An Integrated Model for the Latency and Steady State Throughput of TCP Connections *

Biplab 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
Phone: 518 276 8289

Abstract

Most TCP connections in today's Internet transfer data on the order of only a few KBytes. Such TCP transfers are very short and spend most of their time in the slow start phase. Thus the underlying assumptions made by steady-state models cease to hold making them unsuitable for modeling finite flows. In this paper, we we propose an accurate model for estimating the transfer times of TCP flows of arbitrary size. Our model gives a more accurate estimation of the transfer times than those predicted by [2], which extends the steady state analysis of Padhye et al. [11] to model finite flows. The main features of out 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 model the steady state throughput of TCP connections. The model is verified using web based measurements of real life TCP connections. We also introduce an empirical model which allows a better "feel" of TCP latency and the nature of its dependence on loss probabilities and window limitation. Finally, the paper investigates the the effect on window limitation and packet size on TCP latency.

*General Conference Topics: Internet, Computer Communications

†Corresponding Author

1 Introduction

The dominance of TCP as the transport protocol of choice in the Internet and a majority of today's networking applications has led to efforts to develop models to characterize its behavior. However, the complex nature of its congestion control mechanism and its dependence on feedback in the form of acknowledgments makes the development of accurate models difficult. Most of the existing models [3, 10, 5, 9, 7, 6, 11] confine their attention to the steady state throughput of long TCP connections. A model for the performance analysis of HTTP over TCP assuming no loss is given in [5]. In contrast to the infinite flow assumptions of the steady state models, most TCP transfers are short and high loss rates not uncommon. Short flows spend most of their time in the slow start phase and are over before their *cwnd* becomes relatively large. Thus any loss a flow suffers leads to 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.

Recent studies have shown that most TCP connections today carry HTTP traffic [13]. Web browsers with non-persistent connections (HTTP 1.0) lead to TCP connections with a median size of 2-3 KBytes, an average size of 8-12 KBytes with the vast majority of the connections transferring files less than 10 KBytes [4, 8, 13]. With persistent TCP connections (HTTP 1.1) the average size of the file transfer becomes larger and as indicated in [8], the average size of is is around 26-32 KBytes. In general, most of these TCP transfers are quite small from a steady-state analysis point of view. For example, with a typical MSS of 1460 bytes, a 10 KByte transfer is only 7 segments long.

This motivates the development of analytical models which take the short size of most TCP transfers into account. In this paper, we propose a single model which can model the latency of a TCP connection of any size and loss probability as well as give the steady state throughput for infinite flows. In [2], the authors propose a model which extends the results from [11] to finite flows by accounting for the connection establishment phase and an approximate analysis of the initial slow start. The authors model the portion of flow after the first loss using the steady state analysis. Unlike [2], 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 and both these factors lead to more a more accurate model as evinced by the results. The main contributions of the paper can be summarized as

- A single, accurate model which accurately predicts the performace of both short as well as steady state TCP transfers. The model is valid for any loss probability and accounts for the effects of window limitation.

- An empirical model for TCP transfer times which is particularly useful in giving a "feel" of TCP latency and developing design guidelines.
- A sensitivity analysis on the effect of packet sizes and window limitation on TCP latency.

We also explore the effects of window limitation and maximum segment size on TCP latencies. The sensitivity analysis shows that though the transfer time reduces as the maximum window size increases, the improvements tends to saturate and beyond a point (dependent on the loss probability) there is no significant improvement.

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. Section 3 introduces the model for finite TCP transfers while Section 4 extends it to model the steady state throughput of infinite flows. Section 5 presents the validation results, Section 6 introduces the empirical model, Section 7 deals with the sensitivity analysis of TCP transfer times. Finally, Section 8 presents a discussion of the results and concluding remarks.

2 Assumptions

We follow a network and TCP transfer scenario similar to those in [11] and [2]. 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 acknowledgment scheme specified in RFC 2581. As in [2], we do not account for the effects of Nagle's algorithm and silly window avoidance.

As in [11] and [2] we model TCP latency by considering "rounds". A round begins with the transmission of a window of packets from the receiver and ends when the sender receives an acknowledgment for one or more of these packets. We assume that the losses in a round are independent of losses in other rounds. Unlike [11] and [2], in this paper we assume that losses in a round are independent. This assumption is better suited to model losses occurring in networks with RED queues while the correlated error model of [11] and [2] are better suited for FIFO drop tail queueing [2]. 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. A binomial model is thus well suited for modeling the total number of losses suffered by a flow [1].

