A Hysteresis Model for Web/TCP Transfer Latency

Yujian Li **Carey Williamson**
Department of Computer Science, University of Calgary, 2500 University Drive N.W.
Calgary, AB, CANADA T2N 1N4

Abstract

This paper presents an accurate stochastic model for transfer latency of short-lived Web-like TCP flows with random packet losses. Our model characterizes a data transfer in alternating cycles, with TCP state information carried over from one cycle to the next. Simulation experiments show that our model matches simulation results for short-lived flows better than earlier TCP models, and fits long-lived TCP flows as well. Our model is then extended to estimate transfer times for CATNIP TCP, which is shown to be 5-42% faster than TCP Reno, depending on transfer size and packet loss ratio.

1. Introduction

Response time is a primary concern for Web users. Users are unlikely to wait for a Web page that takes a long time to retrieve. However, continued growth in Internet traffic can cause congestion problems, leading to delays in Web page delivery due to packet loss. The Web, like many other Internet applications, uses the Transmission Control Protocol (TCP) as its transport layer protocol for reliable data transfer. The dynamics of TCP greatly affect Web performance (Claffy et al. 1998; Thompson et al. 1997), and TCP transfer latency often dominates Web response time.

Modeling TCP via mathematical analysis provides a way to characterize TCP performance quantitatively under specific operating conditions. In this paper, we focus on modeling short-lived TCP flows, which are representative of Web traffic. Measurements (Cunha et al. 1995; Mah, 1997; Thompson et al., 1997) show that the average size of TCP flows is generally less than 10 Kbytes for non-persistent connections, and 26-32 Kbytes with persistent connections.

Some TCP models for short-lived flows (e.g., Cardwell et al., 1998, 2000) are directly extended from TCP steady-state throughput analysis (e.g., Padhye et al. 1998). However the underlying assumptions of these models may not hold for short-lived transfers, since short-lived flows spend most of their time in the slow start phase, rather than in the congestion avoidance phase like long-lived flows.

More recent TCP models are derived explicitly for short-lived flows (e.g., Cardwell, et al., 1998; Heidemann et al. 1997; Mahdavi, 1997; Partridge & Shepard, 1997). However, some of these models (Heidemann et al., 1997; Mahdavi, 1997; Partridge & Shepard, 1997) do not consider packet losses, which can have a dramatic impact on TCP performance. Measurement results (Balakrishnan et al., 1998; Paxson, 1997) suggest that packet loss rates in the Internet can be as high as 5%. The work in (Cardwell et al., 1998) assumes that a packet loss always triggers a TCP timeout, which is overly pessimistic. Short-lived TCP flows could recover from losses using fast retransmit (Mathis et al., 1997; Padhye et al., 2000) and fast recovery, depending on which packet is lost and the congestion window size at the time of the loss. In addition, most research in the literature ignores subtle features of TCP, such as the timer-based TCP delayed acknowledgement (ACK) mechanism to reduce ACK packet overhead. Since delayed acknowledgments complicate matters, most models ignore its effects, or use a simple approximation. These simplifications inevitably compromise the accuracy of the models.

In this paper, we propose a stochastic model for short-lived TCP flows with random packet losses. The distinctive feature of our model is that we model a data transfer as alternating cycles, with TCP state information (e.g., congestion window) carried over from one cycle to the next. We believe that this "hysteresis" property is essential in capturing TCP dynamics effectively. In addition, our model includes delayed acknowledgment effects, time-outs, and the fast retransmission mechanism. Our simulation experiments show that our stochastic model matches the simulation results for short-lived flows more accurately than earlier models, and fits long-lived TCP flows as well.

The remainder of this paper is organized as follows. Section 2 provides background on TCP and related work on modeling TCP response time. Section 3 describes our proposed TCP model for short-lived flows. Section 4 describes the simulation experiments and results. Section 5 extends our proposed model to CATNIP TCP. Section 6 concludes the paper.

2. Background and Related Work

TCP Overview

TCP is a connection-oriented transport layer protocol, widely used on the Internet. It provides reliable data delivery through positive acknowledgement with retransmission, as well as flow control to prevent fast senders from overloading slow receivers.

There are two parts in the TCP congestion control algorithm, known as slow start and congestion avoidance. Successful transmission results in the congestion window size (cwnd) growing exponentially up to a threshold value (ssthresh), and then linearly thereafter.

TCP uses packet loss as an implicit signal of network congestion. When a packet loss happens, two possible events might occur: one is a retransmission time-out (RTO) at the sender; the other is the sender receiving duplicate ACKs to trigger fast retransmission. For TCP Reno, it

retransmits the lost packet, and reduces cwnd by half. This technique is called fast recovery.

Another technique widely adopted by TCP implementations is delayed ACK. This allows the receiver to delay the acknowledgement of a data packet for a short period of time - the delayed ACK interval.

