Analytic Models and Comparative Study of the Latency and Steady-State Throughput of TCP Tahoe, Reno and SACK

B. Sikdar, S. Kalyanaraman and K. S. Vastola
Dept. of ECSE, Rensselaer Polytechnic Institute
Troy, NY 12180 USA
email: {bsikdar,shivkuma,vastola}@networks.ecse.rpi.edu

Abstract— **Continuing the process of improvements made to TCP through the addition of new algorithms in Tahoe and Reno, TCP SACK aims to provide robustness to TCP in the presence of multiple losses from the same window. In this paper we present analytic models to estimate the latency and steady-state throughput to TCP Tahoe, Reno and SACK and validate our models using both simulations and TCP traces collected from the Internet. In addition to being the first models for the latency of finite Tahoe and SACK flows, our model for the latency of TCP Reno gives a more accurate estimation of the transfer times than existing models. Our models show that under the losses introduced by the droptail queues which dominate most routers in the Internet, current implementations of SACK fail to provide adequate protection against timeouts and a loss of roughly more than half the packets in a round will lead to timeouts. We also show that with independent losses, SACK performs better than Tahoe and Reno and as losses become correlated, Tahoe can outperform both Reno and SACK.**

I. Introduction

Early TCP implementations used a go-back-n model and required the expiration of a retransmission timer to recover any loss. TCP Tahoe added added the *slow-start, congestion avoidance* and *fast retransmit* algorithms to TCP [7]. TCP Reno retained the new algorithms of TCP Tahoe while adding *fast recovery* to the implementations [8]. TCP SACK allows receivers to ACK out of sequence data and is aimed at eliminating the timeouts which arise in TCP Reno due to multiple losses from the same window [1], [2]. The most recent proposal for adding SACK to TCP is in [12].

In this paper we first present analytic models for estimating the latencies and steady state throughput of TCP Tahoe, Reno and SACK. Our models are validated using both traces collected from the Internet as well as simulations. We then compare the performance of these versions of TCP under independent and correlated loss scenarios. Existing analytic models for TCP focus mainly on TCP Reno [4], [6], [10], [11], [14], [15] as it the most widely implemented version of TCP. Also, these models are for the steady state throughput of infinite TCP connections. In [9], models for the steady state throughput of Tahoe, Reno and NewReno are presented assuming an independent loss model without delayed ACKs. In [3] and [16] models for estimating the latency of TCP Reno are presented for correlated and independent losses respectively.

Though TCP Reno has been modeled extensively no literature exists on modeling the latencies of finite TCP flows using Tahoe or SACK. In [5] simulations using the the simulator *ns* are used to compare the performance of the three versions with four sets of simulations. This paper, in addition to being the first to present models for the latency of TCP Tahoe and SACK, also presents a model for the latency of TCP Reno gives yields more accurate results compared to existing models. We also show a serious drawback in current SACK implementations (which use a variable `pipe` which can lead to severe performance degradations. We show how TCP SACK can lead to timeouts even on the receipt of 3 duplicate ACKs and derive the precise conditions under which these timeouts occur.

Another important contribution of this work is to show that Tahoe can outperform both Reno and SACK under correlated losses. In [5] simulation scenarios were used show that both Tahoe and SACK outperform Reno in the presence of multiple losses in a window. Also, SACK performs better than Tahoe in the examples considered. However, as our analysis shows, with correlated losses which arise from the droptail queues dominating the routers in the Internet, Tahoe can perform better than both Reno and SACK. But, using simulations we show that if the loss scenario is changed to an independent model, both SACK and Reno outperform Tahoe. Thus, proper determination of the loss model is of utmost importance in determining TCP performance.

The rest of the paper is organized as follows. In Section II we present the assumptions made in our models. Section III presents the models for the latencies and their validation while Section IV presents the models for the steady state throughput. In Section V we compare the performance of the three versions of TCP in terms of their latencies and their steady-state throughput. Finally we present a discussion on the results and concluding remarks in Section VI.

II. Assumptions

We follow assumptions on the network and the TCP sender and receiver similar to those in [3], [15], [16]. Our models account only for the delays arising from TCP's performance and we ignore the delays arising at the endpoints from factors like buffer limitations. We assume that the sender transmits full sized segments as fast as its congestion window allows and that the receiver advertises a consistent flow control window, W_{max}. We assume that the receiver uses the delayed acknowledgment scheme specified in RFC 2581. As in [3], we do not account for the effects of Nagle's algorithm and silly window avoidance.

This work supported in part by DARPA under contract numbers F19628-98-C-0057 and F30602-00-2-0537, NSF contract AWI 9806660 and by MURI contract F49620-97-1-0382 through AFOSR.

We model the latency of TCP flows in terms of "rounds" as defined in [15]. A round begins with the transmission of a window of packets and ends on the receipt of an ACK for one of these packets.

In this paper we assume the correlated loss model of [15] which is better suited for the droptail queues currently prevalent in the Internet. In this model, a packet in a round is lost independently of losses in other rounds. However, losses within a round are correlated and all packets following the first packet to be lost in round are also assumed to be lost. The probability that any given packet is lost is denoted by p. Finally we assume that the time to transmit all the packets is much smaller than the duration of the round and that the duration of the round is independent of the window size.

III. Models for TCP Latency

Our approach towards modeling the latency TCP flows is to estimate the transfer time given that the flow experiences a given number of loss indications. We break the modeling in three parts: flow with no losses, a single loss indication and multiple loss indications. Expressions for the latency are derived as a function of the transfer size N (in number of packets), the packet loss probability p and the RTT.

A. Connection Establishment

A TCP connection begins with a three-way handshake. During this process, if either host does not receive the ACK it is expecting within a timeout period T_s, it retransmits its SYN and then waits twice as long for an ACK. Following [3], the expected duration of the connection setup time can be approximated as

$$t_{setup} = RTT + 2T_s\left((1-p)/(1-2p) - 1\right) \tag{1}$$

B. The Slow-start Phase

TCP starts its transmission in the slow-start phase. With delayed ACKs, the receives sends one ACK for every two packets that it receives or if the delayed ACK timer expires. In [16] the authors proposed a model for the $cwnd$ increase pattern which accounts for the intricacies of the delayed ACK timer by modeling the expected number of packets sent in each round. This results in a more accurate model of the slow start phase as compared to 1.5^n packets in the n^{th} round as used in [3]. The number of packets transmitted in the n^{th} round according to the model of [16] is given by

$$w(n) = \left\lfloor 2^{\frac{n-1}{2}} + 2^{\frac{n-2}{2}} \right\rfloor \tag{2}$$

The number of packets transmitted in the first k rounds of the slow start phase is then given by

$$pkt(k) = \sum_{n=1}^{k} w(n) = \left\lfloor 2^{\frac{n+1}{2}} + 3(2)^{\frac{4n-3}{8}} - 2 - 3/\sqrt{2} \right\rfloor \tag{3}$$

Note that after the first packet of a flow is sent, the receiver has to wait until the delayed ACK timer expires and an ACK is sent, increasing the $cwnd$ to 2. This delay due to the delayed ACK, t_{dack}, is 100ms for UNIX and 150ms for Windows platforms.