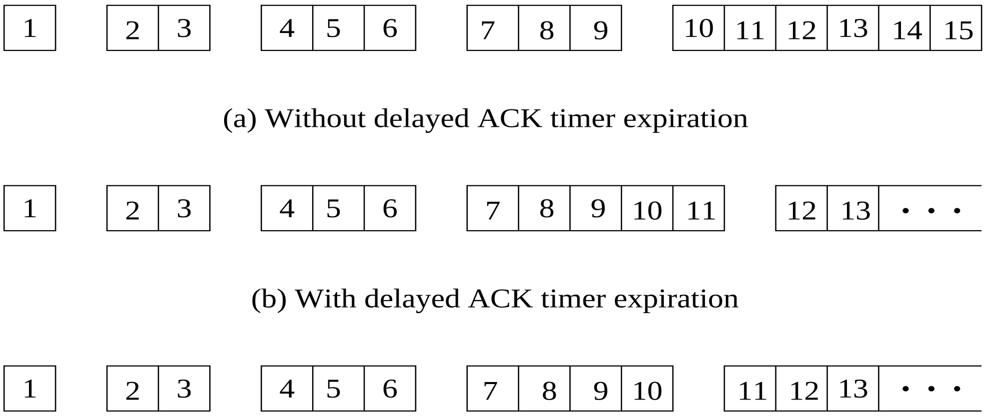
Finally, we assume that the time to transmit all the packets in a round is smaller than the duration of the round and that the duration of a round is independent of the window size. The amount of data transferred is assumed to be arbitrary allowing on one hand for extremely short transfers which suffer 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. We propose a single model which can be used for arbitrary data transfer sizes and can also give the steady state throughput of infinite TCP connections. 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. The model, in addition to accounting for the slow start at the beginning of each connection, also accounts for such events later in the flow which may arise due to timeouts. 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 [2],



(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.

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}, can be approximated as

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

3.2 Window Increase Pattern

TCP starts its transmission in the slow start phase and increases its window by one for every acknowledgment it gets. However, the receiver (with delayed acknowledgments) sends one acknowledgment for every two packets that it gets or if the delayed acknowledgment timer expires. The factor of increase is thus $\gamma = 1 + 1/2 = 1.5$. In most UNIX based systems this timer is set to 200ms which leads to an expected delay of 100ms before the acknowledgement for the first packet of the flow is sent (assuming an initial window of 1). In Windows 95 and Windows NT 4.0 systems this delay is uniformly distributed between 100ms and 200ms.

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 acknowledgment 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 acknowledgment for the first two packets and delays sending the acknowledgment 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 acknowledgment 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 γ^n for the n^{th} round as used in [2]. The number of packets transmitted in the n^{th} round according to this 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

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

Note that after the first packet of a flow is sent the receiver waits in vain for the second packet to arrive. Eventually the acknowledgment timer expires and an acknowledgment is sent which increases the *cwnd* to 2. We denote this delay due to the delayed acknowledgment 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

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

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

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

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 $cwnd = 2$. The flow now gets into the congestion avoidance mode and increases $cwnd$ by 1 every two RTTs till either the remaining packets are transmitted or $cwnd$ reaches W_{max}. The average duration of the timeout period is given by βRTT as calculated in [11] and is given by

$$\beta = \frac{2(1+p+2p^2+4p^3+8p^4+18p^5+32p^6)}{1-p} \tag{7}$$

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

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

where $x = \lfloor \frac{-1+\sqrt{1+4(a+b(b-1))}}{2} \rfloor$. Note that the first part of Equation (8) is the case when $cwnd$ does not reach W_{max} and the second part is for the other case. 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 it takes to transmit i without any loss assuming there is no effect of window limitation. If this 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

$$cwnd(i) = \begin{cases} W_{max} & \text{if } r(i) > n_{wmax} \\ \left\lfloor 2^{\frac{r(i)-1}{2}} + 2^{\frac{r(i)-2}{2}} \right\rfloor & \text{otherwise} \end{cases} \tag{9}$$

where $r(i) = \lceil 2\log_2(\frac{2i+8.243}{7.455})\rceil$ and n_{wmax} is given in Equation (4). The sequence number of the last packet transmitted in this round, $n_{max}(i)$, can be obtained using Equation (3) and is given by

$$n_{max}(i) = \begin{cases} N_{exp}\left\lceil \frac{\max(0, i-N_{exp})}{W_{max}} \right\rceil W_{max} & \text{if } r(i) > n_{wmax} \\ \left\lfloor 2^{\frac{r(i)+1}{2}} + 2^{\frac{4r(i)-3}{8}} - 2 - \frac{3\sqrt{2}}{2} \right\rfloor & \text{otherwise} \end{cases} \tag{10}$$

where $r(i) = \lceil 2\log_2(\frac{2i+8.243}{7.455})\rceil$ and n_{wmax} is given in Equation(4).