Related Work

Stochastic models of TCP for short-lived flows can be classified into two types: models based on steady-state analysis, and models explicitly for short-lived flows. Four representative models are described here.

Padhye Model: Padhye et al. (1998) derived a model for the steady-state throughput of a bulk data transfer. This model later was used in (Cardwell et al., 1998) as the estimate for the bandwidth achieved for short-lived flows. By simply adding the cost of connection establishment and the expected cost of delayed ACKs, the TCP transfer latency for short-lived flows was constructed (Cardwell et al., 1998).

Cardwell-00 Model: Cardwell et al. (2000) provided another model for short-lived flows, based on the steady-state results in (Padhye et al., 1998). The authors decomposed the data transfer into four aspects: the initial slow start phase, the resulting packet loss (if any), the transfer of any remaining data, and the additional expected delay from the delayed ACK timer. The throughput of the flow after the first loss was estimated using steady-state analysis.

Cardwell-98 Model: Cardwell et al. (1998) developed a more detailed model of short-lived TCP flows. They viewed a short-lived TCP flow as an initial connection establishment handshake, followed by alternating phases of slow start and successive RTOs. The progress of a TCP connection is thus considered as a series of phases where TCP is sending data, and each sending phase is separated from the next by one or more RTOs. Fast retransmission is ignored.

Sikdar Model: Sikdar, Kalyanaraman, & Vastola (2001) presented a mathematical model for TCP flows of arbitrary size. Their model decomposed the TCP transfer latency into three cases: the no loss case, single loss case, and multiple loss case. For the window increase pattern in the slow start phase, the authors introduced the expression

$$packets(n) = \lfloor 2^{(n-1)/2} + 2^{(n-2)/2} \rfloor \quad (1)$$

to approximate the number of packets in the nth round, instead of the general exponential increase pattern (Cardwell, et al., 2000)

$$packets(n) = 1.5^n \quad (2)$$

Simulation-based comparisons (Li, 2002) show that TCP latency for short-lived flows is roughly logarithmic in the size of the data. Among these four models, the Cardwell-00 model provides more accurate predictions than other models. The Sikdar model performs well when the loss probability is 1% to 5%. Work in (Li, 2002) also shows that for long-lived TCP flows, TCP latency is approximately linear with the transfer data size. The explanation is that the flows are nearing steady state, where the transfer time is well modeled as $t \approx data / bandwidth$, for some steady-state bandwidth estimate. Models extended from the steady-state modeling work, such as the Cardwell-00 and the Padhye models, provide reasonable prediction for these scenarios.

3. A Hysteresis-based Model

Assumptions

The proposed model is based on the TCP Reno release from Berkeley (Stevens, 1994), which is still prevalent in the Internet today (Mathis et al., 1996; Padhye & Floyd, 2001). Since we are only concerned about modeling TCP performance, we assume that the link speed is high, the sender sends full-sized packets whenever the congestion window allows, and that the receiver advertises a consistent flow control window W_{max}.

We model the dynamics of TCP in terms of "rounds" as in (Cardwell et al., 2000; Padhye et al., 2000; Sikdar et al., 2001). A round starts when the sender sends a window of packets, and ends when one or more acknowledgements are received for these packets. We assume that the packet loss behavior follows the Bernoulli loss model, i.e., packet losses in one round are independent of the losses in any other round, and losses within a single round are independent. We allow packet loss indications by either RTO or triple duplicate ACKs.

We do not explicitly model the congestion avoidance algorithm. Rather we assume that cwnd always increases by one packet for each ACK (just as in slow start), to make the mathematics tractable. As demonstrated by the simulation results, this assumption has minimal effects on the accuracy of our model. The main reasons are threefold. First, the initial slow start threshold is typically set to 64 KB (Tanenbaum, 1996). Short-lived flows commence in slow start, and rarely reach large cwnd values, so they will spend a majority of their time in slow start. Second, TCP enters the congestion avoidance phase only when the congestion window size is 2 or more (Stevens, 1994). Third, when the window size is very small, the increase of the congestion window under slow start is similar to that under congestion avoidance.

The effect of delayed acknowledgement is also considered. The most common delay occurs whenever TCP suffers a packet loss and restarts with a cwnd of 1 packet. The receiver waits for a second packet, until finally its delayed ACK timer expires and it sends an ACK. In most UNIX based systems this timer is set to 200 ms, which leads to an expected delay of 100 ms before the ACK for the first packet of the flow is sent.

Model Overview

We adopt the analysis of connection establishment from (Cardwell et al., 2000) as the expected three-way-handshake duration, as in (Sikdar et al., 2001). However, in model validation, we focus on the data transfer part, and ignore the

three-way handshake latency. One reason is that the one-way TCP implementation in the ns-2 network simulator does not include the latency for the three-way handshake. Simplifying our model makes direct comparison to simulation results possible.