C. Timeouts and Congestion Avoidance

When TCP flows experience a loss, the window reduces and then increases again either in the slow-start or congestion avoidance mode. The average duration of a timeout is given by [15]

$$E[TO] = T_o \frac{1 + p + 2p^2 + 4p^3 + 8p^4 + 18p^5 + 32p^6}{1-p} \tag{4}$$

where T_o is the duration of time the sender waits before retransmitting the first lost packet. Once the flow reaches the congestion avoidance phase, the $cwnd$ increases linearly till it reaches W_{max}, increasing by 1 every two RTTs. The number of rounds required to transmit a packets in the congestion avoidance mode with the initial value of $cwnd = b$ is obtained by solving $a = \sum_{i=1}^{k}(b + \lfloor \frac{i-1}{2} \rfloor)$ for k and is given by

$$t_{lin}(a,b) = \begin{cases} \left\lceil \frac{a - x(x+1) + b(b-1)}{x+1} \right\rceil + 2(x - b + 1) & \text{if } a \le N_{lin} \\ \left\lceil \frac{a - W_{max}(W_{max}+1) + b(b-1)}{W_{max}} \right\rceil & \text{otherwise} \\ +2(W_{max} - 2b + 1) & \end{cases} \tag{5}$$

where $N_{lin} = W_{max}(W_{max}+1) - b(b-1)$ and denotes the number of packets that can be sent before $cwnd$ reaches W_{max} and $x = \lfloor(-1 + \sqrt{1 + 4(a + b(b-1))})/2\rfloor$.

D. Flows Without Losses

If the TCP flow does not experience any losses, the behavior of all the three versions of TCP under consideration will be identical. The number of round required by the TCP flow to reach a $cwnd$ of w_{max} can be computed from Eqn. (2) and is given by

$$n_{wm} = \left\lfloor 2\log_2\left(\frac{2W_{max}}{1+\sqrt{2}}\right) \right\rfloor \tag{6}$$

The number of packets transmitted when $cwnd$ reaches W_{max}, N_{exp}, is given by

$$N_{exp} = \left\lfloor 2^{\frac{n_{wm}+1}{2}} + 3(2)^{\frac{4n_{wm}-3}{8}} - 2 - 3/\sqrt{2} \right\rfloor + W_{max} \tag{7}$$

The time to transfer N packets, $N < N_{exp}$, can be obtained by solving $N = \sum_{n=1}^{k} pkt(n)$ for k and is given by

$$t_{nl}(N) = \begin{cases} \left\lceil 2\log_2\left(\frac{2N+4+3\sqrt{2}}{2\sqrt{2}+3(2)^{\frac{5}{8}}}\right) \right\rceil RTT, & \text{if } N \le N_{exp} \\ \left\lceil n_{wm} + \left\lceil \frac{N - N_{exp}}{W_{max}} \right\rceil \right\rceil RTT, & \text{otherwise} \end{cases} \tag{8}$$

E. TCP Tahoe

When a Tahoe flow does not experience any losses, the time taken to transfer N packets is given by Eqn. (8). We now consider the cases when a Tahoe flow experiences a single or multiple loss indications.

E.1 Single Loss

We first present some expressions which will be used frequently in the derivations. To find the $cwnd$ of the round when the i^{th} packet is transmitted, $cwnd_i^0$, we first find the number of rounds it takes to transmit i packets, $r(i)$, assuming there is no effect of window limitation. $r(i)$ can be calculated as in Eqn. (8) and is given by

$$r(i) = \left\lceil 2\log_2\left(\frac{2i + 8.243}{7.455}\right)\right\rceil \quad (9)$$

If $r(i) \geq n_{wm}$, we know that $cwnd = W_{max}$. For all other cases, $cwnd < W_{max}$ and is given by Equation (2). Then, $cwnd_i^0$ is given by

$$cwnd_i^0 = \begin{cases} W_{max} & \text{if } r(i) > n_{wm} \\ \left\lfloor 2^{\frac{r(i)-1}{2}} + 2^{\frac{r(i)-2}{2}} \right\rfloor & \text{otherwise} \end{cases} \quad (10)$$

The sequence number of the last packet transmitted in this round, $n_{max}(i)$, is obtained using Equation (3) and is given by

$$n_{max}(i) = \begin{cases} N_{exp} + \left\lceil \frac{\max(0, i - N_{exp})}{W_{max}} \right\rceil W_{max} & \text{if } r(i) > n_{wm} \\ \left\lfloor 2^{\frac{r(i)+1}{2}} + 3(2)^{\frac{4r(i)-3}{8}} - 2 - \frac{3\sqrt{2}}{2} \right\rfloor & \text{otherwise} \end{cases} \quad (11)$$