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 $\lceil\frac{i-n_{max}(i)+cwnd(i)-1}{2}\rceil$ acknowledgments 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 $i = 1, \cdots, 6$

$$t_{1\_loss}(N) = [t_{no\_loss}(i) + \beta + t_{linear}(N-k-1, 2) - 1]\, RTT \tag{11}$$

Observe that the first term in Equation (11) corresponds to the time to transmit the first i packets, β 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 acknowledgments are received the sender infers a loss and retransmits. As before, the time to transmit the first i packets is given by $t_{no\_loss}(i)$ from Equation (6). 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 three duplicate acknowledgments. Otherwise, a third duplicate acknowledgment 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

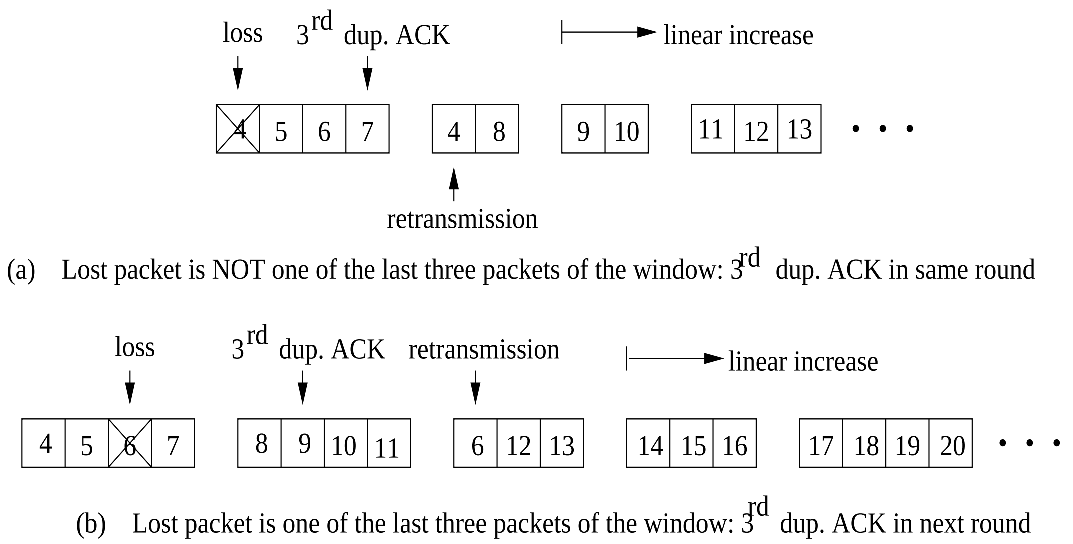


Figure 2: Window increase patterns for TCP flows with different positions of the lost packet in a window.

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 $\lceil\frac{i-n_{max}(i)+cwnd(i)-1}{2}\rceil$ acknowledgments 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 the receiving the third duplicate acknowledgment, the window halves, three added to $cwnd$, the lost is packet retransmitted and the window increased by one for every duplicate acknowledgment 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 acknowledgment is received. Again, on the receipt of the third duplicate acknowledgment, the window reduces to half, the lost packet is retransmitted and the window is increased by three. For every additional duplicate acknowledgment 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

$$k = \begin{cases} \min\left\{W_{max}, \left\lceil\frac{cwnd_{i+1}}{2}\right\rceil + cwnd_{i+1} - 1\right\} + i - 1 & \text{if } n_{max}(i) - i < 4 \\ \min\left\{W_{max}, \left\lceil\frac{cwnd_{i+1}}{2}\right\rceil + n_{max}(i) - i\right\} + i - 1 & \text{otherwise} \end{cases} \tag{12}$$

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 acknowledgment 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

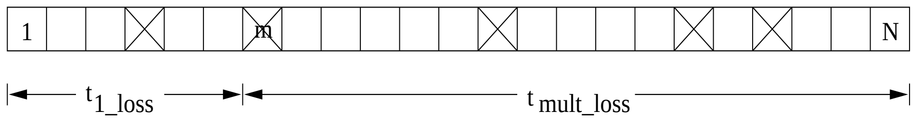


Figure 3: Modeling a flow by breaking it up into the section which has the first loss and the remaining section with the rest of the losses.

transfer N packets is then given by

For $i = 7, \cdots, N$

$$t_{1\_loss} = [t_{no\_loss}(i) + 3]\, RTT + \begin{cases} [t_{linear}(N-k, n) - n + 1]\, RTT, & \text{if } n_{max} - i < 3 \\ [t_{linear}(N-k, n) - n]\, RTT, & \text{otherwise} \end{cases} \tag{13}$$

where $n = \lceil \frac{cwnd_{i+1}}{2} \rceil$. Note that the term β does not appear in Equation (13) as there is no timeout.

