TCP Reno with Independent losses: Latency, Throughput and Sensitivity Analysis

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

Abstract

While almost all of the existing models for TCP performance focus on the steady-state throughput of infinite flows, most TCP connections in today's Internet are very short and spend most of their time in the slow start phase, making steady-state models inappropriate. This paper focuses on two of the most important shortcomings of current models. Firstly, we present models for the latency of TCP Reno with independent losses for which currently no models exist. Secondly, we investigate the effects of network and protocol factors on TCP performance and quantify and isolate their contribution to TCP latency using an empirical model. Our analytic model gives a more accurate estimation of TCP latencies in Internet measurements than those predicted by [3], which extends the steady state analysis of Padhye et al. [14] to model finite flows with correlated losses. The main features of our work are the modeling of timeouts and slow start phases which occur anywhere during the transfer and a more accurate model for the evolution of the *cwnd* in the slow start phase. Additionally, the proposed model can also estimate the steady-state throughput of long flows. We also introduce empirical models for TCP Reno which allows a better "feel" of TCP latency and the nature of its dependence on loss probabilities and window limitation along with a sensitivity analysis of the effects of delayed acknowledgments (ACKs), window limitation and packet size on TCP latency.

1 Introduction

Widespread interest in TCP performance has led to the development of numerous models, each with different assumptions and approximations. Most of the existing models [4, 6, 7, 8, 10, 12, 13, 14] confine their attention to the steady state throughput of long TCP connections. In contrast to the infinite flow assumptions of the steady state models, recent studies have shown that most TCP connections today carry HTTP traffic [16] with average connection sizes of 8-12 KBytes and the vast majority of the connections transferring files less than 10 KBytes [5, 9, 16]. Short flows spend most of their time in the slow start phase and are over before their *cwnd* becomes relatively large and most losses recovered using timeouts rather than fast retransmits and congestion avoidance. The underlying assumptions of steady state models thus cease to hold for short transfers and consequently they are unsuitable for modeling shorter flows. In [3], the authors propose a model which extends the results from [14] to finite flows by accounting for the connection establishment phase and an approximate analysis of the initial slow start.

While TCP modeling has been widely investigated in literature, this paper's contribution is more than just marginal. Almost all the literature on TCP modeling concentrates on steady-state throughput and the derived expressions are complex and do not shed much light on the contribution of various network parameters on TCP's performance. In this paper, one of our main concerns is to explore these effects in greater detail and we quantify the effects of factors like maximum allowable window sizes, maximum segment sizes, effect of delayed versus undelayed ACKs etc. on TCP performance. Also, we focus on the performance of TCP Reno in the presence of independent losses, both in terms of its latency as well as the steady-state throughput. The throughput of TCP Reno under independent losses has been considered in [7, 8, 10]. However, no models exist for estimating the latency of TCP Reno under independent losses. The independent loss model becomes important while considering scenarios in the Internet [2], losses and time variations due to wireless links [8] and UBR and ABR service classes in ATM networks [1].

Our model for the latency of TCP Reno is better suited to TCP latencies measured over the Internet as compared to the only other model proposed by Cardwell et al. in [3] (which assumes correlated losses) as shown by our measurements. Unlike [3], our model captures the slow start effects subsequently in the transfer which arise due to timeouts. We also use a more accurate expression for modeling the *cwnd* increase pattern in TCP flows with delayed ACKs and both these factors lead to more a more accurate model. Our sensitivity study and the empirical models pinpoint the effects of various factors contributing to the latency and can be used as design guidelines and determining quality of service guarantees.

The rest of the paper is organized as follows. In Section 2 we give a description of the assumptions about the network and the TCP transfer. We first present models for the latency and steady-state throughput of TCP Reno with delayed ACKs in Sec. 3 and 4 respectively. We then model TCP Reno without delayed ACKs in Sec. 5. Section 6 presents the validation results and Section 7 introduces the empirical models which isolates the contributions of various network factors on TCP latency. Section 8 deals with the sensitivity analysis of TCP transfer times with respect to segment sizes and maximum window sizes. Finally, Section 9 presents a discussion of the results and concluding remarks.

2 Assumptions

We follow a network and TCP transfer scenario similar to those in [14] and [3]. We assume that the hosts use a congestion control algorithm from the TCP Reno family. Also, our model considers the latency arising only from TCP's performance and thus we do not account for delays arising at the end points from factors such as buffer limitations. We assume that the sender sends full-sized segments as fast as its congestion window allows and the receiver advertises a consistent flow control window. We assume that the receiver uses the delayed ACK scheme specified in RFC 2581. As in [3], we do not account for the effects of Nagle's algorithm and silly window avoidance.

As in [14] and [3] we model TCP latency by considering "rounds". A round begins with the transmission of a window of packets from the sender and ends when the sender receives an ACK for one or more of these packets. We assume that the losses in a round are independent of losses in other rounds and unlike [14] and [3] we assume that losses in a single round are also independent of each other. This loss model is applicable in Internet scenarios [2], wireless links [8] and ATM networks [1]. Also, for short transfers, the *cwnd* values are expected to be small and the flows usually do not use a significant portion of the path's bandwidth. The total number of losses suffered by a flow with independent losses is given by a binomial distribution.

Finally, we assume that the time to transmit all the packets in a round is smaller than the duration of the round and that the duration of a round is independent of the window size. The amount of data transferred is assumed to be arbitrary allowing on one hand for extremely short transfers which suffer extremely few or no losses and on the other hand, for large transfers whose latencies might be modeled using their steady state throughput.

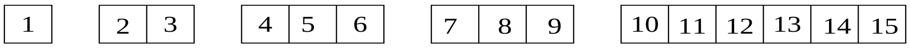
3 Modeling the Latency of TCP Connections

In this section we propose our model to estimate the transfer time of TCP connections as a function of the RTT and the loss probability on the path. Our approach is based on estimating the transfer time, given that a flow suffers a specified number of losses. This, combined with the probability that a flow suffers a given number of losses, then gives us the expected transfer time of a TCP flow. We first derive the transfer time of a TCP connection when it does not suffer any losses and then derive the expressions for TCP flows with a single loss. The model is then extended for multiple losses. We derive the transfer times as a function of N, the total number of unique packets to be transferred, where $N = \lceil \frac{data\_size}{mss} \rceil$.

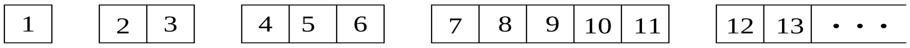
3.1 Connection Establishment

A TCP connection begins with a three-way handshake, beginning with the initiating host sending a SYN segment. The receiver responds with an ACK for the initiating host's SYN and also sends a SYN packet of its own. When the initiating host receives this SYN/ACK packet it assumes that the connection has been established and confirms this by sending an ACK of its own. 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. Then, following the arguments and the derivation of [3], it can be shown that, for loss rates p low enough, most connections are successfully established before TCP gives up and the expected duration of the connection setup phase, t_{setup}, is given by

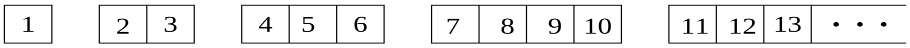
$$t_{setup} = RTT + 2T_s \left(\frac{1-p}{1-2p} - 1 \right) \tag{1}$$



(a) Without delayed ACK timer expiration



(b) With delayed ACK timer expiration



(c) Proposed model

Figure 1: Window increase patterns for TCP flows in the slow start phase.