From our correlated loss model, if packet i is lost, then all the packets following it in the same round, i.e., $i+1$ to $n_{max}(i)$ are also lost and we denote the number of losses by $nloss$. Also, in this round $i - n_{max}(i) + cwnd_i^0 - 1$ packets are transmitted correctly before the i^{th} packet. Since the receiver used delayed ACKs, we can thus expect ACKs for half of these packets, each of which increases $cwnd$ by one. We denote the $cwnd$ in the round which now follows by $cwnd_i^1$ and the number of packets transmitted in it by $nrnd_i^1$. On detecting the loss the sender retransmits the lost packets and sets its $ssthresh$ to $n = \max\{2, \lceil cwnd_i^1/2 \rceil\}$. Let the duration of the slow start phase following a loss indication in TCP Tahoe be denoted by $r(n)$ and the number of packets transmitted in these rounds be denoted by k'. Each of the quantities is then given by

$$\begin{aligned} nloss &= n_{max}(i) - i + 1 \\ cwnd_i^1 &= \min\{W_{max}, cwnd_i^0 + \lceil (cwnd_i^0 - nloss)/2 \rceil\} \\ nrnd_i^1 &= cwnd_i^1 - nloss \\ k' &= \left\lfloor 2^{\frac{r(n)+1}{2}} + 3(2)^{\frac{4r(n)-3}{8}} - 2 - 3/\sqrt{2} \right\rfloor \end{aligned} \quad (12)$$

and $r(n)$ is obtained from Eqn. (9). Now if $nrnd_i^1 \geq 3$, the sender will get at least three duplicate ACKs for packet i and enter fast retransmit and recovery. Otherwise, the connection times out. When the congestion avoidance phase begins, $nrnd_i^1 + k'$ additional packets have been correctly transferred. The transfer time for the remaining $a = N - nrnd_i^1 - k' - i + 1$ packets to be transmitted in the linear increase phase can be found using Eqn. (5) with $b = n$. Also, the time to transmit the first $i - 1$ packets can be obtained using the expression for the no loss case. The time to transmit N packets with a loss indication at packet i is then given by

$$t_{sl}(N) = \begin{cases} [t_{nl}(i) + r(n) + 1 & \text{if } nrnd_i^1 \geq 3 \\ +t_{lin}(a,b)]\, RTT & \\ [t_{nl}(i) + r(n) + t_{lin}(a,b) + & \text{else} \\ I(nrnd_i^1 > 0) + E[TO]]\, RTT & \end{cases} \quad (13)$$

E.2 Multiple Losses

Let there be M loss indications in a flow of N packets, the second of which occurs at packet number m. The first $m-1$ packets have one loss indication and the time to transmit these packets can be obtained from the results of the previous subsection. The average number of packets between two successive losses is given by

$$D_a = \frac{N - m + 1}{M - 1} \quad (14)$$

To obtain the transfer time for the last $N - m + 1$ packets, we first compute the average time to transmit D_a packets. After the first loss, we approximate the possible range of values of $cwnd$ when the subsequent losses occur by $1, \cdots, \min\{W_{max}, \lceil \frac{-1+\sqrt{1+16D_a}}{2} \rceil\}$. Also, to keep the analysis tractable, we assume that each of these possible values of $cwnd$ and the position of the lost packet within a $cwnd$ are equally likely when the loss occurs.

Now consider the flow with $cwnd = h$ where the loss indication occurs at the j^{th} packet of the round. We want to find the time to transmit D_a packets correctly following the j^{th} packet. For analytic tractability, we now assume that the flow was in the congestion avoidance mode when this loss indication occurred, as it will subsequently be following the first loss indication. The number of packets lost in this round is now given by $h - j + 1$ while $j - 1$ packets are transmitted successfully. When the loss is detected, $ssthresh$ is set to $n = \max\{2, \lceil h/2 \rceil\}$. In case of a timeout we need $r(n)$ rounds with k' packets transmitted in them till the flow enters the congestion avoidance mode again. Using the same definition for the quantities $nloss$, $cwnd_j^0$, $cwnd_j^1$, $nrnd_j^1$, $r(n)$ and k' as for the single loss case, we have

$$\begin{aligned} cwnd_j^0 &= h \\ nloss &= h - j + 1 \\ cwnd_j^1 &= h \\ nrnd_j^1 &= cwnd_j^1 - nloss = j - 1 \end{aligned} \quad (15)$$

and $r(n)$ and k' are obtained from Eqn. (9) and (12) respectively. Now if $nrnd_j^1 \geq 3$, we get enough duplicate ACKs to lead to a fast retransmit. Again since $a = D_a - k' - j + 1$ packets remain to be transmitted in the congestion avoidance phase, the time to transmit D_a packets after the loss indication is thus

$$t_{M\_loss}(D_a) = \begin{cases} [r(n) + 1 & \text{if } nrnd_i^1 \geq 3 \\ +t_{lin}(a,n)]\, RTT & \\ [t_{lin}(a,n) + E[TO] + & \text{otherwise} \\ r(n) + I(j - 1 > 0)]\, RTT & \end{cases} \quad (16)$$

The expected duration to transmit the D_a packets can now be obtained by averaging Eqn. (16) for the possible value of h and j. The time to transmit the N packets with M loss indications is then

$$t_{ml}(N) = E\{t_{sl}(m-1)\} + (M-2)E\{t_{M\_loss}(D_a)\} \quad (17)$$

File Size	$p = 0.001$		$p = 0.005$		$p = 0.010$	
	sim.	ana.	sim.	ana.	sim.	ana.
1 KB	0.28	0.28	0.47	0.47	0.94	1.04
4 KB	0.39	0.37	0.75	0.76	1.56	1.47
8 KB	0.72	0.68	1.23	1.27	1.78	1.82
16 KB	0.82	0.81	1.69	1.73	2.73	2.68

File Size	$p = 0.001$		$p = 0.005$		$p = 0.010$	
	sim.	ana.	sim.	ana.	sim.	ana.
1 KB	0.36	0.35	0.63	0.64	1.02	1.17
4 KB	0.63	0.60	1.08	1.12	1.81	1.74
8 KB	0.99	0.94	1.48	1.52	2.62	2.69
16 KB	1.42	1.37	2.16	2.21	3.47	3.49

TABLE I

TCP TAHOE TRANSFER TIMES. $RTT = 100ms$ (TOP) AND $RTT = 200ms$ (BOTTOM), $MSS = 1.4KB$, $W_{max} = 24$.

E.3 The Expected Transfer Time

Combining the results of the previous sections, the expected transfer time for a flow of N packets is given by

$$T_{transfer}(N) = t_{setup} + t_{dack} + (1-p)^N t_{nl}(N) + p(1-p)^{N-1} E\{t_{sl}(N)\} + t_{ml}(N) \qquad (18)$$

where t_{setup}, $t_{nl}(N)$, $t_{sl}(N)$ and $t_{ml}(N)$ are defined in Eqns. (1), (8), (13) and (17) respectively.

In Tables I we present the comparison of the results from our model with those from simulations using *ns*. A 2-state Markov error model was used to model the correlated loss process. We note the close agreement between the analytic and simulation results.