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 (11) and (13) 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. Assuming uniform distribution of losses, the average distance (in number of packets) between two consecutive losses, D_{ave}, is given by

$$D_{ave} = \frac{N - m + 1}{M - 1} \tag{14}$$

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

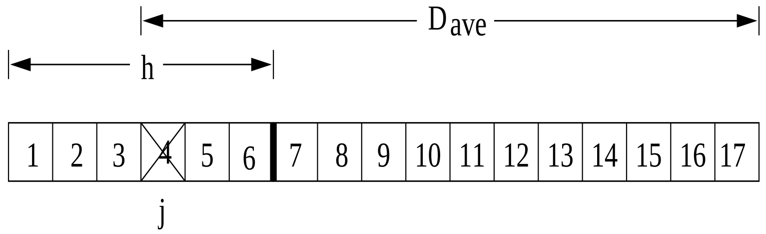


Figure 4: The parameters h and j and their relation to D_{ave}.

neglect the cases when the subsequent losses occur while the connection is in the slow start mode after a timeout). The duration of the timeout is again given by βRTT. 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. 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 acknowledgments 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

$$t_{timeout}(l) = [\beta + I(j > 1) + t_{linear}(D_{ave} - h, 2) - 2]RTT \tag{15}$$

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. As explained in the previous section, if the lost packet is one of the last three packets of the round, an additional round of packets is transmitted before the third duplicate acknowledgment is received. Again, on receiving the third duplicate acknowledgment, $cwnd$ 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 acknowledgment 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

$$k = \begin{cases} \min\{h + \lceil \frac{h}{2} \rceil - 1, W_{max}\} - 1 & \text{if } h - j < 3 \\ \min\{h - j + \lceil \frac{h}{2} \rceil, W_{max}\} - 1 & \text{otherwise} \end{cases} \tag{16}$$

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

$$t_{fast\_retrans}(l) = \begin{cases} \left[4 - \lceil\frac{h}{2}\rceil + t_{linear}(D_{ave}-k, \lceil\frac{h}{2}\rceil)\right] RTT, & \text{if } h-j<3 \\ \left[3 - \lceil\frac{h}{2}\rceil + t_{linear}(D_{ave}-k, \lceil\frac{h}{2}\rceil)\right] RTT, & \text{otherwise} \end{cases} \tag{17}$$

Note that Equations (15) and (17) have a form similar to that of Equations (11) and (13).

We can now use Equations (15) and (17) to calculate the average time between two losses and thus the transfer time of the remaining $N-m+1$ packets. We first find 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 time to transmit the first $m-1$ packets using Equations (11) and (13) 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

$$t_{mult\_loss}(N) = E\{t_{1\_loss}(m-1)\} + E\{(M-2)t_{timeout}(D_{ave})\} + E\{(M-2)t_{fast\_retrans}(D_{ave})\} \tag{18}$$

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

$$T_{transfer}(N) = t_{setup} + (1-p)^N t_{no\_loss}(N) + p(1-p)^{N-1}E\{t_{1\_loss}(N)\} + t_{mult\_loss}(N) + t_{dack} \tag{19}$$

where t_{setup}, $t_{no\_loss}(N)$ and $t_{mult\_loss}(N)$ are defined in Equations (1), (6) and (18) respectively and $t_{1\_loss}(N)$ is defined in Equations (11) and (13). A program to calculate the expected transfer time is available online and can be downloaded from `http://networks.ecse.rpi.edu/~bsikdar/t_time_tcp.c`

4 Modeling Infinite Flows

In this section we extend the analysis of the previous section to model infinite TCP flows and their steady state throughput. As in [11], we assume that the source has an unlimited amount of data to send. The loss model assumed for this section is the independent loss model introduced in Section 2 in contrast with the burst loss model of [11].