3.2 Window Increase Pattern

TCP starts its transmission in the slow start phase and increases its window by one for every ACK it receives. However, the receiver (with delayed ACKs) sends one ACK for every two packets that it gets or if the delayed ACK timer expires. Examples of the increase pattern of the *cwnd* of a TCP flow starting with $cwnd = 1$ is shown in Figure 1 with the receiver sending one ACK for every two packets it receives. Now consider the round with $cwnd = 3$ (round in which packet numbers 4, 5 and 6 are sent). The receiver sends an ACK for the first two packets and delays sending the ACK for packet number 6. If a new packet arrives before the ACK timer expires, packet number 6 is acknowledged along with the new packet and the subsequent *cwnd* increase pattern is shown in Figure 1(a). However, if the ACK timer expires before a new packet arrives, an ACK is sent with results in a *cwnd* of 5 as shown in Figure 1(b).

To account for such complex behavior of the *cwnd*, we develop a more accurate model which tries to model the increase pattern based on the expected value of the *cwnd* for any round. This results in a more accurate representation of the number of packets transmitted in any round as compared to 1.5^n for the n^{th} round as used in [3]. The number of packets transmitted in the n^{th} round according to our model is given by

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

The predicted window increase pattern for this model is shown in Figure 1(c). Note that our model predicts the number of packets transmitted in the fourth round as 4 which is the average of the packets transmitted in examples (a) and (b). The number of packets transmitted in the first k rounds of the slow start phase is then given by

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

Note that after the first packet of a flow is sent the receiver waits in vain for the second packet to arrive. Eventually the delayed ACK timer expires and an ACK is sent which increases the *cwnd* to 2. We denote this delay due to the delayed ACK as t_{dack} and is 100ms for UNIX systems and 150ms for Windows.

3.3 Flows Without Losses

When a flow does not experience any losses, the congestion window increases exponentially until it reaches W_{max} and then stays there until the transfer is complete. The number of rounds required to reach a *cwnd* of W_{max} can be computed from Equation (2) and is given by

$$n_{wmax} = \left\lfloor 2\log_2\left(\frac{2W_{max}}{1+\sqrt{2}}\right)\right\rfloor \tag{4}$$

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

$$N_{exp} = \left\lfloor 2^{\frac{n_{wmax}+1}{2}} + 3(2)^{\frac{4n_{wmax}-3}{8}} - 2 - \frac{3\sqrt{2}}{2}\right\rfloor + W_{max} \tag{5}$$

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

$$t_{no\_loss}(N) = \begin{cases} \left\lceil 2\log_2\left(\frac{2N+4+3\sqrt{2}}{2\sqrt{2}+3(2)^{\frac{5}{8}}}\right)\right\rceil RTT, & \text{if } N \leq N_{exp} \\ \left\lceil n_{wmax} + \left\lceil\frac{N-N_{exp}}{W_{max}}\right\rceil\right\rceil RTT, & \text{otherwise} \end{cases} \tag{6}$$

3.4 Flows with a Single Loss

We now consider the case when the flow experiences a single loss. Consider a flow of N packets where the i^{th} packet is lost. The window increases exponentially till the first i packets are transmitted. The time to transmit the first i packets is then given by $t_{no\_loss}(i)$ from Equation (6). Now, if *cwnd* is less than 4 in the round when the packet is lost, the loss will lead to a timeout. The *cwnd* now reduces to 1 and the exponential slow start phase after the timeout lasts till $cwnd = 2$ and then gets into the congestion avoidance mode. The average duration of the timeout period is given by $E[TO]RTT$ as calculated in [14] and is given by

$$E[TO] = T_o\frac{2(1+p+2p^2+4p^3+8p^4+18p^5+32p^6)}{1-p} \tag{7}$$

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, 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. The solution for this equation (and accounting for the effect of window limitation) is given by

$$t_{linear}(a,b) = \begin{cases} \left\lceil\frac{a-x(x+1)+b(b-1)}{x+1}\right\rceil + 2x - 2(b-1) & \text{if } a \leq W_{max}(W_{max}+1) - b(b-1) \\ \left\lceil\frac{a-W_{max}(W_{max}+1)+b(b-1)}{W_{max}}\right\rceil + 2W_{max} - 2(b-1) & \text{otherwise} \end{cases} \tag{8}$$

where $x = \lfloor\frac{-1+\sqrt{1+4(a+b(b-1))}}{2}\rfloor$. The first part of Equation (8) is for cases when *cwnd* does not reach W_{max} and the second part is for the other cases. To find the time to transmit the N packets, we first need to find the number of packets that need to be transmitted in the congestion avoidance mode following the timeout. Before we address this, we first introduce the expressions for the *cwnd* of the round when the i^{th} packet was transmitted and the sequence number of the last packet of that round. To find the *cwnd* of the round when the i^{th} packet was transmitted we first find the number of rounds, denoted by $r(i)$, it takes to transmit i packets without any loss assuming there is no effect of window limitation. From Eqn. (6), $r(i)$ can be expressed as

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

If $r(i)$ is greater then or equal to n_{wmax}, the number of rounds it takes for *cwnd* to reach W_{max}, we know that $cwnd = W_{max}$. For all other cases, $cwnd < W_{max}$ and is given by Equation (2). Then, $cwnd(i)$, the *cwnd* of the round when the i^{th} packet is transmitted can be expressed as

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

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

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

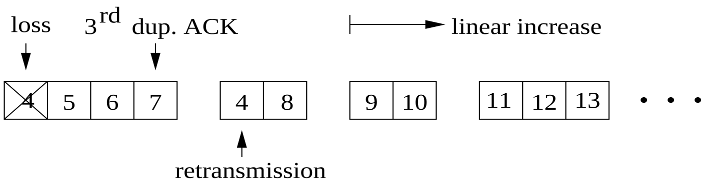
We now proceed to find the number of packets that remain to be transmitted in the congestion avoidance phase. In the round the i^{th} packet is transmitted, $i - n_{max}(i) + cwnd(i) - 1$ packets are transmitted before the i^{th} packet. Thus we can expect $\lceil \frac{i - n_{max}(i) + cwnd(i) - 1}{2} \rceil$ ACKs for these packets. Thus the *cwnd* when the i^{th} packet is lost increases from $cwnd(i)$ to $cwnd(i) + \lceil \frac{i - n_{max}(i) + cwnd(i) - 1}{2} \rceil$. The total number packets which are sent till the flow experiences the timeout is then $k = cwnd(i) + \lceil \frac{i - n_{max}(i) + cwnd(i) - 1}{2} \rceil + i - 2$. One more packet gets sent in the exponential phase with $cwnd = 1$ following the timeout. Thus the number of packets which are sent in the congestion avoidance mode is $N - k - 1$ and the time required to transmit these is then given by $t_{linear}(N - k - 1, 2)$ from Equation (8). The total time required to transfer the N packets is then given by

For $i = 1, \cdots, 6$

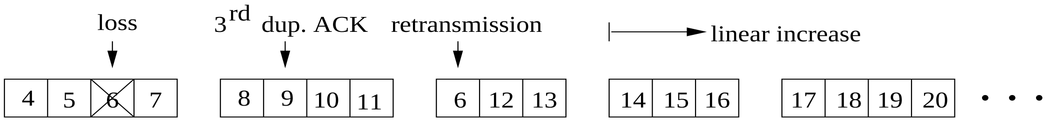
$$t_{1\_loss}(N) = [t_{no\_loss}(i) + E[TO] + t_{linear}(N - k - 1, 2) + 1]\, RTT \tag{12}$$

Observe that the first term in Equation (12) corresponds to the time to transmit the first i packets, $E[TO]$ gives the duration of the timeout period and the last two terms corresponds to the time spent in the congestion avoidance phase to transmit the $N - k - 1$ remaining packets.