We model the data transfer part as alternating cycles. Each cycle includes a slow start phase and a successive packet loss phase, except that the last cycle has no packet loss phase. Figure 1 shows the evolution over time of the congestion window size in our model, where loss indications are either by RTO or triple-duplicate ACKs. In case of RTO, TCP re-enters slow start with cwnd set to 1 (e.g., the second loss in Figure 1). In case of triple duplicate ACKs, it triggers fast retransmission (e.g., the first loss in Figure 1). For this case, TCP reduces cwnd by half and then re-enters slow start (exponential increase), rather than congestion avoidance (linear increase).

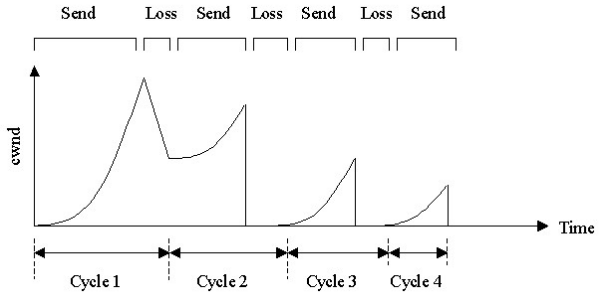


Figure 1: Congestion window evolution for the proposed model.

Let N represent the total packets to transfer initially, and d the remaining data to transfer. Initially, d is equal to N. TCP begins with the first cycle, where it sends data packets in slow start, quickly increasing its congestion window, until it detects a packet loss. The congestion window is reset to 1 (due to RTO) or reduced by half (due to triple duplicate ACKs). Then the TCP flow enters the next cycle until all data are transferred.

An important property of the proposed model is that it carries over state information (i.e., cwnd and remaining data to transfer) from one cycle to the next. This is different from regenerative Markov models (Cardwell et al., 2000; Padhye et al., 2000; Sikdar et al., 2001). The calculations for the slow start phase and the packet loss phase within each cycle are described in the following subsections.

The Slow Start Phase We follow the work in (Cardwell et al., 2000) to determine the expected latency for the slow start phase within each cycle. First, based on the Bernoulli packet loss assumption, the expected number of data packets sent in the slow start phase before a loss occurs, $E[d_{ss}]$, can be calculated as:

$$E[d_{SS}] = \sum_{k=0}^{d-1} (1-p)^k \, pk + (1-p)^d \, d = \frac{\left(1-(1-p)^d\right)(1-p)}{p} \tag{3}$$

With the exponential congestion window increase pattern shown in Equation 2, the number of slow start rounds to transfer $E[d_{ss}]$ packets of data is:

$$i = \log_\gamma \left(\frac{\gamma\text{-}1}{w_1} \cdot E[dss] + 1 \right) \tag{4}$$

where:

γ: congestion window increase factor, equal to 1.5;

w_1: initial window size for the current cycle. It is set to 1 for the first cycle.

$E[W_{ss}]$, the expected window size at the end of slow start (ignoring W_{max}) is calculated as:

$$E[W_{SS}] = \frac{\gamma\text{-}1}{\gamma} \cdot E[d_{SS}] + \frac{w_1}{\gamma} \tag{5}$$

Finally, the time to send $E[d_{ss}]$ packets of data in slow start can be calculated as:

$$E[T_{SS}] = \begin{cases} RTT \cdot [\log_\gamma (W_{\max}/w_1) + 1 + \frac{1}{W_{\max}} (E[d_{SS}] - \frac{\gamma W_{\max} - w_1}{\gamma - 1})] & E[W_{SS}] > W_{\max} \\ RTT \cdot \log_\gamma \left(\frac{E[d_{SS}](\gamma - 1)}{w_1} + 1 \right) & E[W_{SS}] \le W_{\max} \end{cases} \tag{6}$$

After the initial slow start phase, the remaining data to transfer is $d=N-E[d_{ss}]$.

The Packet Loss Phase We use an analysis similar to (Cardwell et al., 2000) to determine the expected cost of packet loss within each cycle. First, the probability that slow start ends with a packet loss is:

$$p_{SS} = 1 - (1-p)^d \tag{7}$$

The probability that TCP detects a packet loss with RTO is adopted from (Padhye et al., 2000) as follows:

$$Q(p,w) = \min\left(1, \frac{1-(1-p)^3}{1-(1-p)^w} \cdot \left(1 + (1-p)^3 \cdot \left(1 - (1-p)^{w-3}\right)\right)\right) \tag{8}$$