F. TCP Reno

TCP Reno adds the fast-recovery algorithm to TCP Tahoe which can results in substantial improvements in the presence of single packet losses. However with multiple packet losses, TCP Reno has to resort to timeouts to recover the losses and its performance degrades. When the Reno flow does not experience any losses, the transfer time is obtained using Eqn. (8).

F.1 Single Loss

Following the notation used for the Tahoe model we again obtain $cwnd_i^0$ from Eqn. (10) and $nloss$, $cwnd_i^1$ and $nrnd_i^1$ from Eqn. (12). However, since Reno can recover multiple lost packets (with one recovery per round), we can have another round of $nrnd_i^2$ packets with $cwnd$ denoted by $cwnd_i^2$ before a timeout may be detected. If $nrnd_i^1 \geq 3$, we have three duplicate ACKs in the next round resulting in a $cwnd$ of

$$cwnd_i^2 = \min\{W_{max}, \lceil cwnd_i^1/2 \rceil + nrnd_i^1\} \qquad (19)$$

The number of packets unacknowledged at the end of first round following the round with the losses is $cwnd_i^1$. Thus if $cwnd_i^2 > cwnd_i^1$

$$nrnd_i^2 = cwnd_i^2 - cwnd_i^1 \qquad (20)$$

With TCP Reno, a single packet loss in a window of less than 4 ($tcprextthresh + 1$), two or more losses in windows between 4 and 8 ($tcprextthresh + 1$ and $2(tcprextthresh + 1)$) and three or more losses for higher windows lead to timeouts [13]. For all other cases, the losses can be recovered using fast retransmissions. Let us first consider the simpler case when the losses are recovered using fast retransmissions. Since $nrnd_i^1 + nrnd_i^2$ new packets along with the $nloss$ retransmitted packets are sent before the congestion avoidance phase begins, we only have $a = N - nloss - nrnd_i^1 - nrnd_i^2 - i + 1$ packets to send in congestion avoidance phase. Also, since each recovered loss reduces $ssthresh$ by half, we have $n = \max\{2, \lceil cwnd_i^1/nloss \rceil\}$. Then

$$t_{sl}(N) = [t_{nl}(i) + nloss + 1 + t_{lin}(a,n)]\, RTT \qquad (21)$$

If the flow times out while recovering the the losses, we can have between 0 and 3 additional rounds of packets where the first couple of losses may be recovered. If the first packet of the round is lost then we do not have any additional rounds and the flow directly times out. If $nrnd_i^1$ is one or two, then we have another round with $nrnd_i^1$ packets before the flow times out. For other cases we have at least one packet which is recovered using fast recovery. In this round $nrnd_i^2$ additional packets are sent along with the retransmitted packet. Also, if $nrnd_i^2 \geq 3$ we have a third additional round with another fast retransmit. The flow now inevitably times out. We denote the number of additional packets transmitted before the timeout by k'' and during the slow start phase following the timeout by k'. We also denote the number of rounds before the timeout starts by t_{TO} and the duration of the slow start phase by $r(n)$. After some algebraic simplifications we then have

$$k'' = \begin{cases} nrnd_i^1 & \text{if } nrnd_i^1 < 3 \\ \max\{cwnd_i^1, cwnd_i^2\} - nrnd_1^1 - 1 & \text{if } nrnd_i^2 < 3 \\ \max\{cwnd_i^1, cwnd_i^2\} - nrnd_1^1 & \text{otherwise} \end{cases} \qquad (22)$$

$$t_{TO} = \begin{cases} I(nrnd_i^1 > 0) & \text{if } nrnd_i^1 < 3 \\ 2 & \text{if } nrnd_i^2 < 3 \\ 3 & \text{otherwise} \end{cases} \qquad (23)$$

$$n = \begin{cases} \max\{2, \lceil cwnd_i^1/2 \rceil\} & \text{if } nrnd_i^1 < 3 \\ \max\{2, \lceil cwnd_i^1/4 \rceil\} & \text{if } nrnd_i^2 < 3 \\ \max\{2, \lceil cwnd_i^1/8 \rceil\} & \text{otherwise} \end{cases} \qquad (24)$$

with $r(n)$ and k' given by Eqns. (9) and (12) respectively. Since we only have to transmit $a = N - k' - k'' - i + 1$ packets in the congestion avoidance phase, the time to transmit N packets is then given by

$$t_{sl}(N) = [t_{nl}(i) + t_{TO} + r(n) + E[TO] + t_{lin}(a,n)]\, RTT \qquad (25)$$

F.2 Multiple Losses

We follow the same approach as used in modeling Tahoe flows with multiple losses and use the same expressions for finding D_a and the range of possible $cwnd$ values. Now consider the flow with $cwnd = h$ where the loss indication occurs at the j^{th} packet of the round. Using the same definition for the quantities $nloss$, we can use Eqn. (15) to find $cwnd_j^0$, $cwnd_j^1$, $nloss$ and $nrnd_j^1$. Also, $cwnd_j^2$ and $nrnd_j^2$ correspond to $cwnd_i^2$ and $nrnd_i^2$ of the single loss case and given by Eqns. (19) and (20) respectively. When all the losses are successfully recovered using fast retransmits, $a = D_a - nloss - nrnd_j^1 - nrnd_j^2$ packets

File Size	$p = 0.010$ sim.	$p = 0.010$ ana.	$p = 0.005$ sim.	$p = 0.005$ ana.	$p = 0.001$ sim.	$p = 0.001$ ana.
1 KB	0.27	0.26	0.53	0.54	0.96	0.96
4 KB	0.42	0.41	0.83	0.85	1.63	1.62
8 KB	0.63	0.62	1.18	1.14	2.05	2.08
16KB	0.80	0.80	1.34	1.37	2.66	2.68

File Size	$p = 0.010$ sim.	$p = 0.010$ ana.	$p = 0.005$ sim.	$p = 0.005$ ana.	$p = 0.001$ sim.	$p = 0.001$ ana.
1 KB	0.38	0.36	0.70	0.65	1.00	1.06
4 KB	0.65	0.63	1.06	1.07	2.08	2.10
8 KB	0.97	0.87	1.62	1.64	2.47	2.52
16 KB	1.41	1.31	2.16	2.17	3.72	3.71

TABLE II

TCP RENO TRANSFER TIMES. $RTT = 100ms$ (TOP) AND $RTT = 200ms$ (BOTTOM), $MSS = 1.4KB$, $W_{max} = 24$.