For rounds in which *cwnd* is greater than or equal to 4, a single error in any packet will lead to congestion avoidance, not timeout. Once three duplicate ACKs are received the sender infers a loss and retransmits. To find the total transfer time, we first find the number of packets k that are transmitted *before* the window cuts down to $cwnd/2$. Note that now we can have two cases depending on whether or not the lost packet was amongst the last three packets of the round. If the lost packet is among the last three packets in the round a new round of packets is transmitted before TCP gets



(a) Lost packet is NOT one of the last three packets of the window: 3rd dup. ACK in same round



(b) Lost packet is one of the last three packets of the window: 3rd dup. ACK in next round

Figure 2: Window increase patterns for TCP flows with different positions of the lost packet in a window.

three duplicate ACKs. Otherwise, a third duplicate ACK received at the end of the same round and the lost packet is retransmitted in the following round. Figure 2 illustrates this concept.

First consider the case when the lost packet is not one of the last three packets of the round. In the round the i^{th} packet is transmitted, $i - n_{max}(i) + cwnd(i) - 1$ packets are transmitted before the i^{th} packet and we can thus we can expect $\lceil \frac{i-n_{max}(i)+cwnd(i)-1}{2} \rceil$ ACKs for these packets. Thus the $cwnd$ when the i^{th} packet is lost is $cwnd(i) + \lceil \frac{i-n_{max}(i)+cwnd(i)-1}{2} \rceil$. On receiving the third duplicate ACK, $cwnd$ is halved, three added to $cwnd$, the lost is packet retransmitted and the window increased by one for every duplicate ACK that is received. Once the lost packet is acknowledged, the flow goes into the congestion avoidance mode. On the other hand, for the case when the lost packet is one of the last three packets of the round, an additional window of $cwnd(i) + \lceil \frac{i-n_{max}(i)+cwnd(i)-1}{2} \rceil$ packets is sent before the third duplicate ACK is received. Again, on the receipt of the third duplicate ACK, the window reduces to half, the lost packet is retransmitted and the window is increased by three. For every additional duplicate ACK the window is increased by one and when the lost packet is acknowledged, the flow goes into the congestion avoidance mode. The number of packets which remain to be transmitted in the congestion avoidance mode is then given by $N - k$ where

$$k = \begin{cases} \min\left\{W_{max}, \left\lceil \frac{cwnd_{i+1}}{2} \right\rceil + cwnd_{i+1} - 1\right\} + i - 1 & \text{if } n_{max}(i) - i < 3 \\ \min\left\{W_{max}, \left\lceil \frac{cwnd_{i+1}}{2} \right\rceil + n_{max}(i) - i\right\} + i - 1 & \text{otherwise} \end{cases} \tag{13}$$

where $cwnd_{i+1} = \min\{W_{max}, \lceil \frac{i-n_{max}(i)+cwnd(i)-1}{2} \rceil + cwnd(i)\}$. At the beginning of the congestion avoidance mode, $cwnd$ becomes half of its value when the third duplicate ACK was received and is given by $\lceil \frac{cwnd_{i+1}}{2} \rceil$. The time to transmit the remaining $N - k$ packets in the congestion avoidance mode can then be calculated using $t_{linear}(N - k, \lceil \frac{cwnd_{i+1}}{2} \rceil)$ from Equation (8). The time required to transfer N packets is then given by

For $i = 7, \cdots, N$

$$t_{1\_loss} = [t_{no\_loss}(i) + 1]\, RTT + \begin{cases} [t_{linear}(N-k, n) + 1]\, RTT, & \text{if } n_{max}(i) - i < 3 \\ [t_{linear}(N-k, n)]\, RTT, & \text{otherwise} \end{cases} \tag{14}$$

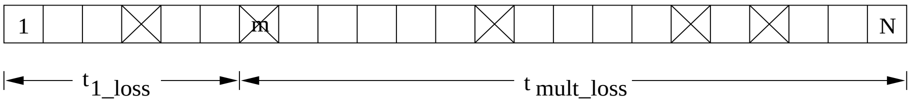


Figure 3: Modeling a flow by breaking it up into the section which has the first loss and the remaining section with the rest of the losses.

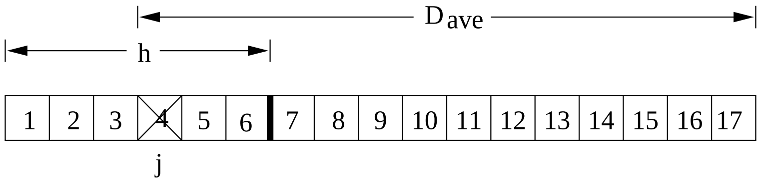


Figure 4: The parameters h and j and their relation to D_{ave}.

where $n = \lceil \frac{cwnd_{i+1}}{2} \rceil$.

3.5 Flows with Multiple Losses

We now consider the case when there is more than one loss. For a flow of N packets which experiences M losses, we assume that the second loss occurs at the m^{th} packet and find the time taken to transmit the first $m-1$ packets (where there is one loss) using Equations (12) and (14) of Section 3.4. Please refer to Figure 3 for an illustration. After the second loss, $N-m+1$ packets remain to be transmitted with $M-2$ losses within these packets. With independent losses, the average distance (in number of packets) between two consecutive losses, D_{ave}, is given by

$$D_{ave} = \frac{N-m+1}{M-1} \tag{15}$$

We now take D_{ave} and look at its substructure. We compute the average time to transmit D_{ave} packets which multiplied by $M-2$ gives the time required to transfer the remainder of the flow after the first $m-1$ packets. After the first loss, we approximate the possible range of values of *cwnd* when the subsequent losses occur by $\{1, \cdots, \lceil c\frac{-1+\sqrt{1+16D_{ave}}}{2} \rceil\}$. To account for the effect of window limitation on the possible values of *cwnd*, we limit its range to $\min\{W_{max}, \lceil c\frac{-1+\sqrt{1+16D_{ave}}}{2} \rceil\}$. The factor $c = \frac{(3+10RTT)(1-p)^2}{4(1+p+p^2)^3}$ was empirically derived using simulations and curve fitting. 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.

When a TCP transfer is in the congestion avoidance mode, as it will be after the first loss, a subsequent loss leads to a timeout only if the *cwnd* value during that round is less than 4. (For tractability, we neglect the cases when the subsequent losses occur while the connection is in the slow start mode after a timeout). Now, consider such a flow with $cwnd = h$ when the j^{th} packet in the round is lost (see Figure 4). We want to find the time to transmit D_{ave} packets following the lost packet. Let us first consider the case when $cwnd < 4$. Note that if $j \neq 1$, i.e., if the lost packet is not the first packet of the round, then some packets from the round get valid ACKs and the window slides forward. This allows the transmission of additional packets in the following round before the flow experiences the timeout. The number of packets successfully transmitted following the j^{th} packet before the timeout can be shown to be $h-1$. The slow start phase following the timeout lasts effectively for one round with $cwnd = 1$ since the exponential increase lasts only till $cwnd = 2$. Thus $D_{ave} - h$ packets remain to be transmitted in the congestion avoidance mode and the time for this can be found using $t_{linear}(D_{ave} - h, 2)$ using Equation (8). The total time to transmit the D_{ave} packets is then given by

$$t_{timeout}(l) = [E[TO] + I(j > 1) + t_{linear}(D_{ave} - h, 2)]RTT \tag{16}$$

where $I(x)$ is the indicator function and takes a value of 1 when condition x is satisfied.