It is a function of packet loss rate (p) and window size (w). The expected cost of an RTO is derived in (Padhye et al., 2000):

$$E[Z^{TO}] = \frac{f(p)}{1-p} \cdot T_0 \tag{9}$$

where T_0 is the average duration of the first timeout in a sequence of one or more successive timeouts, and f(p) is given by:

$$f(p) = 1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6 \tag{10}$$

The expected duration of a fast recovery period is a single round trip time (RTT) (Cardwell et al., 2000). The algorithm to calculate the cost for the packet loss phase within each cycle is as follows:

Generate a random number RAND
If (RAND<=Q(p,w))
    $T_{loss}=p_{ss}$*E[ZTO] // RTO*
Else
    $T_{loss}=p_{ss}$ *RTT // fast retransmit*

We use the above algorithm, rather than a combination of weighted loss by RTO and loss by triple duplicate ACKs (Cardwell et al., 2000), to model the cost of the packet loss phase within each cycle. The reason is based on the choice of the initial window size for the next cycle, and consideration of the effect of delayed acknowledgements. The congestion window is set to 1 packet in the beginning. For each cycle, if packet loss triggers a RTO, cwnd is reset to 1 for the next cycle to resume slow start. If the packet loss is detected with triple duplicate ACKs, the initial window for the next cycle is reduced to half of the previous congestion window size:

$$w_1 = \frac{1}{2} \cdot E[W_{ss}] \tag{11}$$

The expected cost for delayed acknowledgment, T_{delack}, is added to the latency for the current cycle if slow start commences from a window size of 1 for the current cycle. This is different from previous work (Cardwell et al., 1998, 2000; Sikdar, et al., 2001), which simply adds T_{delack} as a one-time cost in the overall TCP latency.

Total Latency Combining the results of the previous subsections, the transfer latency for a flow of N packets is the sum of the latencies for all of the alternating cycles:

$$T = \sum_{i=1}^{lastCycle} \left(E[T_{ss}]^i + T_{loss}^i \right) \tag{12}$$

Note that Equation 12 is a prediction of latency experienced by a "typical" flow given the input parameters. Because the packet loss case within each cycle is randomly determined, Equation 12 does not necessarily yield a closed form unique answer. Using the algorithm to sample "typical" transfer latencies is necessary to get the expected time for a data transfer given the particular parameters.

4. Simulation Experiments

Methodology

We use the ns-2 network simulator for the simulation experiments. Figure 2 shows the simple network topology used in our experiments. Since there is no competing cross-traffic, we attached a packet error model to the router R1 to create random packet losses. For CATNIP TCP validation, we created our own CATNIP Error Model since ns-2 does not have a CATNIP loss model built-in. This model drops packets of low priority with probability p, and packets of high priority with probability p'. The Bernoulli loss model is a special case of the CATNIP error model when $p' = p$. We use FTP as the application model for sending a specified number of packets over a 10 Mbps link. However, we set the number of packets in each flow to be representative of HTTP Web transfer sizes.

Each experiment consists of 1000 trials with different seeds for the random packet loss generation process. This number is adequate to provide a good estimate of the TCP latency distribution, and thus the expected latency. In each trial, a Reno TCP agent on the server opens a connection and immediately begins sending the required amount of data. Once the last data packet is acknowledged, the data transfer is finished.

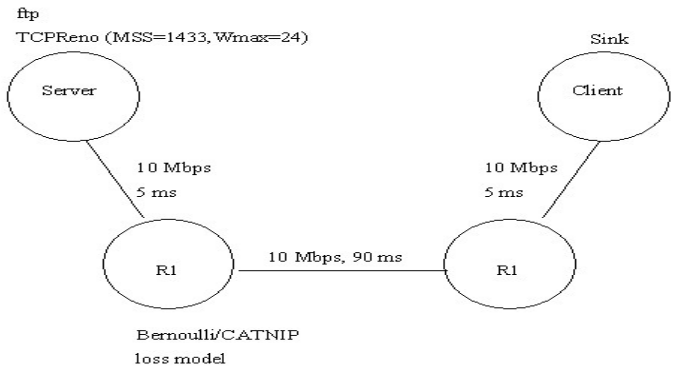


Figure 2: Simulated network topology

The primary performance metric is the data transfer time, by which we mean the time from when the sender sends the first packet until the time when the sender receives the acknowledgement of the last data packet. The time for three-way handshake establishment and teardown are not simulated in the Reno TCP agent. Hence, our experiments focus only on the data transfer time.

There are two main factors in our simulation experiments: transfer size, and packet loss probability. A one-factor-at-a-time experiment is conducted using these factors. A summary of the experimental design appears in Table 1. Only a subset of the simulation results are presented here.