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 $p > 0$. Since these flows will have multiple losses, we can use Equations (15) and (17) to find the average transmission time between successive errors. Also, for a packet loss probability of p, the average number of packets transmitted between two successive losses, d, is given by $1/p$. Using this and the average transmission time between successive losses, 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

$$cw_{max} = min\left\{\left\lceil c\frac{-1+\sqrt{1+16/p}}{2}\right\rceil, W_{max}\right\} \tag{20}$$

where $c = \frac{(3+10RTT)(1-p)^2}{4(1+p+p^2)^3}$. The expected time to transfer d packets can then be written as

$$t_{ss}(p) = \frac{2}{cw_{max}(cw_{max}+1)}\left[\sum_{i=1}^{3}\sum_{j=1}^{i} t_{timeout}(d) + \sum_{i=4}^{cw_{max}}\sum_{j=1}^{i} t_{fast\_retransmit}(d)\right] \tag{21}$$

where $t_{timeout}(d)$ and $t_{fast\_retransmit}(d)$ are defined in Equations (15) and (17). The steady state throughput in bytes per second of a TCP connection is thus

$$R = \frac{dMSS}{t_{ss}} = \frac{MSS}{t_{ss}p} \tag{22}$$

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 [11], 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 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. We also verify the extension of our model to calculate the steady state throughput of TCP connections by comparing it with the results of [11].

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 midwest USA, west

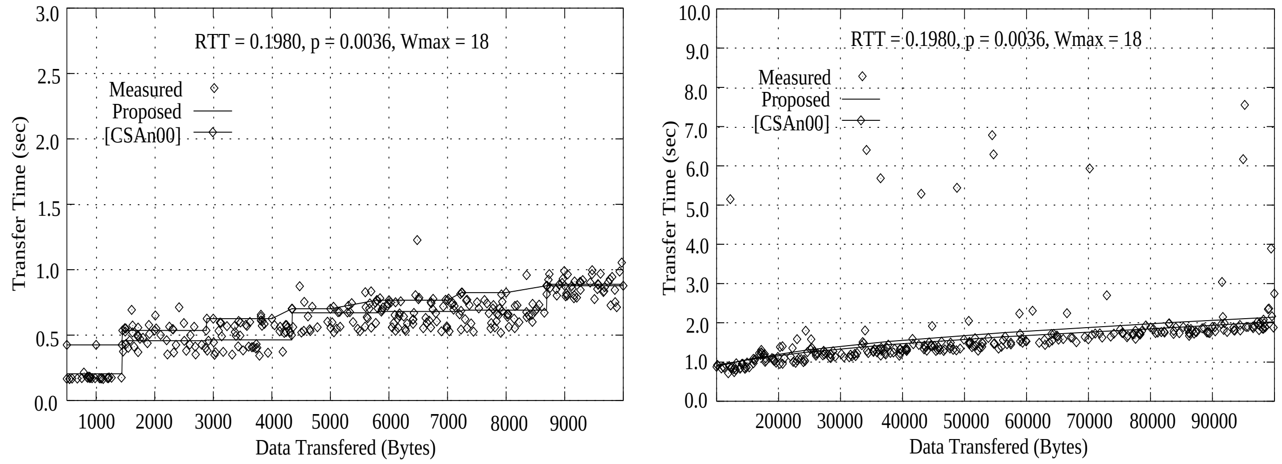


Figure 5: Transfer times of measured and modeled TCP transfer from the local machine to University of Pisa, Italy.

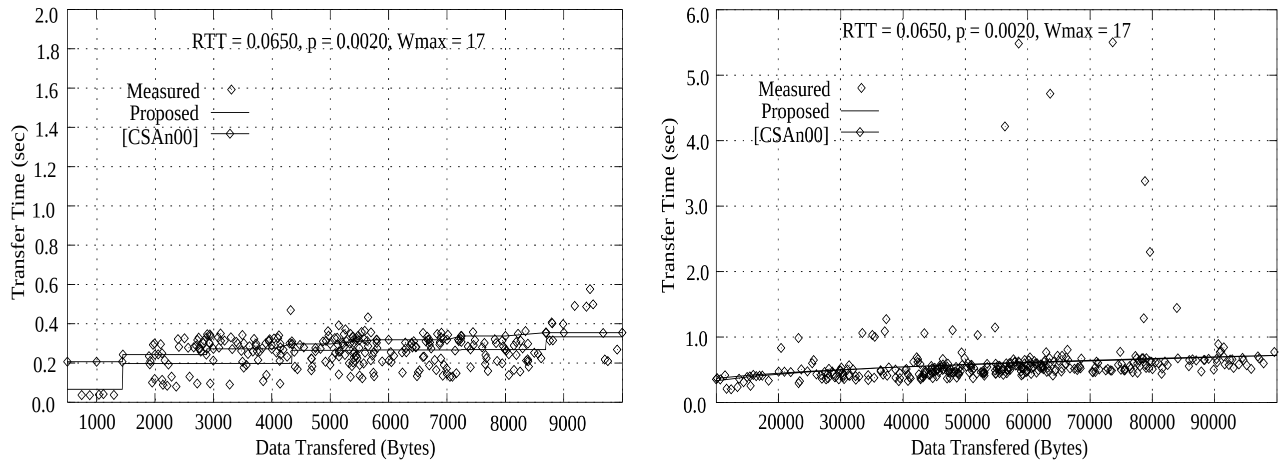


Figure 6: Transfer times of measured and modeled TCP transfer from the local machine to Ohio State University.