On the other hand, losses which occur when $cwnd \geq 4$ lead to fast retransmissions. Again, on receiving the third duplicate ACK, $cwnd$ cuts down to $\lceil \frac{h}{2} \rceil$. The lost packet is then retransmitted and 3 added to the $cwnd$. The $cwnd$ increases by one for every duplicate ACK that is received. When the retransmitted packet is acknowledged, $cwnd$ is set to $\lceil \frac{h}{2} \rceil$ and congestion avoidance takes over. The number of packets successfully transmitted following the lost packet before congestion avoidance takes over can be calculated as

$$k = \begin{cases} \min\{h + \lceil \frac{h}{2} \rceil - 1, W_{max}\} - 1 & \text{if } h - j < 3 \\ \min\{h - j + \lceil \frac{h}{2} \rceil, W_{max}\} - 1 & \text{otherwise} \end{cases} \tag{17}$$

The time to transmit the remaining $D_{ave} - k$ packets in the congestion avoidance mode can be found using $t_{linear}(N - k, \lceil \frac{h}{2} \rceil)$ since the congestion avoidance mode starts with $cwnd = \lceil \frac{h}{2} \rceil$. The time taken to transmit D_{ave} packets after such a loss, given that the j^{th} packet of a window of size h is lost, is thus given by

$$t_{fast\_retrans}(l) = \begin{cases} \left[2 + t_{linear}(D_{ave} - k, \lceil \frac{h}{2} \rceil)\right] RTT, & \text{if } h - j < 3 \\ \left[1 + t_{linear}(D_{ave} - k, \lceil \frac{h}{2} \rceil)\right] RTT, & \text{otherwise} \end{cases} \tag{18}$$

We can now use Equations (16) and (18) to calculate the expected time to transmit D_{ave} packets and multiply it by $M-2$ to get the time required to transmit the packets after the second loss. This, combined with the expected time to transmit the first $m-1$ packets using Equations (12) and (14) then gives the expected transfer time for the flow. The expected transfer time for a flow of N packets with multiple losses, $t_{mult\_loss}(N)$, is thus given by

$$t_{mult\_loss}(N) = E\{t_{1\_loss}(m-1)\} + E\{(M-2)t_{timeout}(D_{ave})\} + E\{(M-2)t_{fast\_retrans}(D_{ave})\} \tag{19}$$

where m is the sequence number of the second packet which is lost and the expectation operation is carried over all possible values of m and the number of losses M.

3.6 The Expected Transfer Time

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

$$T_{transfer}(N) = t_{setup} + (1-p)^N t_{no\_loss}(N) + p(1-p)^{N-1} E\{t_{1\_loss}(N)\} + t_{mult\_loss}(N) + t_{dack} \tag{20}$$

where t_{setup}, $t_{no\_loss}(N)$ and $t_{mult\_loss}(N)$ are defined in Equations (1), (6) and (19) respectively and $t_{1\_loss}(N)$ is defined in Equations (12) and (14). A program to calculate the expected transfer time and can be downloaded from `http://networks.ecse.rpi.edu/~bsikdar/t_time_tcp.c`

4 Steady-State Throughput

In this section we extend the analysis of the previous section to model infinite TCP flows and their steady state throughput. As in [14], we assume that the source has an unlimited amount of data to send. To extend the model of Section 3 to an infinite flow, we first note that for an infinite flow, the probability that the flow experiences a single or no loss is 0 for $p > 0$. For a packet loss probability of p, the average number of packets transmitted between two successive losses, d, is given by $1/p$. We can now use Eqns. (16) and (18) to find the average transmission time between successive errors. Dividing d by this expected transfer time, we can calculate the steady state throughput of the flow. The possible values of the *cwnd* in this case can thus be approximated to vary uniformly between 1 and cw_{max}, where cw_{max} is given by

$$cw_{max} = min\left\{\left\lceil c\frac{-1+\sqrt{1+16/p}}{2}\right\rceil, W_{max}\right\} \tag{21}$$

where $c = \frac{(3+10RTT)(1-p)^2}{4(1+p+p^2)^3}$. The expected time to transfer d packets can now be obtained using the derivations of Section 3.5. We again note that any loss in a window less than 4 leads to a timeout and in these cases, the time to transmit the d packets is obtained using $t_{timeout}(d)$ of Equation (16). On the other hand, any loss in a window greater than or equal to 4 is recovered using a fast retransmit and we can use Equation (18) to calculate the transfer time. The expected time to transfer d packets is obtained by averaging the transfer times for all cw_{max} possible values of the *cwnd* and all possible positions of the lost packet within the window. Thus, the expected transfer time can be expressed as

$$t_{ss}(p) = \frac{2}{cw_{max}(cw_{max}+1)}\left[\sum_{i=1}^{3}\sum_{j=1}^{i} t_{timeout}(d) + \sum_{i=4}^{cw_{max}}\sum_{j=1}^{i} t_{fast\_retransmit}(d)\right] \tag{22}$$

where $t_{timeout}(d)$ and $t_{fast\_retransmit}(d)$ are defined in Equations (16) and (18). The steady state throughput in bytes per second of a TCP connection is thus

$$R(p) = \frac{dMSS}{t_{ss}(p)} = \frac{MSS}{t_{ss}(p)p} \tag{23}$$

We note that though the expression for the steady state throughput does not have the same closed form as that derived by Padhye et al. in [14], their numerical values are almost the same. The numerical results and their comparison with measurements from TCP connections are given in the next section.

5 TCP Reno Without Delayed ACKs

In the absence of delayed ACKs, the receiver sends an ACK for each new packet it receives. This results in a doubling of *cwnd* every RTT in the slow start phase and an increase of 1 every RTT in the congestion avoidance phase. This faster increase of *cwnd* can lead to considerable reductions in the transfer times and increases in the steady-state throughput. We now present a model for TCP without delayed ACKs. The model follows the same structure and arguments as in the previous section and due to space restrictions, we do not elaborate them in detail again in this section.

We note that the absence of delayed ACKs does not affect the connection setup time and we again use Eqn. (1) to evaluate the setup time t_{setup}. Also, in the slow-start phase, the *cwnd* in the n^{th} round is given by 2^{n-1} and the time to transmit a packets in the congestion avoidance phase, with an initial window of b is given by

$$t_{linear}(a,b) = \begin{cases} \left\lceil \frac{-1+\sqrt{1+4(2a+b(b-1))}}{2} \right\rceil - b + 1 & \text{if } 2a \leq W_{max}(W_{max}+1) - b(b-1) \\ W_{max} + \left\lceil \frac{2a - W_{max}(W_{max}+1)+b(b-1)}{2W_{max}} \right\rceil - b + 1 & \text{otherwise} \end{cases} \tag{24}$$