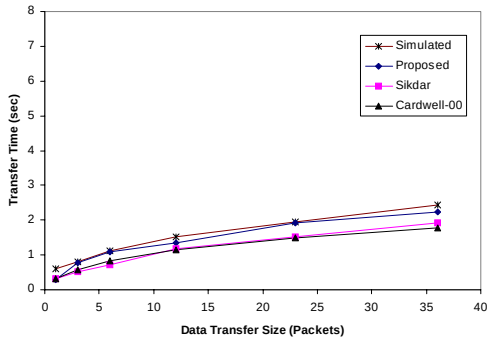
Table 1: Experimental factors and levels

Factor	Levels
Transfer Size (KB)	1, 4, 8, 16, 32, 50, 64, 90, 110, 128, 160, 180, 200
Packet Loss Probability	1%, 3%, 5%, 8%, 10%

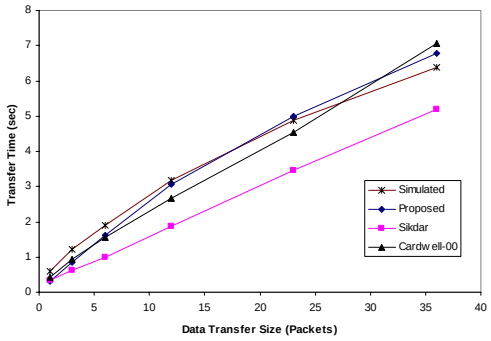
Transfer sizes are divided into short-lived TCP flows from 1 KB to 50 KB (1 – 36 data packets), and long-lived flows from 50 KB to 200 KB (36 – 143 data packets). For clarity, we present the results for short-lived and long-lived TCP flows separately. The default parameters for simulations are based on values that are representative of Internet Web traffic: MSS=1433 bytes, 200 ms RTT, W_{max}=24. The minimum RTO timer is set to 1 second. When any successive retransmission timer is set, its timeout is set using TCP's exponential back-off algorithm (Stevens, 1994).

Comparison with Previous TCP Models

Figure 3 compares TCP latencies between our model and the Cardwell-00 and Sikdar models for short-lived TCP



(a) p=1%



(b) p=10%

Figure 3: Simulated and modeled TCP transfer latencies for small transfer sizes.

flows transferring between 1 KB and 50 KB, when packet loss probability is 1% and 10%, respectively. Our model fits the simulation results closely for these loss probabilities. Our model's prediction values match the simulated values better than those obtained by the other models. We further compare the relative error (variance) estimation (Law & Kelton, 2000) for a given loss probability:

$$V(p) = \frac{\Sigma transfers \left(T_{predicted}(p) - T_{simulated}(p)\right)^2}{Number \quad of \quad transfers} \quad (13)$$

where $T_{predicted}$ is the predicted value and $T_{simulated}$ is the simulated value. A smaller error (or variance) implies a better model accuracy. Table 2 provides the relative error estimation for short-lived flows. It shows that in all cases the relative error is less than 10% for our hysteresis model, while it often exceeds 10% for the Cardwell-00 model and 15% for the Sikdar model.

While our model and the Cardwell-00 model both have an initial slow start and packet loss phase, our model improves significantly upon the Cardwell-00 model. To estimate the time to send any data remaining after the first loss, the Cardwell-00 model applies the steady-state throughput result from long-lived TCP flows (Padhye et al., 1998). This is not a good approximation due to different behavior for short-lived TCP flows (Sikdar et al., 2001), and it introduces several errors (Cardwell et al., 2000). Our proposed model does not use the results from the steady-state model (Padhye et al., 1998) for the remaining data. Instead, we apply the same mathematics used for the first cycle (including the initial slow start and the first packet loss) to the remaining data. This is similar to the situation that a typical TCP flow suffers with random losses, i.e., the flow periodically experiences slow start and successive packet loss. The success of our model comes from TCP state information carried over from one cycle to the next.

Another source of improvement for our model over the Cardwell-00 model and other models (Cardwell et al., 1998; Sikdar et al., 2001) is the consideration of delayed acknowledgements. The Cardwell-00 model simply adds the expected cost of delayed ACK, T_{delack}, once as one component of TCP latency. In our model, T_{delack} is naturally integrated into the packet loss phase whenever a packet loss triggered by RTO resets cwnd to 1 packet.

Table 2: Relative error comparison of the hysteresis model with existing models for short-lived flows.

Model	p=1%	p=3%	p=8%	p=10%
Sikdar	0.15	0.41	0.51	1.06
Cardwell-00	0.17	0.44	0.10	0.15
Hysteresis	0.03	0.05	0.05	0.08

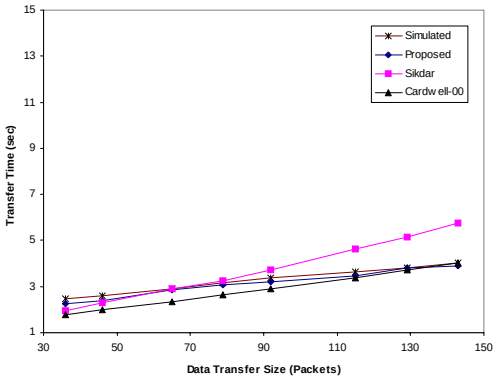
Compared with the Sikdar model (Sikdar et al., 2001), our model includes not only the effect of round trip time and loss probability as given in the Sikdar model, but also the factors such as T_{delack}, and loss detection by RTOs and triple duplicate ACKs. Hence, the proposed model captures the dependence of TCP latency on delayed acknowledgements, and the cost of loss events triggered by either RTO or triple duplicate ACKs.

