OldTahoe is less sensitive to changes in h than Reno as it does not rely on TD to infer packet loss.

Notice that for the case of lossy paths, the effect of packet loss will be an additional loss in goodput which will result in having the three curves in Figure 5 shifted downwards by an amount proportional to the packet loss rate.

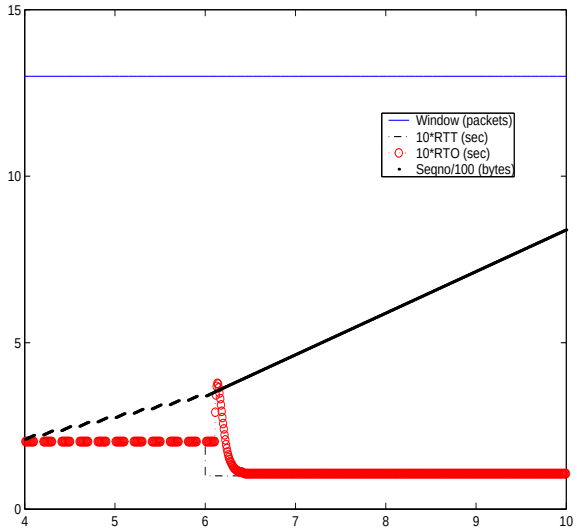

Fig. 3. Link Speed is 1Mbps (125 packets/sec), packet size is 1KBytes and $w_m = 26$ packets. Initially, $RTT = T_1 = 0.2$ sec. At $t = 6.0$ sec. (which falls within the idle period of a round), $RTT = T_2 = 0.1$ sec. There is no TD.

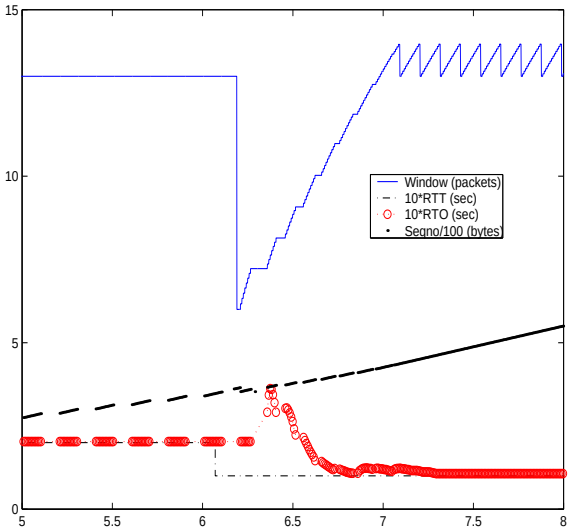

Fig. 4. Same as simulation in Fig. 3 but RTT is decreased at $t = 6.07$ within the busy period of a round. A TD takes place.

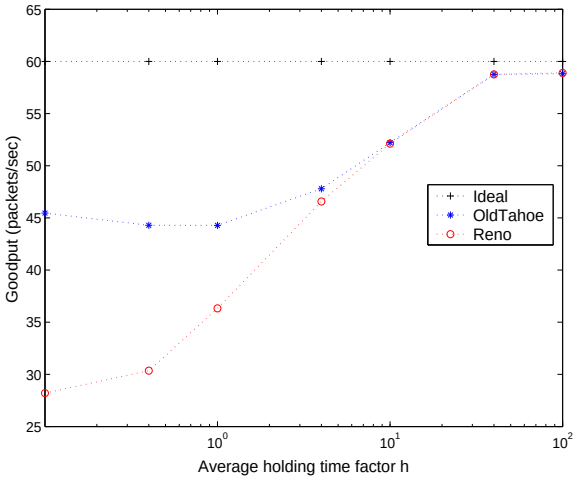

Fig. 5. $\mu = 1$ Mbps, $T_1 = 0.1$, $T_2 = 0.3$, $w_m = 12$, $p = 0$.