5.1 Flow Without Losses

When a flow does not experience any losses, the congestion window increases exponentially till it reaches W_{max} and then stays there till the transfer is complete. The number of packets transmitted when *cwnd* reaches W_{max}, N_{exp}, is given by

$$N_{exp} = 2^{\lceil log_2 W_{max} \rceil} + W_{max} - 1 \tag{25}$$

The transfer time for N packets is then

$$t_{no\_loss}(N) = \begin{cases} \lfloor \log_2 N \rfloor + 1, & \text{if } N \leq N_{exp} \\ \lfloor \log_2 W_{max} \rfloor + \lceil \frac{N - N_{exp}}{W_{max}} \rceil + 2, & \text{otherwise} \end{cases} \tag{26}$$

5.2 Flows with a Single Loss

Consider again a flow of N packets where the i^{th} packet is lost. The highest sequence number of any packet sent in the round where the i^{th} packet was transmitted, n_{max}, can be expressed as

$$n_{max} = 2^{m+1} + \left\lceil \frac{\max\{i - 2^{m+1} + 1, 0\}}{W_{max}} \right\rceil W_{max} - 1 \tag{27}$$

where $m = \min\{\lfloor \log_2 i \rfloor, \lfloor \log_2 W_{max} \rfloor\}$. Following the arguments of the previous section, a loss will lead to a timeout only if *cwnd* is less than 4 and the number of packets successfully transmitted before the congestion avoidance phase is given by $k = \min\{i, W_{max}\} + i - 2$. The time required to transfer N packets can be expressed as

For $i = 1, 2, 3$

$$t_{1\_loss}(N) = [t_{no\_loss}(i) + E[TO] + t_{linear}(N - k - 1, 2) + 1]\, RTT \tag{28}$$

For rounds in which $cwnd \geq 4$, a single loss in any packet is recovered using a fast-retransmit. As in Section 3, to find the total transfer time we first find the number of packets k that are transmitted before the window cuts down to $cwnd/2$.

$$k = \begin{cases} i + \min\left\{W_{max}, \left\lceil \frac{\min\{i,W_{max}\}}{2} \right\rceil + \min\{i, W_{max}\} - 1\right\} - 1, & \text{if } n_{max} - i < 3 \\ i + \min\left\{\lceil \frac{i}{2} \rceil + n_{max} - i, W_{max}\right\} - 1, & \text{otherwise} \end{cases} \tag{29}$$

The time to transmit the remaining $N - k$ is then obtained by solving for n in $\sum_{j=cwnd/2}^{n} j = N - k$ and given by Eqn. (24). The time required to transfer N packets is then given by
For $i = 4, \cdots, N$

$$t_{1\_loss} = [t_{no\_loss}(i) + 1]\, RTT + \begin{cases} [t_{linear}(N-k, n) + 1]\, RTT, & \text{if } n_{max} - i < 3 \\ [t_{linear}(N-k, n)]\, RTT, & \text{otherwise} \end{cases} \tag{30}$$

where $n = \lceil \frac{\min\{i,W_{max}\}}{2} \rceil$.

5.3 Flows with Multiple Losses

For the multiple loss case, we follow the same approach as in Section 3.5. Again, we limit the range of possible values of the $cwnd$ to $\min\{\lceil c\frac{-1+\sqrt{1+16D_{ave}}}{2} \rceil, W_{max}\}$ where D_{ave} is given by Equation (15) and $c = \frac{(3+10RTT)(1-p)^2}{4(1+p+p^2)^3}$. We first consider the cases with timeouts. Following the arguments of Section 3.5, for a flow with $cwnd = h$ when the j^{th} packet in the round is lost, $h-1$ packets are transmitted after the j^{th} packet before the loss is detected. The time to transmit D_{ave} packets is then

$$t_{timeout}(l) = [E[TO] + I(j > 1)]RTT + t_{linear}(D_{ave} - h, 2)]RTT \tag{31}$$

On the other hand, losses which occur when $cwnd \geq 4$ lead to fast retransmissions. The number of packets successfully transmitted after the j^{th} packet is transmitted and before congestion avoidance begins is given by

$$k = \begin{cases} \min\{h + \lceil \frac{h}{2} \rceil - 1, W_{max}\} - 1 & \text{if } h - j < 3 \\ \min\{h - j + \lceil \frac{h}{2} \rceil, W_{max}\} - 1 & \text{otherwise} \end{cases} \tag{32}$$

The time to transmit D_{ave} subsequent packets is then given by

$$t_{fast\_retrans}(l) = \begin{cases} \left[2 + t_{linear}(D_{ave} - k, \lceil \frac{h}{2} \rceil)\right] RTT, & \text{if } h - j < 3 \\ \left[1 + t_{linear}(D_{ave} - k, \lceil \frac{h}{2} \rceil)\right] RTT, & \text{otherwise} \end{cases} \tag{33}$$

The expected time to transfer the N packets in the presence of multiple losses, $t_{mult\_loss}(N)$, is thus given by

$$t_{mult\_loss}(N) = E\{t_{1\_loss}(m-1)\} + E\{(M-2)t_{timeout}(D_{ave})\} + E\{(M-2)t_{fast\_retrans}(D_{ave})\} \tag{34}$$

where m is the sequence number of the second packet which is lost and the expectation operation is carried over all possible values of m and the number of losses M.

5.4 The Expected Transfer Time

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

$$T_{transfer}(N) = t_{setup} + (1-p)^N t_{no\_loss}(N) + p(1-p)^{N-1} E\{t_{1\_loss}(N)\} + t_{mult\_loss}(N) \tag{35}$$

where t_{setup}, $t_{no\_loss}(N)$ and $t_{mult\_loss}(N)$ are defined in Equations (1), (26) and (34) respectively and $t_{1\_loss}(N)$ is defined in Equations (28) and (30).

5.5 Steady-State Throughput

Following the methodology of Sec. 4, we first note that for infinite flows with $p > 0$, the average number of packets transmitted between two successive losses, d, is given by $1/p$. The expected time to transfer d packets can then be written as

$$t_{ss}(p) = \frac{2}{c_{wm}(c_{wm}+1)} \left[\sum_{i=1}^{3} \sum_{j=1}^{i} t_{timeout}(d) + \sum_{i=4}^{c_{wm}} \sum_{j=1}^{i} t_{fast\_retransmit}(d) \right] \tag{36}$$

where $t_{timeout}(d)$ and $t_{fast\_retransmit}(d)$ are defined in Equations (31) and (33). The steady state throughput in bytes per second of a TCP connection is thus

$$R(p) = \frac{dMSS}{t_{ss}(p)} = \frac{MSS}{t_{ss}(p)p} \tag{37}$$

6 Model Verification Results

The models proposed in the previous sections are verified in this section using measurements from real world TCP connections over the Internet and simulations using the simulator *ns*. We also verify the extension of our model to calculate the steady state throughput of TCP connections by comparing it with the results of [14].

To verify the accuracy of the proposed model in real world TCP connections, we conducted a number of measurements for TCP connections from a local machine to machines in various domains both in the United States and abroad. We show the results for TCP file transfers to the west coast of USA and Europe from our machine in the east coast of USA. The local machine was running Solaris 5.6 while the others were running FreeBSD CAIRN-2.5 and FreeBSD 3.3 respectively. The traces were collected at the sending machine using `tcpdump` which did not lose any packets and has an accuracy of 10 microseconds. In Figures 5 and 6 we show the transfer times for various files sizes to these destinations and compare them to our model. The measurement results were generated by conducting FTP transfers of randomly generated file sizes (using a uniform distribution) with the sender at the local machine and the receiver at the remote domain. The two sets of results correspond to the cases when the file sizes generated were between 100-10000 bytes and 10000-100000 bytes. The results from our model match very closely with those of the measurements.