We further conducted experiments for long-lived flows as shown in Figure 4, with statistical comparison given in Table 3. Surprisingly, the results from our model match very closely with simulation results, and are a significant improvement over the Cardwell-00 and Sikdar models.

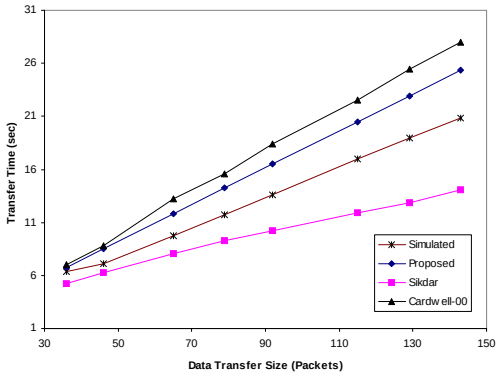
Table 3: Relative error comparison of the hysteresis model with existing models for long-lived flows.

Model	p=1%	p=3%	p=8%	p=10%
Sikdar	0.78	1.82	4.75	16.12
Cardwell-00	0.22	0.47	4.20	22.24
Hysteresis	0.02	0.09	2.20	8.66

When the loss probability is 10%, our proposed model overestimates the transfer latency for long-lived flows, as does the Cardwell-00 model. The main reason comes from Equation 8, i.e., the probability that a sender detects a packet loss with RTO, which we adapted from the steady-state TCP model analysis in (Padhye et al., 2000). We observe that this function is sensitive to the packet loss probability. For example, with the packet loss probability of 10%, about 10 packets are expected to be sent before a packet is lost. Assume the congestion window size is 10



(a) p=1%



(b) p=10%

Figure 4: Simulated and modeled TCP transfer latencies for large transfer sizes

data packets. When the packet loss probability increases from 3% to 10%, the probability that a packet loss is detected via RTO increases from 0.39 to 0.57. The smaller the congestion window size is, the higher is the probability that a packet loss requires an RTO. Thus, for a transfer of 92 data packets when the packet loss probability is 10% (the expected number of losses is 9.2), all the losses are expected to be detected via RTO. This adds a lot of time to the data transfer time in the model, but is not the case for simulations.

5. Extension to CATNIP TCP

In this section, we demonstrate the generality of our TCP model by extending it to model CATNIP TCP. Recently, Wu and Williamson (2002) proposed CATNIP TCP, a Context-Aware Transport/Network Internet Protocol for the Web. CATNIP TCP classifies packets in a transfer into two categories (high priority and low priority) based on their potential impact on TCP response time when lost. CATNIP TCP allows network routers to use this priority information to make smarter packet-discard decisions when congested, rather than using (for example) Drop-Tail (Padhye et al., 2000) or Random Early Detection (RED) (Mah, 1997) as a queue management strategy.

Extending the Model to Partial CATNIP TCP

Due to the difficulty of modeling the TCP congestion window dynamics for a target packet loss probability, we approximate CATNIP TCP by considering only the first 3 and last 3 packets as high priority packets. That is, we ignore the high priority marking of any intermediate packets sent when $cwnd \le 3$. With this pessimistic assumption, our stochastic model is easily extended to CATNIP TCP. Assume that there are N packets to transfer. The packet loss probability for any of the high priority packets is p', and the loss probability of any other packets is p, where $p' \le p$. Considering that each packet loss is independent of sender behavior, the modeling work can be decomposed into three steps:

1. Transfer the first 3 packets.
2. Transfer the N-6 packets assuming there are N-3 packets to transfer.
3. Transfer the last 3 packets.

Note that for step 2, we count the time to transfer N-6 packets with the total number of packets to transfer as N-3, rather than N-6. Otherwise, Equation 3 in the model generates fewer data packets than expected to be sent in the slow start phase before a loss occurs for each cycle. This forces more cycles to complete a data transfer, and thus overestimates the transfer latency. Also for step 3, the initial congestion window size, w_1, can be calculated based on the value of $E[W_{SS}]$ at the end of last round when transferring the N-6 packets, as $w_1 = E[W_{SS}] \cdot \gamma$. If we consider the limitation of the maximum congestion window size, there are restrictions on w_1 as follows:

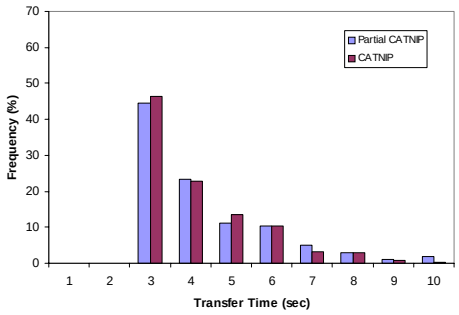
$$if\ w_1 \ge W_{\max},\ w_1 = W_{\max}.$$

$$if\ w_1 < 2.0,\ w_1 = 2.0.$$

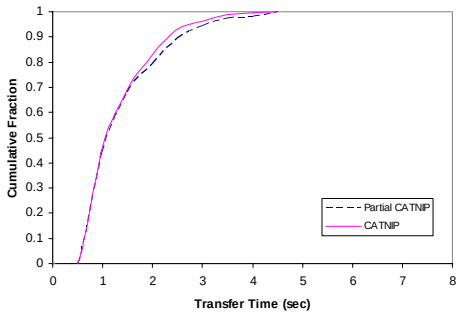
By applying the new model to each step and summing the latencies, we obtain the expected transfer latency for the entire transfer. The above approach to model CATNIP TCP is referred to as the Partial CATNIP TCP model, because it does not consider any intermediate high priority packets sent when $cwnd \le 3$. The differences between Partial CATNIP TCP and CATNIP TCP are compared in simulation experiments.

Comparison between CATNIP TCP and Partial CATNIP TCP

Figure 5 compares the frequency distribution and cumulative distribution of transfer times between CATNIP TCP and Partial CATNIP TCP for a short-lived flow transferring 32 KB when the packet loss probability is 5%.

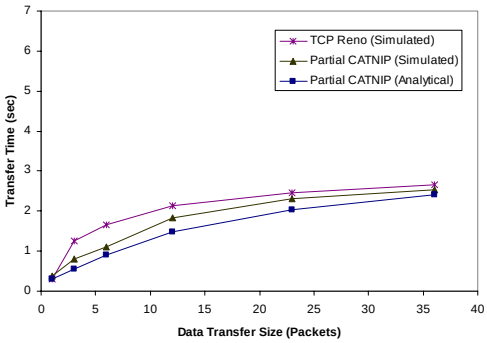


(a) p=5% (PDF)

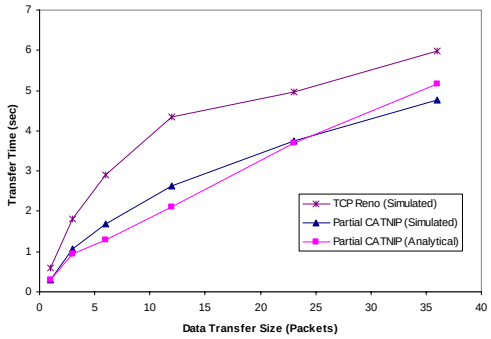


(b) p=5% (CDF)

Figure 5: Comparison between CATNIP & Partial CATNIP for 32 KB data transfers



(a) p=1%



(b) p=10%

Figure 6: Simulated and modeled transfer latencies of short-lived flows for Partial CATNIP

The transfer time distributions of CATNIP TCP and Partial CATNIP TCP are almost the same. This is because the expected number of losses (congestion window decrease events) is about 1, and thus the number of packets sent when $cwnd \leq 3$ (except the first 3 and last 3 packets) is small. So the difference between CATNIP TCP and Partial CATNIP TCP is not apparent.

Performance Evaluation for Partial CATNIP TCP

Figures 6 shows the simulated transfer latencies for short-lived flows when the loss probability p for low priority packets is 1% and 10%, and the loss probability p' for high priority packets is 0%. This is the best possible case for Partial CATNIP TCP. Obviously, Partial CATNIP TCP improves the TCP latencies compared to TCP Reno. When the data transfer size increases from 6 to 36 data packets, the ratio of the number of high priority packets over the total number of packets in a transfer decreases from 100% to 17%. Correspondingly, the transfer time of Partial CATNIP TCP is less than that of TCP Reno when p is set at 1% and 10%, and the difference ranges by 5-32% and 20-42%, respectively. Thus, we can conclude that Partial CATNIP TCP is 5-42% faster than TCP Reno, depending on the transfer size and the packet loss ratio.

To evaluate the Partial CATNIP TCP model, the predicted transfer latencies are also plotted in Figure 6. Regardless of the loss probability p, the Partial CATNIP TCP model fits the simulation closely. Partial CATNIP TCP model provides relative error less than 22%.