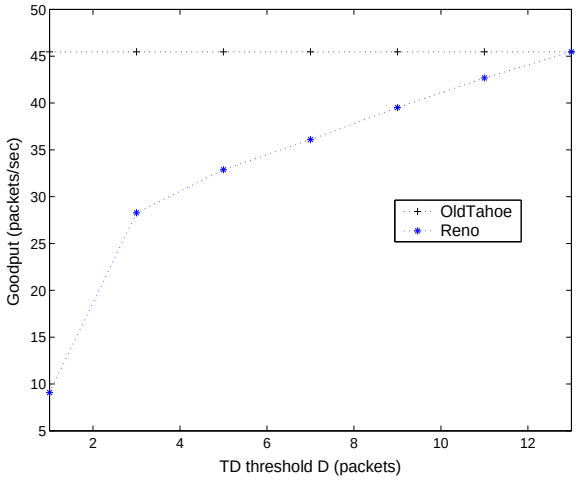

Fig. 6. $\mu = 1$ Mbps, $T_1 = 0.1$, $T_2 = 0.3$, $h = 0.1$, $w_m = 12$, $p = 0$.

C. *Effect of varying Reno's TD threshold*

Since OldTahoe's algorithm in essence is equivalent to Reno with a very large (e.g. infinite) DUPACK threshold D, it leads naturally to an investigation of the effect of varying D on Reno's performance. The results in Figure 6 for a lossless path show that there is an optimum value for D above which the goodput of Reno approaches that of OldTahoe. This choice of D depends on the end-to-end RTT behavior and underscores the fact that using a fixed value (such as 3 in most TCP implementations) is not optimal in any sense.

Figure 6 shows that for a lossless link, setting $D = w_m$ is the optimal choice. This is expected since in this case, no packet loss takes place and hence the best choice is never to trigger a TD. Figure 7 shows the results of four experiments $S_1, \ldots, S_4$. In each experiment, we vary D for a fixed (p, h) pair. We observe that the optimum D varies depending on the *relative* impact of abrupt RTT changes (i.e. smaller h) to the packet loss rate (i.e. higher p). For example, $D = 1$ for the case with relatively high p (e.g. S_4), $D = w_m$ for the case of relatively low h (e.g. S_1 and S_3) and $D = [4, 11]$ for S_2.

D. *Effect of varying RTO_{min}*

Previous evaluation of TCP RTO's estimation algorithm concluded that TCP throughput is highly dependent on RTO_{min} [9]. There are two opposing objectives in selecting RTO_{min}; the first is to decrease the possibility of spurious timeouts, which calls for using a large RTO_{min}, and the second is allowing TCP to detect TO's quickly that requires smaller RTO_{min}. Allman and Paxson recommend $RTO_{min} = max(1, 2G)$ with $G = 1$ tick; the motivation behind this rule is to limit the number of spurious retransmissions in case of fine timer granularity. Obviously, this

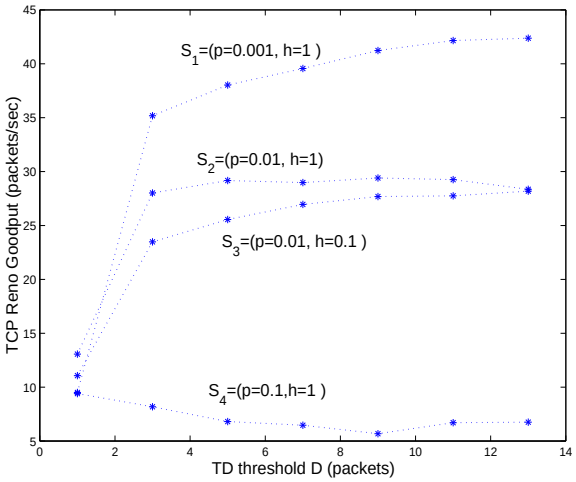

Fig. 7. $\mu = 1$ Mbps, $T_1 = 0.1$, $T_2 = 0.3$, $w_m = 12$.