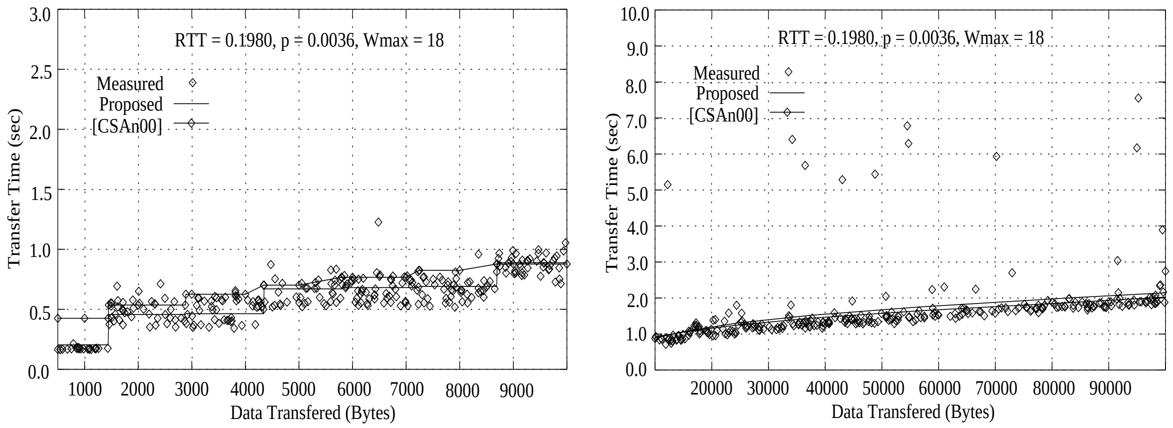


Figure 5: Transfer times of measured and modeled TCP transfers from Troy, NY to Pisa, Italy.

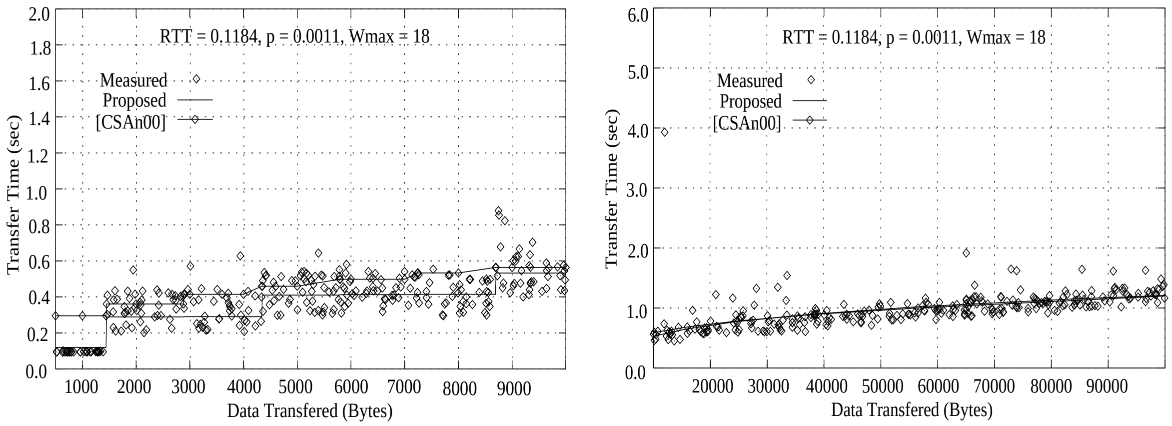


Figure 6: Transfer times of measured and modeled TCP transfers from Troy, NY to Los Angeles, CA.

Note that our results are a better match with TCP latencies measured over the Internet as compared to the model proposed in [3], particularly in transfers between 0-20000 bytes. For larger transfers, there is no appreciable difference in the results. The improvement is primarily due to fact that our model is able to capture the effects of timeouts more accurately and follows a more realistic model for the *cwnd* increase pattern.

Fig. 7 shows the results of our model extended to calculate the steady state throughput of TCP connections. We compare our results with those of Padhye et al. [14] and the traces reported therein. We plot the graphs for two of the 100 sec transfers of [14] (Figs. 16 and 17 of [14]) and we see that our results match closely with both the measurements and the model proposed in [14]. We also note that there is a little deviation in the two models as the loss probability increases which is due to the difference in the loss models. However, this does not affect the accuracy and validity of the results as our results are very close to the measured values.

The model for TCP Reno without delayed ACKs is verified using the network simulator *ns*. We resort to simulations instead of measurements over the Internet since most TCP implementations use delayed

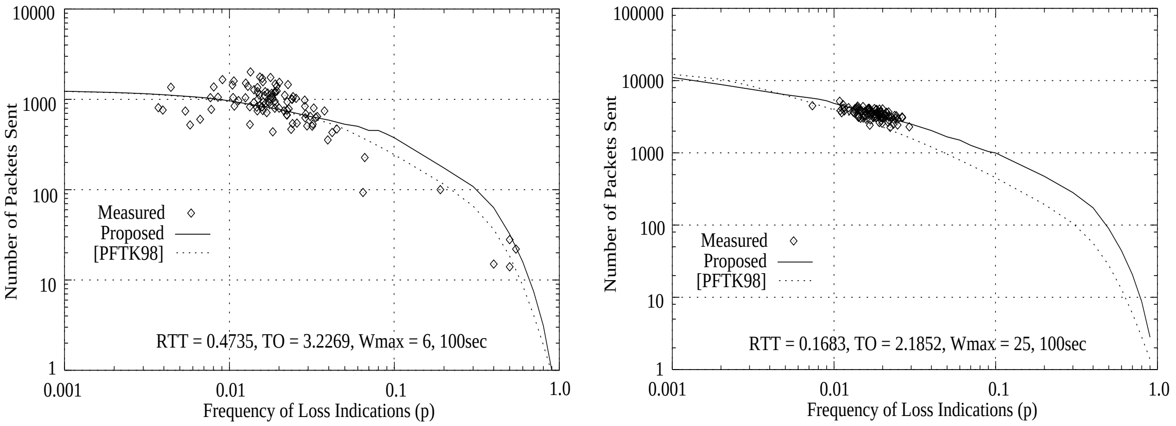


Figure 7: Steady state throughput of TCP connections from the proposed model compared with the model and measurements from [14] (Fig. 16 and 17 of [14]).

ACKs and it is difficult to access machines running TCP without delayed ACKs. For our simulations, we used a simple topology with a sender connected to a receiver through a router which randomly dropped packets with a fixed packet drop probability. The links speeds were assumed to be 1Mbps and the router had a loss probability p in the forward direction. We also assumed that there was no cross traffic and the `FullTCP` model of *ns* was used as the transport protocol. The transfer time for a given file size was determined by averaging over 1000 trials. The results are shown in Figure 8 for RTTs of 100 and 200 milliseconds loss probabilities of 0.01 and 0.05. The model fits the experimental results very well and the worst case error over all the cases we tried was less than 5%.