are transmitted during congestion avoidance which begins with a $cwnd$ of $n = \max\{2, \lceil cwnd_i^1/nloss \rceil\}$. The time to transmit the D_a packets can now be expressed as

$$t_{M\_loss}(D_a) = [nloss + 1 + t_{lin}(a, n)]\, RTT \qquad (26)$$

For flows resorting to timeouts, we use the same notation for k'', k', t_{TO} n and $r(n)$ in the single loss case, and obtain them using Eqns. (22), (12), (23), (24) and (9) respectively. In the congestion avoidance phase we are now left with $a = D_a - k' - k''$ to be transmitted. The time to transmit the D_a packets can now be expressed as

$$t_{M\_loss}(D_a) = [E[TO] + t_{TO} + r(n) + t_{lin}(a, n)]\, RTT \qquad (27)$$

Following the same expectation operations as for TCP Tahoe, the time to transmit the N packets with M loss indications is then

$$t_{ml}(N) = E\{t_{sl}(m-1)\} + (M-2)E\{t_{M\_loss}(D_a)\} \quad (28)$$

F.3 The Expected Transfer Time

Combining the results of the previous sections, the expected transfer time for a flow of N packets is given by

$$\begin{aligned} T_{transfer}(N) &= t_{setup} + (1-p)^N t_{nl}(N) + t_{dack} + \\ & \quad p(1-p)^{N-1} E\{t_{sl}(N)\} + t_{ml} \qquad (29) \end{aligned}$$

where t_{setup}, $t_{nl}(N)$ and $t_{ml}(N)$ are defined in Eqns. (1), (8) and (28) respectively and $t_{sl}(N)$ is defined in Eqns. (21) and (25).

In Tables II we present the comparison of the results from our model with those from *ns* simulations. Again we have a close agreement between the analytic and simulation results. For TCP Reno we also validate our model by comparing the delays predicted by our model with measurements of actual TCP transfers over the Internet. These measurements were done for files with randomly generated sizes transferred between machines at RPI Troy, NY and Ohio State University, Columbus OH, MIT, Boston MA, University of California, Los Angeles CA and University of Pisa, Pisa Italy. For space restrictions, we only show the results for transfers to Los Angeles in Fig. 1. The results for the others are similar. We also plot the results from the model given in [3]. For short transfers, our results are clearly much closer to the measured average while for large transfers there is no appreciable difference between the two models.

G. TCP SACK

TCP SACK is specially aimed at eliminating timeouts in the presence of multiple losses. However, we now show that with correlated losses, SACK fails to achieve this goal and timeouts occur quite frequently. For our analysis, we assume the SACK implementation of [5]. We use the same definitions from the previous subsections and obtain $cwnd_i^0$ from Eqn. (10) and $nloss$, $cwnd_i^1$ and $nrnd_i^1$ from Eqn. (12).

We first identify the cases in which TCP SACK can lead to a time out. It is easy to see that if $nrnd_i^1 < 3$ (i.e. $tcprextthresh$) the sender does not receive enough duplicate acknowledgments and resorts to timeouts. However, in the presence of correlated losses, there is another factor which leads to most of the timeouts in current SACK implementations and this is the variable `pipe`. In TCP SACK, `pipe` controls the transmission of new or retransmitted packets during fast recovery. The sender sends new or retransmitted packets only if the value of `pipe` is less than $cwnd$. When the third duplicate ACK is received and the first packet is retransmitted, `pipe` is initialized to $cwnd_i^1 - 3$ and $cwnd$ is set to $cwnd_i^1/2$. For each of the additional duplicate ACKs resulting from the rest of the $nrnd_i^1 - 3$ packets `pipe` is decremented by one resulting in

$$\texttt{pipe} = cwnd_i^1 - nrnd_i^1 = nloss \qquad (30)$$

If `pipe` $\geq cwnd$ no more packets are transmitted. The partial ACK corresponding to the first retransmitted packet comes, `pipe` is decremented by two. Now if `pipe` is still greater than equal to $cwnd$ the flow will timeout. The condition for timeouts resulting due to the implementation of `pipe` can then be expressed as

$$nloss - 2 \geq \lceil cwnd_i^1/2 \rceil \qquad (31)$$

This roughly translates to: if more than half of the packets in a round are lost, SACK is unable to recover them without a timeout. Thus with correlated losses, Tahoe can out perform SACK since it is the most conservative and starts retransmitting all lost packets on the receipt of 3 duplicate ACKs.

G.1 Single Loss

Let us first consider the case when the flow resorts to a timeout to recover the losses, which happens if $nrnd_i^1 < 3$ or the if the condition of Eqn. (31) is satisfied. If $nrnd_i^1 > 0$ we have at least one round following the round with losses and if $nrnd_i^1 \geq 3$ we have one additional round where one packet is recovered with a fast retransmit. A fast retransmit results in the $cwnd$ being halved and the value of $cwnd$ and the number of packets transmitted in the next round, $nrnd_i^2$ are given by

$$\begin{aligned} cwnd_i^2 &= \lceil cwnd_i^1/2 \rceil \\ nrnd_i^2 &= 1 + \max\{0, cwnd_i^2 - \texttt{pipe}\} \qquad (32) \end{aligned}$$

Also, when the loss is detected, $ssthresh$ is set to $n = \min\{2, \lceil cwnd_i^1/2 \rceil\}$ and the number of packets transmitted in the slow start phase, k' is obtained from Eqn. (12). The number of packets remaining to be transmitted in the congestion avoidance mode is thus $a = N - nrnd_i^1 - nrnd_i^2 - k' - i + 1$. The time to transmit N packets with one loss indication resulting in