coast 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 HP-UX, FreeBSD CAIRN-2.5 and FreeBSD 3.3 respectively. In Figures 5, 6 and 7 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.

Note that our results are a significant improvement over the model proposed in [2], particularly in

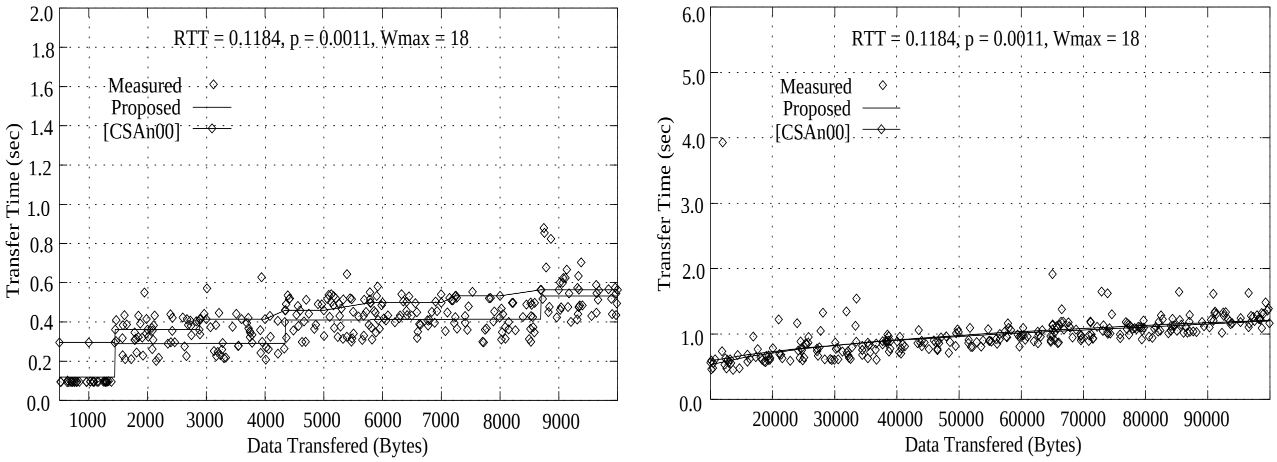


Figure 7: Transfer times of measured and modeled TCP transfer from the local machine to University of California at Los Angeles.

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.

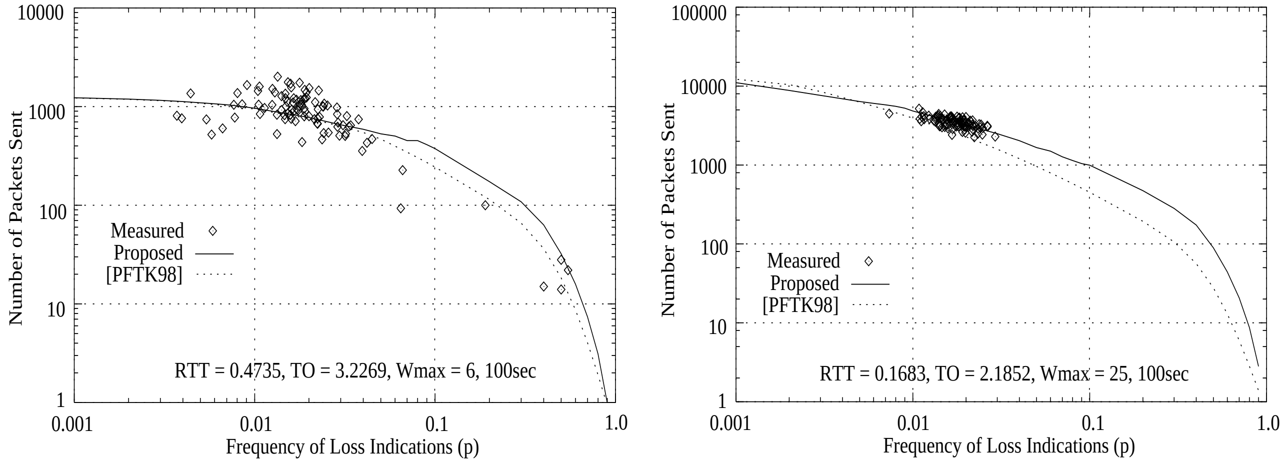
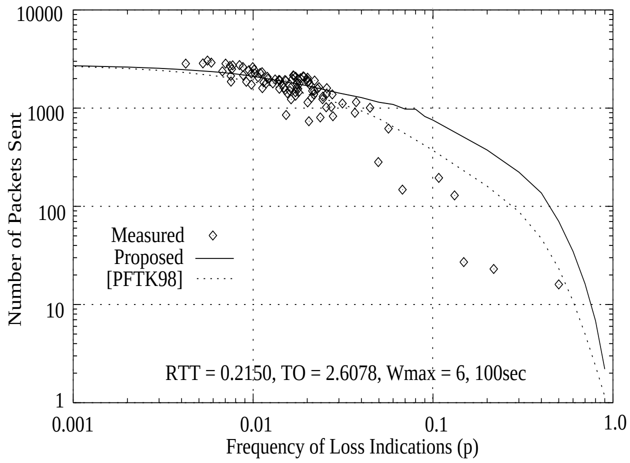