7 Empirical Model

To get a "feel" of TCP's behavior which might lead to design guidelines, we now propose an empirical model for TCP latency. The model gives a rough idea of the dependence of TCP latency on the loss probability and window limitation. Empirically, the time to transfer N packets using TCP can be approximated as

$$T_{emp}(N) = \left[\log_{1.57} N + \{f(p, RTT)N + 4p\log_{1.57} N + 20p\} + \frac{(10 + 3RTT)}{4(1-p)W_{max}\sqrt{W_{max}}} N\right] RTT \quad (38)$$

where $f(p, RTT) = \frac{2.32(2p+4p^2+16p^3)}{(1+RTT)^3}N + \frac{(1+p)}{RTT10^3}$. The first expression in the model corresponds to the case when the flow does not experience any losses. The second term accounts for the increase in the transfer times dues to losses while neglecting any window limitation effects. The third term accounts for the effects of window limitation. The model indicates that the transfer time increases exponentially with an increase in the loss rate. Also, for larger RTTs, the rate of increase decreases. The third term shows that the transfer time is inversely proportional to W_{max} and thus there is no significant improvement in the latency if W_{max} is increased beyond a threshold.

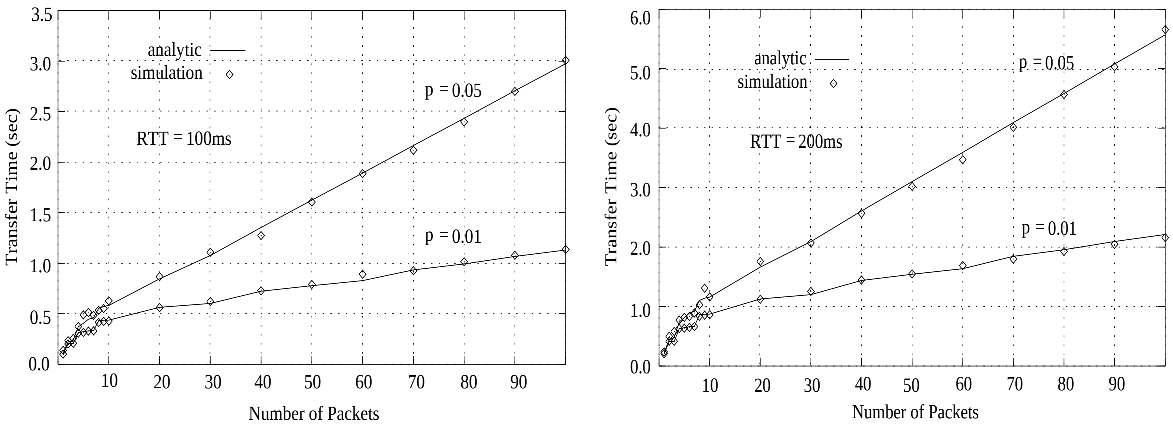


Figure 8: Transfer times from analytic and simulation results for various file sizes and loss probabilities of 0.01 and 0.05 for TCP flows without delayed ACKs. The figures show the results for RTTs of 100 and 200ms.

The transfer times predicted by the empirical model are compared with those obtained from Equation (20) in Figure 9 and as can be seen, the empirical model is a very close approximation over a wide range of RTTs, loss rates, W_{max} and file sizes. The empirical model can thus be used to get a rough indication of the dependence of TCP's latency on various parameters.

For the case of TCP Reno without delayed ACKs, empirically, the time to transfer N packets using TCP can be approximated as

$$T_{transfer}(N) = \left[\log_2 N + \frac{4.5}{1 + RTT^2} pN + \frac{1+p}{W_{max}\sqrt{W_{max}}}\right] RTT \tag{39}$$

The first expression in the model corresponds to the case when the flow does not experience any losses. The second term accounts for the increase in the transfer times dues to losses while the third term accounts for the effects of window limitation. The transfer times predicted by the empirical model are compared with those obtained from Equation (35) in Figure 10 and as can be seen, the empirical model is a very close approximation over a wide range of RTTs, loss rates, W_{max} and file sizes.

8 Sensitivity Analysis

The previous sections concentrated on the effect of the loss probability and RTT on TCP transfer times. Other factors which influence the transfer time are the packet lengths and the maximum window size W_{max}. We now investigate the sensitivity of TCP latency to variation in packet sizes and the effect of window limitation. In Figure 11 we plot the transfer time of TCP connections as a function of the packet size and W_{max} with other parameters kept constant. As expected, the graphs indicate that the transfer time increases as packet sizes get smaller and W_{max} decreases. However, we note the reduction in the transfer time with increase in W_{max} is not linear and tends to saturate. Also, for smaller data transfers, there is no effect of W_{max} on the transfer time as the transfer is over before *cwnd* reaches W_{max}. However the decrease in the transfer time with large values of W_{max} is

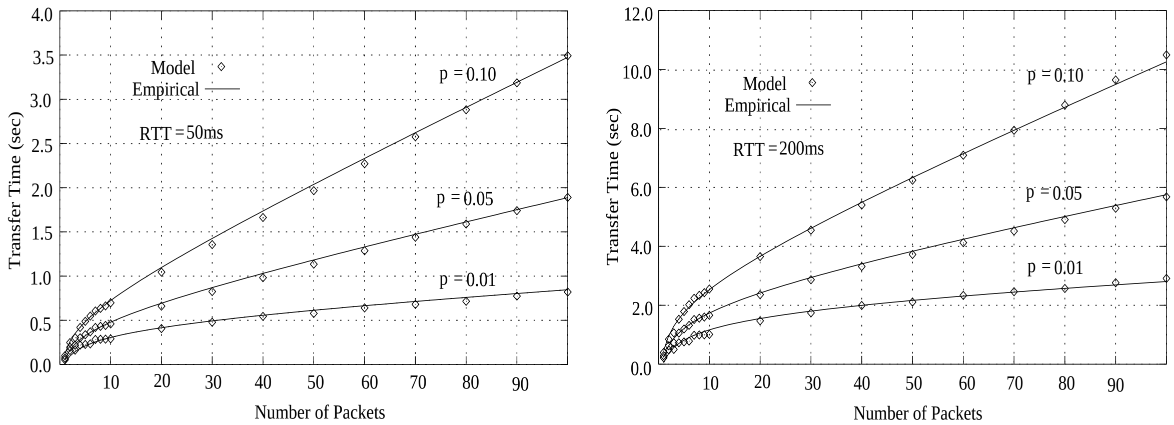


Figure 9: Transfer times from the empirical model and the analytic model of Equation (20). The first figure corresponds to a RTT of 50ms, while the second is for a RTT of 200ms.

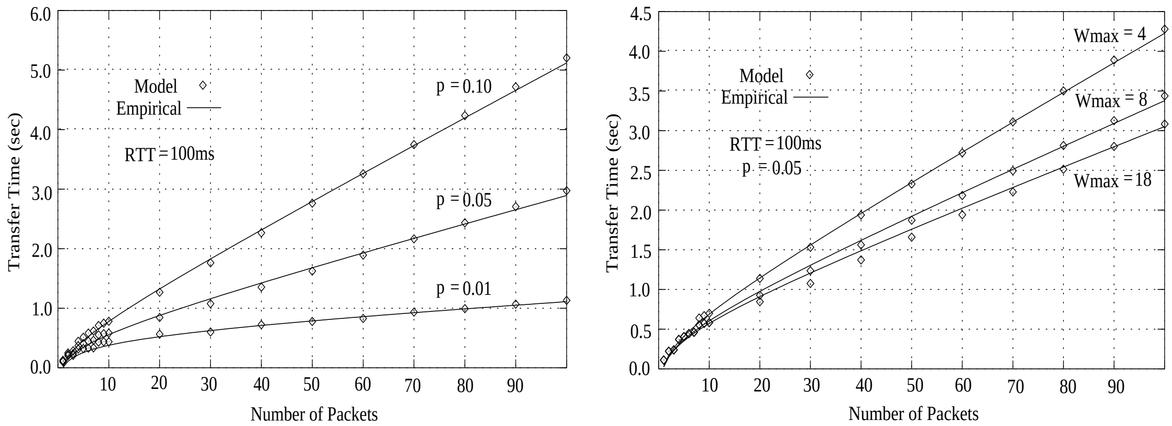


Figure 10: Transfer times from the empirical model and the analytic model of Equation (35) for a) various loss rates (without window limitation) and b) various values of W_{max}.