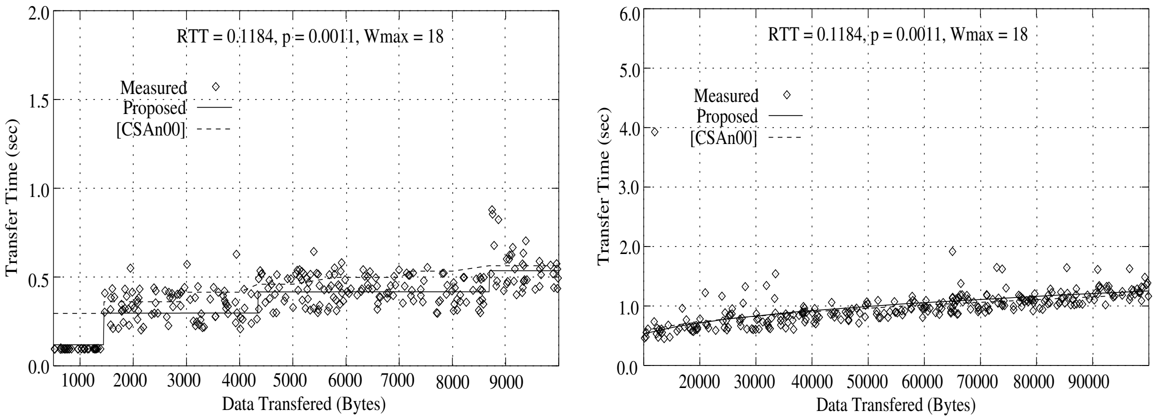


Fig. 1. Comparison of latency of TCP Reno transfer between Troy, NY and Los Angeles, CA.

a timeout is thus given by

$$t_{sl}(N) = \left[t_{nl}(i) + E[TO] + I(nrnd_i^1 > 0) + I(nrnd_i^1 > 3) + t_{lin}(a,n)\right] RTT \quad (33)$$

In the case that all the losses are recovered without any timeout, in each round the partial ACKs decrease pipe by two. Since $nrnd_i^2$ packets are retransmitted in the round following the receipt of the duplicate ACKs, the number of losses which remain to be transmitted, rl, is given by $rl = nloss - nrnd_i^2$ The number of rounds spent in this exponential phase till all the losses are recovered, $r(n)$ and the packets transmitted in these $r(n)$ rounds, k', can be easily shown to be given by

$$\begin{aligned} r(n) &= \left\lceil \log_2\left(\frac{rl}{nrnd_i^2} + 1\right)\right\rceil \\ k' &= nrnd_i^2(2^{r(n)} - 1) \end{aligned} \quad (34)$$

We then have $a = N - nrnd_i^1 - nrnd_i^2(2)^{r(n)} - i + 1$ packets to be transmitted in the congestion avoidance mode resulting in a total transfer time of

$$t_{sl}(N) = [t_{nl}(i) + 2 + r(n) + t_{lin}(a,n)]RTT \quad (35)$$

G.2 Multiple Losses

Following the same approach as used for Tahoe and Reno and using the same definitions, we can again use Eqn. (15) to obtain $nloss$, $cwnd_j^0$, $cwnd_j^1$, $nrnd_j^1$. The value of pipe after the receipt of the duplicate ACKs can now be found using Eqn. (30). The quantities $cwnd_j^2$ and $nrnd_j^2$ correspond to $cwnd_i^2$ and $nrnd_i^2$ of the single loss case and are similarly obtained using Eqn. (32). We first consider the case when the flow times outs. When the loss is detected, $ssthresh$ is set to $n = \min\{2, \lceil cwnd_i^1/2\rceil\}$ and the number of packets transmitted in the slow start phase, k' is again obtained from Eqn. (12). The number of packets remaining to be transmitted in the congestion avoidance mode is then $a = D_a - nrnd_j^1 - nrnd_j^2 - k'$ and the time to transmit D_a packets is thus

$$t_{M\_loss}(D_a) = \left[E[TO] + r(n) + I(nrnd_j^1 > 0) + I(nrnd_j^1 > 3) + t_{lin}(a,n)\right] RTT \quad (36)$$

In the case that the losses are recovered without any timeouts, the number of losses which remain to be transmitted after the first round of retransmissions is $rl = nloss - nrnd_j^2$. As for the single loss case, the time to transmit the remain losses, $r(n)$, and the number of packets transmitted in these rounds, k' are given by Eqn. (34). Now, only $a = D_a - nrnd_j^1 - nrnd_j^2(2)^{r(n)}$ packets remain to be transmitted in the congestion avoidance mode and the transfer time for D_a packets is given by

$$t_{M\_loss}(D_a) = [2 + r(n) + t_{lin}(a,n)]\, RTT \quad (37)$$

Following the same expectation operations as in the previous two models, the time to transmit the N packets with M loss indications is given by

$$t_{ml}(N) = E\{t_{sl}(m-1)\} + (M-2)E\{t_{M\_loss}(D_a)\} \quad (38)$$

G.3 The Expected Transfer Time

Combining the results of the previous sections, the expected transfer time for a flow of N packets is given by

$$T_{transfer}(N) = t_{setup} + (1-p)^N t_{nl}(N) + t_{dack} \, p(1-p)^{N-1} E\{t_{sl}(N)\} + t_{ml}(N) \quad (39)$$

where t_{setup}, $t_{nl}(N)$ and $t_{ml}(N)$ are defined in Eqns. (1), (8) and (38) respectively and $t_{sl}(N)$ is defined in Eqns. (33) and (35).

In Table III we present the comparison of the results from our model with those from simulations using *ns*. Again, our results match very well with the simulation results.

File	$p = 0.001$		$p = 0.005$		$p = 0.010$	
Size	sim.	ana.	sim.	ana.	sim.	ana.
1 KB	0.31	0.30	0.62	0.62	0.96	1.09
4 KB	0.40	0.41	0.96	0.98	1.88	1.91
8 KB	0.65	0.62	1.12	1.08	2.00	1.04
16 KB	0.82	0.80	1.58	1.56	2.81	2.88

File	$p = 0.001$		$p = 0.005$		$p = 0.010$	
Size	sim.	ana.	sim.	ana.	sim.	ana.
1 KB	0.37	0.36	0.59	0.61	1.05	1.20
4 KB	0.59	0.59	1.08	1.05	1.81	1.79
8 KB	0.99	0.91	1.50	1.53	2.53	2.56
16 KB	1.43	1.40	2.22	2.36	3.58	3.63

TABLE III

TCP SACK TRANSFER TIMES. $RTT = 100ms$ (TOP) AND $RTT = 200ms$ (BOTTOM), $MSS = 1.4KB$, $W_{max} = 24$.