6. Summary

In this paper, we propose an accurate TCP latency model for short-lived TCP flows with random packet loss, which reflect current TCP transfers carrying Web traffic. We model a data transfer as alternating cycles, with TCP state information (i.e., congestion window size and remaining

data to transfer) carried over from one cycle to the next. This is different from previous Markov regenerative models. Simulation results using the ns-2 simulator show that our model fits the simulated values closely for a wide range of packet loss probabilities, performing better than previous models. Simulation results also show close agreement for long-lived TCP transfers.

We also extended our model to Partial CATNIP TCP. Partial CATNIP TCP is different from CATNIP TCP in that it does not consider as high-priority intermediate packets of the flow sent when $cwnd \leq 3$. Simulation experiments and statistical analysis indicate that the transfer times of Partial CATNIP TCP are within 15% of those of CATNIP TCP. The validation experiments demonstrate that the Partial CATNIP TCP model fits the simulation closely.

In addition, performance comparisons between Partial CATNIP TCP and TCP Reno demonstrate that for short-lived flows, Partial CATNIP TCP is 5-42% faster than TCP Reno when packet loss probability is less than 10%. This shows that CATNIP TCP is a suitable approach to improve TCP performance. The results provide further insight into CATNIP TCP performance.

References

Balakrishnan, H., Padmanabhan, V., Seshan, S., Katz, R., & Stemm, M. (1998, April). TCP behaviour of a busy Internet server: analysis and improvements. *Proc. of IEEE INFOCOM*, San Francisco, CA.

Cardwell, N., Savage, S., & Anderson, T. (1998, October). Modeling the performance of short TCP connections. Retrieved May 1, 2003, from *http://www.cs.washington.edu/homes/cardwell/quals/quals-paper.ps*

Cardwell, N., Savage, S., & Anderson, T. (2000, March). Modeling TCP Latency. *Proc. of IEEE INFOCOM*, Tel Aviv, Israel.

Claffy, K., Miller, G., & Thompson, K. (1998, July). The nature of the beast: recent traffic measurements from an Internet backbone. *Proc. of INET.*

Cunha, C., Bestavros, A., & Crovella, M. (1995, July). *Characteristics of WWW client based traces* (*Tech. Rep. BU-CS-95-010).* Boston University.

Heidemann, J., Obraczka, K., & Touch, J. (1997, October). Modeling the performance of HTTP over several transport protocols. *IEEE/ACM Transactions on Networkin*g, *5*(5), 616-630.

Law, A., & Kelton, W. (2000). *Simulation Modeling and Analysis* (3rd ed.). New York: McGraw-Hill.

Li, Y. (2004). *Modeling Web/TCP Transfer Latency*. Master of Science Thesis, Department of Computer Science, University of Calgary, Calgary, AB.

Mah, B. (1997, April). An empirical model of HTTP network traffic. *Proc. of IEEE INFOCOM*, Kobe, Japan.

Mahdavi, J. (1997, April). TCP performance tuning. Retrieved May 1, 2003, from *http://www.psc.edu/netowrking/tcptune/slides*

Mathis, M., Semke, J., Mahdavi, J., & Ott, T. (1997, July). The macroscopic behaviour of the TCP congestion avoidance algorithm. *ACM Computer Communication Review, 27*(3), 67-82.

Mathis, M., Mahdavi, J., Floyd, S., & Romanow, A. (1996, October). TCP Selective Acknowledgement Options. *RFC 2018*, IETF.

Padhye, J., Firoiu, V., & Towsley, D, & Kurose, J. (1998, September). Modeling TCP Throughput: a simple model and its empirical validation. *Proc. of ACM SIGCOMM,* Vancouver, BC.

Padhye, J., & Floyd, S. (2001, August). On inferring TCP behaviour. *Proc. of ACM SIGCOMM*, San Diego, CA.

Padhye, J., Firoiu, V., & Towsley, D. (2000, April). Modeling TCP Reno performance: a simple model and its empirical validation. *IEEE/ACM Transactions on Networking*, *8*(2), 303-314.

Partridge, C., & Shepard, T. (1997, September). TCP/IP performance over satellite links. *IEEE Network*, *11*(5), 44-49.

Paxson, V. (1997, September). End-to-end Internet packet dynamics. *Proc. of ACM SIGCOMM*, Cannes, France.

Sikdar, B., Kalyanaraman, S., & Vastola, K. (2001a). An integrated model for the latency and steady-state throughput of TCP connections. *Performance Evaluation*, *46*(2), 139-154.

Stevens, W. (1994). *TCP/IP Illustrated*, *Vol. 1*, Addison Wesley.

Tanenbaum, A. (1996). Computer Networks (3rd ed.). Englewood Cliffs, NJ: Prentice-Hall.

Thompson, K., Miller, G., & Wilder, R. (1997, November). Wide-area Internet traffic patters and characteristics. *IEEE Network*, *6*(11), 10-23.

Wu, Q., & Williamson, C. (2002, March). Context-aware TCP/IP. *ACM Performance Evaluation Review*, *29*(4), 11-23.