too small to justify increasing it beyond a limit determined by the loss probability. To explain this, we note that for a packet loss probability p, on average we see a loss every $1/p$ packets. Thus *cwnd* for the flow is generally lower than $1/p$. If $W_{max} > 1/p$, then $cwnd < W_{max}$ for most of the flow and there is no significant increase in the transfer time due to window limitation. Any further increase in W_{max} beyond this point further reduces the effect of window limitation and we see that the transfer time saturates to the value predicted by Equation (20) with $W_{max} = \infty$. We also note that the transfer time decreases as the packet size increases though here too the reduction in the transfer time does not scale linearly with the increase in the packet size.

9 Conclusion and Discussions

In this paper we presented models for TCP Reno transfer times and steady-state throughput. In addition to being the first model for Reno latency with independent losses, in contrast to earlier models

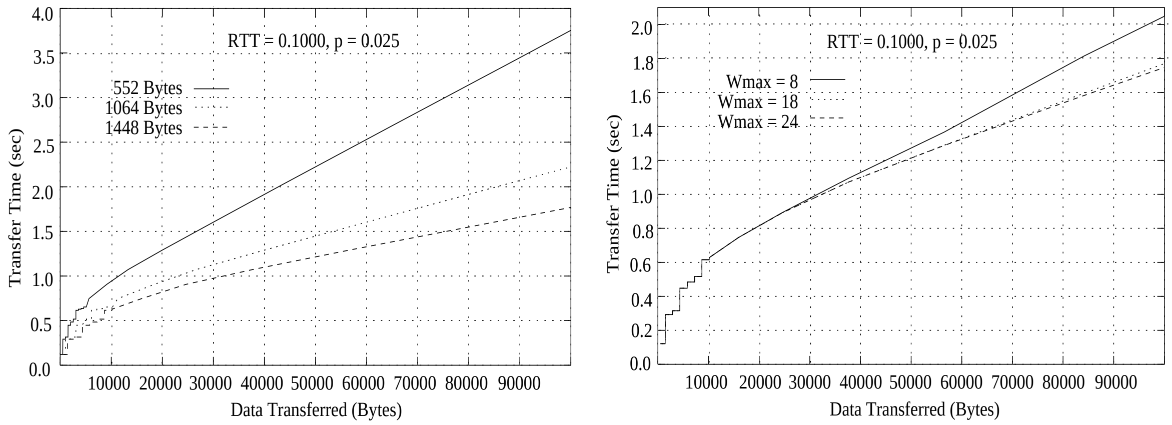


Figure 11: Effect of packet sizes and W_{max} on the transfer time of TCP connections.

which either assume no loss [6, 15], infinite flows [14] or model the slow start of TCP connections in the beginning for the transfer [3], our model accounts for arbitrary file sizes, slow start phases at all possible points in the flow and arbitrary loss probabilities. Our model is also more accurate for estimating latencies over the Internet than those predicted by [3] (which uses correlated losses). This can be attributed to the better modeling of the timeouts experienced by a TCP flow as well as a more accurate model for the evolution of *cwnd*. Our model also accounts for the possibility of window limitation.

In the second part of the paper, we investigated the effects of various network and protocol parameters on TCP latency. We presented a model for the latency and throughput in the absence of delayed ACKs. The paper also presented an empirical model for estimating the latency of TCP transfers which gives an indication of the dependence of the latency on various parameters and is thus helpful in formulating design guidelines. We also carried out sensitivity studies on TCP with respect to the packet sizes and the effect of window limitations. We show that though the transfer time decreases with increasing W_{max}, the improvement tends to saturate as W_{max} increases. This is due to the fact that for a given loss probability p, we expect to see a loss every $1/p$ packets and thus the window value of the flow will generally not exceed $1/p$. Consequently, for $W_{max} > 1/p$, the effect of window limitation is very low and further increases in W_{max} do not affect the transfer time.

It would be of interest to investigate the effects of different buffer management schemes on the loss probability and consequently on TCP transfer times. Also, another avenue to explore is the relationship of the loss probabilities on the link to TCP's slow start and congestion avoidance schemes. Applications of our model could be in designing QoS sensitive admission control algorithms, developing TCP friendly flows and online modeling and simulations.

References

[1] *ATM forum traffic management specification version 4.0, Draft specification ATM Forum/95-*

13RII, ATM Forum, March 1996.

[2] J-C. Bolot, "End-to-end packet delay and loss behavior in the Internet," *Proceedings of ACM SIGCOMM*, pp. 289-298, San Francisco, CA, September 1993.

[3] N. Cardwell, S. Savage and T. Anderson, "Modeling TCP latency," *Proceedings of IEEE INFOCOM,* pp. 1742-1751, Tel Aviv, Israel, March 2000.

[4] C. Casetti and M. Meo, "A New Approach to Model the Stationary Behavior of TCP Connections," *Proceedings of IEEE INFOCOM,* pp. 367-375, Tel Aviv, Israel, March 2000.

[5] C. R. Cunha, A. Bestavros and M. E. Crovella, "Characteristics of WWW client based traces," Technical Report BU-CS-95-010, Boston University, Jul 1995.

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

[7] A. Kumar, "Comparative performance analysis of versions of TCP in a local network with a lossy link," *IEEE/ACM Transactions on Networking,* Vol. 6, No. 4, Aug 1997.

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

[9] B. A. Mah, "An empirical model of HTTP network traffic," *Proceedings of IEEE INFOCOM'97*, Kobe, Apr 1997.

[10] M. Mathis, J. Semke, J. Mahdavi and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," *ACM Computer Communications Review,* Vol. 27, No. 3, pp 67-82, Jul 1997.

[11] P. P. Mishra, D. Sanghi and S. K. Tripathi, "TCP flow control in lossy networks: Analysis and enhancements," *Proceedings of the IFIP TC6 International Conference on Computer Networks, Architecture and Applications,* pp. 181-193, Trivandrum, India, October 1992.

[12] A. Misra and T. J. Ott, "The window distribution of idealized TCP congestion avoidance with variable packet loss," *Proceedings of IEEE INFOCOM,* pp. 1564-1572, New York, NY, March 1999.

[13] T. Ott, J.H.B. Kemperman and M. Mathis, "The stationary behavior of ideal TCP congestion avoidance," `ftp://ftp.telecordia.com/pub/tjo/TCPwindow.ps`, Unpublished Manuscript, Aug 1996.

[14] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, "A simple model and its empirical validation," *Proceedings of ACM SIGCOMM'98,* Vancouver, BC, Canada, pp 303-314, September 1998.

[15] C. Partridge and T. J. Shepard, "TCP/IP performance over satellite links," *IEEE Network,* pp 44-49, Sep/Oct 1997.

[16] K. Thompson, G. J. Miller and R. Wilder, "Wide-area Internet traffic patters and characteristics," *IEEE Network,* Vol. 6, No. 11, pp 10-23, Nov 1997.