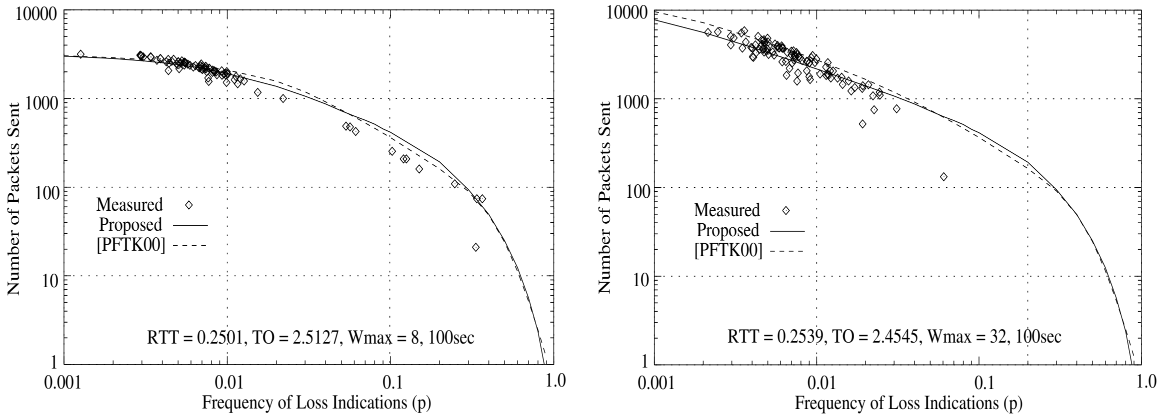


Fig. 2. Steady state throughput of TCP Reno connections.

IV. STEADY-STATE THROUGHPUT

In this section, we model the steady state throughput of infinite Tahoe, Reno and SACK flows by extending the models of the previous section. We first note that for an infinite flow with $p > 0$, the probability that the flow experiences a single on no loss indications is 0 and the average number of packets between two successive losses, $d = 1/p$. The time between two successive losses can be obtained using the equation for $t_{M\_loss}(d)$ for each of the three versions. Dividing d by the expected values of this transfer time then gives us the steady state throughput of the flow. The possible values of the $cwnd$ in this case is again assumed the $cwnd$ to vary uniformly between 1 and c_{wm}, where c_{wm} is given by

$$c_{wm} = min\left\{ \left\lceil (-1 + \sqrt{1+16d})/2 \right\rceil, W_{max} \right\} \tag{40}$$

The expected time to transfer d packets can now be obtained by averaging $t_{M\_loss}(d)$ over all possible values of the $cwnd$ and all possible positions of the loss indication within the round and is given by

$$t_{ss}(p) = \frac{2}{c_{wm}(c_{wm}+1)} \sum_{h=1}^{c_{wm}} \sum_{j=1}^{h} t_{M\_loss}(d) \tag{41}$$

where $t_{M\_loss}(d)$ is given by Eqn. (16) for Tahoe, Eqns. (26) and (27) for Reno, and Eqns. (36) and (37) for SACK. The steady state throughput in bytes per second of a TCP connection is thus

$$R = \frac{dMSS}{t_{ss}} = \frac{MSS}{t_{ss}p} \tag{42}$$

We note that though the expression for the steady state throughput does not have the same closed form as those in [15], their numerical values are almost the same. In Fig. 2 we show the results of our model extended to calculate the steady state throughput. The results are for TCP Reno and we compare our results with the model and traces reported in [15]. We see that our results are almost exact over the whole range of loss probabilities. We could not present any results for the case of TCP Tahoe and SACK as we currently do not have access to machines running Tahoe and SACK at other locations though we hope to add them in the final version of the paper.

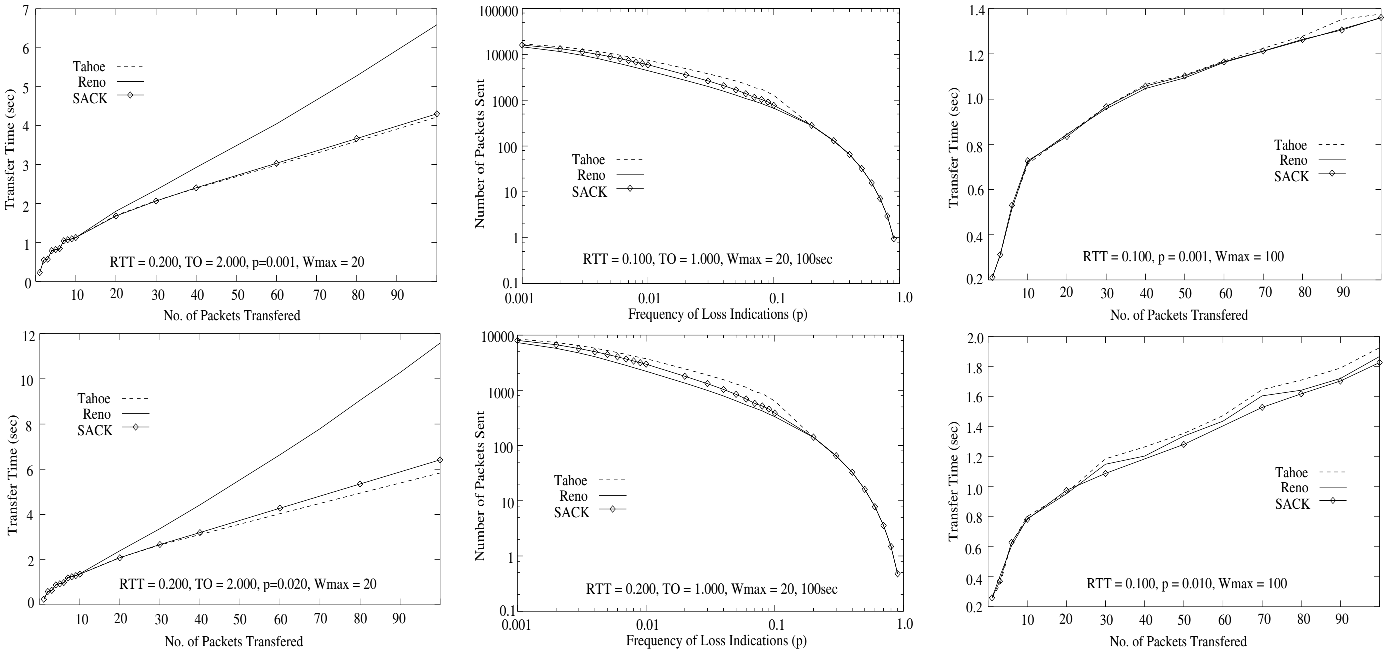
V. COMPARISON OF TAHOE, RENO AND SACK

In this section we compare the performance of the three versions of TCP in terms of both their latency and steady-state throughput. In Fig. 3(a) and 3(b) we plot the expected transfer times and the steady state throughputs of the three versions for different loss probabilities, transfer sizes and RTTs. We note that in all cases, Reno performs worse than the others which is due to the fact that it resorts to timeouts in the presence of multiple losses. The result of importance here is that Tahoe outperforms SACK in the presence of correlated losses. This is explained by considering the fact that with correlated losses, the probability that a packets loss leads to the loss of more than half of the packets of that round is close to 0.5. Thus SACK flows timeout quite frequently, degrading their performance. However, Tahoe has a very conservative retransmission policy and assumes that all outstanding packets following a lost packet are also lost. The immediate retransmission of all the losses in the slow start mode leads to considerable savings and more than makes up for any unnecessary retransmissions.