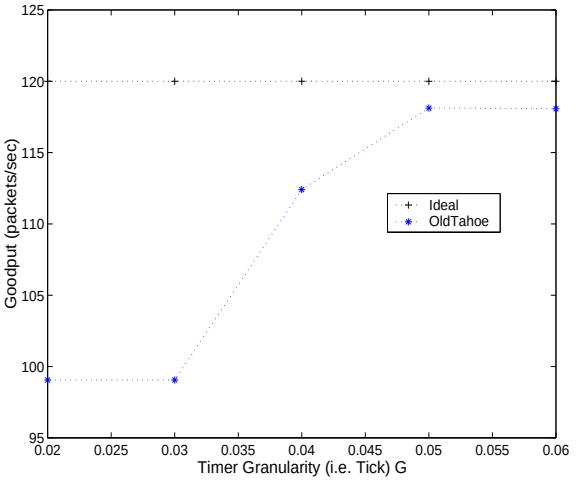

Fig. 8. $\mu = 2$ Mbps, $T_1 = 0.1$, $T_2 = 0.3$, $h = 10$, $w_m = 24$, $p = 0$.

only addresses one of the two objectives. For example, setting $RTO_{min} = 1sec$ would result in drastic performance degradation of a TCP Reno session operating over an end-to-end path with frequent TO's if the typical RTT values over the path are in the order of few milliseconds.

Figure 8 shows the results of a number of simulations using TCP OldTahoe for varying G. Since $RTO_{min} = 2G$, these simulations show the dependence of the performance on RTO_{min} as well. The simulations show that there exist an optimum value for G; similar to the choice of the optimum TD threshold, this value will depend on the end-to-end RTT behavior and the relative effect of the true packet loss rate p to the average holding time factor h between abrupt RTT changes.

V. Conclusion

In this paper, we explored the interaction between TCP's packet loss detection algorithms (timeouts and fast recovery) and abrupt end-to-end RTT changes for lossless and lossy paths.

We first focused on the effect of abrupt end-to-end RTT variations (without packet loss) and showed that small amount of changes may result in drastic deterioration of TCP's goodput. We analyzed the cases that will trigger spurious retransmissions by considering single step changes via simulations.

A number of experiments were performed in which the steady state goodput of a bulk TCP transfer is measured. We observed that the use of fixed values for TCP's packet loss detection thresholds result in suboptimal goodput in many situations. We showed that the optimal choice of the TD threshold D and the RTO_{min} depends on the statistics of the end-to-end RTT variations and packet loss.

Two main extensions of the work presented in this paper are currently under way. First, in this paper, we modeled the end-to-end process using an alternative Markov process. A generalization of this model to a semi-Markov process with a larger number of states allows for an analytic characterization of the steady state goodput. Second, we are currently working on modifications to TCP's packet detection algorithms that adaptively adjust the threshold values depending on feedback measurements implicitly provided by the network.

References

[1] V. Paxson, "End-to-end routing behavior in the Internet," *IEEE/ACM Transactions on Communications*, vol. 5, no. 5, pp. 601–15, October 1997.

[2] T. R. Henderson and R. H. Katz, "Network simulation for LEO satellite networks," in *Proceedings of 18th AIAA International Communications Satellite Systems Conference (ICSSC)*, April 2000.

[3] M. Allman, J. Griner, and A. Richard, "TCP behavior in networks with dynamic propagation delay," in *Proceedings of Globecom 2000*, November 2000.

[4] S. McCanne and S. Floyd, "ns-Network Simulator," *URL http://www-mash.cs.berkeley.edu/ns/*, 1995.

[5] V. Jacobson, "Congestion avoidance and control," in *Proceedings of ACM SIGCOMM'88*, 1988, pp. 314–329.

[6] V. Jacobson, R. Braden, and D. Borman, "TCP extensions for high performance," Tech. Rep. RFC 1323, May 1992.

[7] J. Postel, "Transmission control protocol," RFC 793, IETF, September 1997.

[8] P. Karn and C. Partridge, "Improving round-trip time estimates in reliable transport protocols," in *Proceedings of ACM SIGCOMM'87*, 1987, pp. 2–7.

[9] M. Allman and V. Paxson, "On estimating end-to-end network path properties," in *Proceedings of ACM SIGCOMM'99*, September 1999.