Figure 8: Steady state throughput of TCP connections from the proposed model compared with the model and measurements from [11] (Fig. 16 and 17 of [11]).

Figures 8 and 9 show 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. [11] and the traces reported therein. We plot the graphs for three of the 100 sec transfers of [11] (Fig. 13, 16 and 17 of [11]) and we see that our results match closely with both the measurements and the model proposed in [11]. We also note that there is a little deviation in the two models as the loss probability increases which might be due to the difference in the loss models. However, this does not seem to affect the accuracy



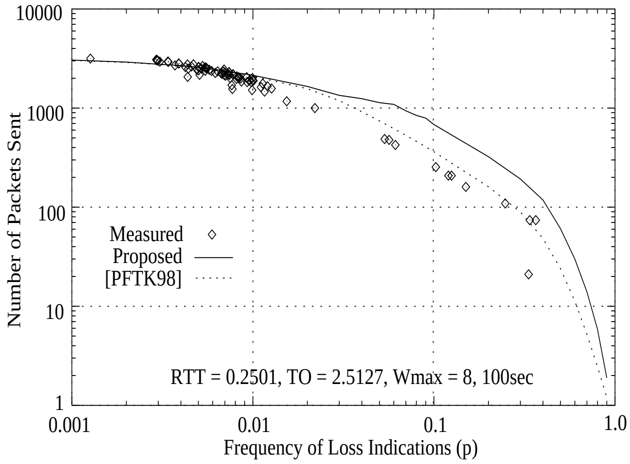


Figure 9: Steady state throughput of TCP connections from the proposed model compared with the model and measurements from [11] (Fig. 13 and 14 of [11]).

and validity of the results as our results are very close to the measured values.

6 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

$$T_{emp}(N) = \left[\log_{1.57} N + \{f(p, RTT)N + 4p\log_{1.57} N + 20p\} + \frac{(10+3RTT)}{4(1-p)W_{max}\sqrt{W_{max}}}N\right] RTT \quad (23)$$

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.

The transfer times predicted by the empirical model are compared with those obtained from Equation (19) 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. The empirical model can thus be used to get a rough