The difference in the performance of these versions of TCP under the influence of a different loss model is highlighted in Fig. 3(c) which shows the simulation results of the transfer times using *ns*. For these simulations, we used an independent loss model. This scenario is more likely for queueing disciplines like RED. With independent losses, SACK performs better than both Tahoe and Reno while Reno performs better than Tahoe. The examples considered in [5] which show similar results also have independent losses. SACK performs better with independent losses as now the probability of the loss of more than half the packets from a window is reduced enormously.

VI. CONCLUSION AND DISCUSSIONS

While TCP Reno has been modeled extensively, no models exist for estimating the latency of TCP Tahoe and SACK. In this paper we presented analytic models for estimating the latencies and the steady-state throughput of TCP Tahoe, Reno and SACK. Our models are based on the assumption of correlated or bursty losses currently prevalent in the Internet due to the dominance of droptail queues in the routers. The models were validated using both simulations as well as traces of TCP transfers over the Internet. Also, the estimates of the transfer times predicted by our TCP Reno model more accurate than existing models,



(a) Correlated losses (model) (b) Correlated Losses (model) (c) Independent Losses (sim.)

Fig. 3. Transfer times and steady state throughput for Tahoe, Reno and SACK for correlated and independent losses.

specially for short transfers.

TCP SACK is specially geared towards providing smaller retransmission times in the presence of multiple losses from the same window. However, we show that with correlated losses losses SACK is unable to provide adequate protection against the occurance of timeouts. If a SACK flow with a current window of $cwnd$ loses $2 + cwnd/2$ packets then current implementations of SACK will timeout. Hence correlated losses, TCP Tahoe performs better than SACK because of its conservative retransmission policy. Though Tahoe may unnecessarily retransmit some correctly received packets, this is more than made up the avoidance of timeouts through the direct retransmission of all the packets which are outstanding at the instant the loss is detected.

It is also of interest to note that the performance of the three versions of TCP changes when an independent loss scenario (unrealistic in current Internet scenarios) is used. With independent losses, the probability of multiple losses from the same window reduces and SACK is able to recover the losses without going into a timeout in most cases. Thus in these cases SACK perform better than both Tahoe and Reno while Reno performs better that Tahoe. Since independent loss models do not reflect reality, it is apparent that such assumptions will lead to incorrect conclusions. The point to note here though is that buffer management policies can make significant differences in protocol performance.

Though there are various possibilities to provide enhanced robustness against timeout to SACK, the most apparent one is to change the implementation of `pipe` to allow retransmission of lost packets on the receipt of partial ACKs even though `pipe` may be less than $cwnd$. Since SACK times out in the presence of heavy losses compared to the window size (where the losses need not be back to back) this change would eliminate timeouts in these circumstances also. Also, with droptail queues here to stay in the foreseeable future, providing protection against bursty losses is necessary to ensure the benefits promised by SACK.

References

[1] R. Braden and V. Jacobson, "TCP extensions for long delay paths," *RFC 1072*, October 1988.

[2] R. Braden, V. Jacobson and L. Zhang, "TCP extensions for high-speed paths," *RFC 1185*, October 1990.

[3] N. Cardwell, S. Savage and T. Anderson, "Modeling TCP latency," *Proceedings of IEEE INFOCOM,* Tel Aviv, Israel, March 2000.

[4] C. Casetti and M. Meo, "A New Approach to Model the Stationary Behavior of TCP Connections," *Proc. IEEE INFOCOM,* Tel Aviv, Israel, March 2000.

[5] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," *Computer Comm. Review,* vol. 26 no. 3, pp. 5-21, July 1996.

[6] J. Heidemann, K. Obraczka and J. Touch, "Modeling the performance of HTTP over several transfer protocols," *IEEE/ACM Trans. on Networking,* vol. 5, no. 5, Oct 1997.

[7] V. Jacobson, "Congestion avoidance and control," *Proceedings of ACM SIGCOMM,* pp. 314-329, 1988.

[8] V. Jacobson, "Modified TCP congestion avoidance algorithm," Email to the end2end-interest mailing list, `ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt`.

[9] A. Kumar, "Comparative performance analysis of versions of TCP in a local network with a lossy link," *IEEE/ACM Trans. on Networking,* vol. 6, no. 4, Aug 1997.

[10] T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss," *IEEE/ACM Trans. on Networking,* vol. 5, no. 3, pp 336-350, Jun 1997.

[11] M. Mathis, J. Semke, J. Mahdavi and T. Ott, "The macroscopic behavior of the TCP congestion Avoidance Algorithm," *Computer Communications Review,* vol. 27, no. 3, pp 67-82, July 1997.

[12] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, "TCP Selective Acknowledgment Options," *RFC 2018,* April 1996.

[13] R. Morris, "Scalable TCP Congestion Control," *Proceedings of IEEE INFOCOM,* Tel-Aviv, Israel, March 2000.

[14] T. Ott, J.H.B. Kemperman and M. Mathis, "The stationary behavior of

ideal TCP congestion avoidance," Unpublished Manuscript, Aug 1996. ftp://ftp.bellcore.com/pub/tjo/TCPwindow.ps
[15] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, "Modeling TCP Reno performance: A simple model and its empirical validation," *IEEE/ACM Trans. on Networking,* vol. 8, no. 2, pp. 133-145, April 2000.
[16] B. Sikdar, S. Kalyanaraman and K. S. Vastola, "TCP Reno with Random losses: Latency, Throughput and Sensitivity Analysis," Proceedings of IEEE IPCCC, Phoenix, AZ, April 2001.
[17] W. R. Stevens, "TCP/IP illustrated volume 1," Addison Wesley, 1994.