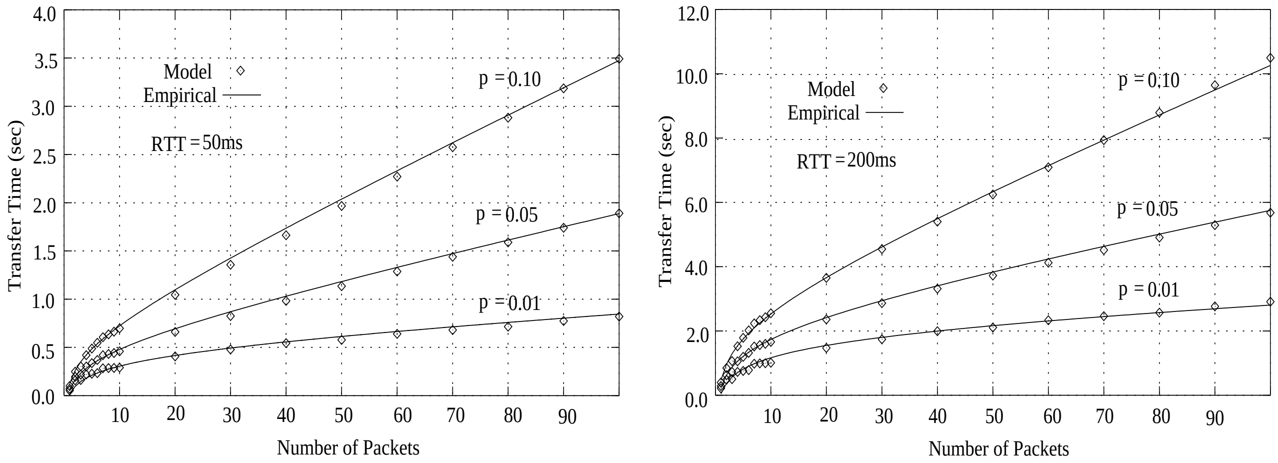


Figure 10: Transfer times from the empirical model and the analytic model of Equation (19). The first figure corresponds to a RTT of 50ms, while the second is for a RTT of 200ms.

indication of the dependence of TCP's latency on various parameters.

7 Sensitivity Analysis

The previous sections concentrated on the effect of the loss probability and RTT on TCP transfer times. Other factor 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 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 (19) 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

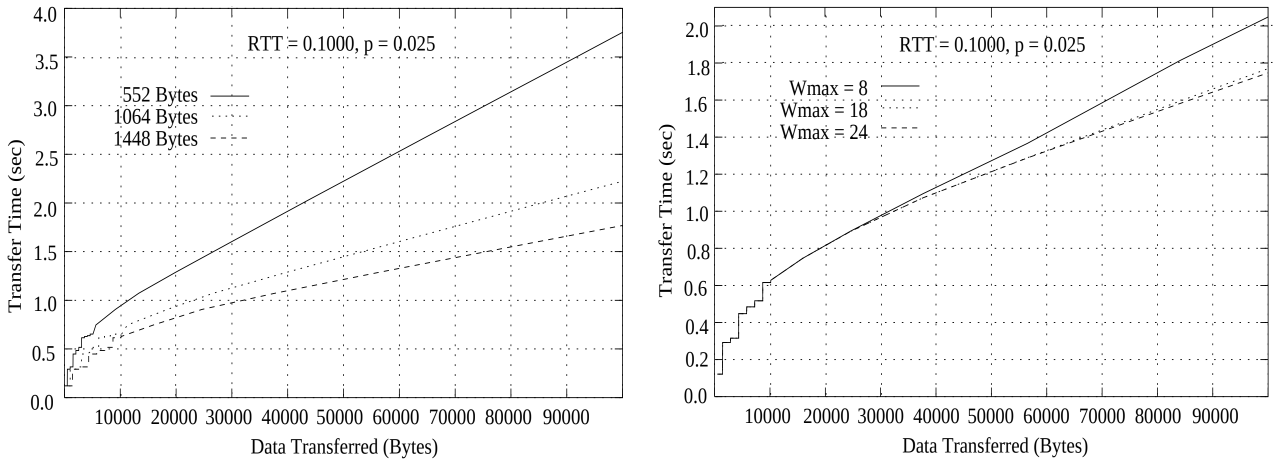


Figure 11: Effect of packet sizes and W_{max} on the transfer time of TCP connections.

scale linearly with the increase in the packet size.

8 Conclusion and Discussions

In this paper we presented a model for TCP transfer times which is capable of modeling TCP latencies for arbitrary file sizes. The model can also be applied to obtain the steady state throughput of TCP connections assuming infinite flows. In contrast to earlier models which either assume no loss [5, 12], infinite flows [11] or model the slow start of TCP connections in the beginning for the transfer [2], 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 than the one for finite TCP flows presented in [2]. 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. The model is verified using real life web based measurements. We also validate our steady state throughput model with the traces from [11] and show the the results are in very good agreement.

The paper also presented an empirical model for estimating the latency of TCP transfers. This model 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.

Acknowledgments

We would like to thank Prof. H. Balakrishnan of MIT, Prof. L. Zhang of UCLA, Prof. L. Rizzo of University of Pisa, Italy, Mr. D. Sisalem of GMD-Fokus, Berlin, Germany and Prof. D. Manjunath of Indian Institute of Technology, Bombay, India for providing us with accounts in their machines and Mr. J. Padhye of University of Massachusetts, Amherst for providing the traces for the steady state analysis.

References

[1] N. Cardwell, S. Savage and T. Anderson, "Modeling the performance of short TCP Connections," University of Washington, Seattle, Oct 1998.

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

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

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

[11] 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, Sep 1998.

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

[13] 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.