# Generalized stochastic performance models for loss-based congestion control

Michele C. Weigle [a,*], Li Cheng [b], Jasleen Kaur [c], V. Kulkarni [b]

[a] Department of Computer Science, Old Dominion University, Norfolk, VA 23529, USA
[b] Department of Statistics and Operations Research, University of North Carolina, Chapel Hill, NC 27599, USA
[c] Department of Computer Science, University of North Carolina, Chapel Hill, NC 27599, USA

## ARTICLE INFO

## ABSTRACT

In this paper, we propose a generalized framework for modeling the behavior of prominent congestion-control protocols. Specifically, we define a general class of loss-based congestion-control (LB-CC) mechanisms and demonstrate that many variants of TCP, including those being proposed for high-speed networks, belong to this class. Second, we develop a stochastic model to predict the transfer time for bulk transmissions by any protocol belonging to the LB-CC class—our model predicts both the mean as well as the variability in the transfer time. Our model is applicable to a wide set of transfer types and network capacities. We validate our model through extensive simulations under controlled settings, as well as with comprehensive HTTP workloads.

We use our empirical analysis to also provide insights into several important issues, including: (i) identifying the settings under which previously-proposed TCP models are accurate, and (ii) identifying the conditions under which only steady-state analysis can be sufficient in modeling transfer performance. Our generalized framework provides a powerful tool that can be used in the design, analysis, and comparison of next-generation transport protocols. We demonstrate this benefit by comparing prominent TCP proposals for high-speed networks.

© 2009 Elsevier B.V. All rights reserved.

## 1. Introduction

TCP is the most widely-used transport protocol in the Internet [1]. Analytical models that accurately predict the performance of a TCP transfer on a given Internet path are needed for several reasons. First, such models can be used to understand how well Internet's dominant transport protocol works under different network and end-host settings. Second, such models are useful in distributed routing frameworks that, for a given TCP transfer, select the best path from a candidate set [2]. Third, these models are an essential ingredient in distributed computing frameworks—such as the GRID [3]—that need to incorporate the cost of network transfers and server computations in deciding how to distribute heavy-duty scientific computations. Finally, TCP models lie at the basis of the design of TCP-friendly congestion control mechanisms [4,5].

The TCP protocol itself is subject to change over time as newer versions are developed and deployed [6–12]. A TCP performance model is most useful when it incorporates this diversity and can be used to compare the different versions. Many TCP performance models have been proposed in the literature over the last decade [13–26]. Unfortunately, many of these models can be applied to very few (and often no more than one) variants of TCP. Furthermore, *all* these models predict either an expected steady-state throughput or an expected transfer time for a given TCP transfer—none of these estimate the *variability* in the transfer times. In this paper, we propose a class of TCP performance models that address these limitations.

Specifically, we make the following contributions. First, we formulate a simple framework that characterizes several TCP protocol variants. In particular, we define a general class of loss-based congestion-control (LB-CC) mechanisms and demonstrate that many variants of TCP, including those being proposed for high-speed networks, belong to this class. Second, we develop a stochastic model to predict the transfer time for bulk transmissions by any protocol belonging to the LB-CC class—our model predicts both the mean and the variability in the transfer time. Our model is applicable to short as well as long transfers and is applicable in more diverse settings of transmission capacity than previous models. We validate our model through extensive simulations under controlled settings, as well as with comprehensive HTTP workloads. Third, through computations and simulations, we identify the settings under which previously-proposed TCP models are accurate. In particular, we draw insights into the question: *when is steady-state-only analysis sufficient for modeling the performance of TCP connections?* Our generalized framework provides a powerful tool that can be used in the design, analysis, and comparison of

* Corresponding author. Tel.: +757 683 6001; fax: +757 683 4900.
  E-mail address: mweigle@cs.odu.edu (M.C. Weigle).

next-generation transport protocols. We demonstrate this benefit by comparing prominent TCP proposals for high-speed networks.

The rest of this paper is organized as follows. In Section 2, we outline our modeling objectives and approach. The LB-CC class is defined in Section 3. In Section 4, we present our transient analysis model for the LB-CC class. The model is validated using extensive simulations and real traces in Section 5. In Section 6, we discuss the computation efficiency of our model, and in Section 7, the applicability of steady-state analysis. We summarize our conclusions in Section 8.

## 2. Objectives and approach

TCP performance models help predict the time it would take to transfer a given number of bytes between two Internet hosts. In this section, we derive the requirements that such models should satisfy and discuss the state of the art in existing models. To inform this discussion, we begin by briefly reviewing the basic mechanisms used in TCP and the factors that impact TCP performance.

### 2.1. The transmission control protocol

TCP provides a reliable, in-order byte-stream service to applications. TCP senders transmit application bytes in chunks, called *segments*, and receivers send cumulative *acknowledgments* (ACKs) to indicate the successful receipt of each segment. Lost segments are detected and retransmitted on the receipt of multiple (typically three) duplicate ACKs (referred to as Fast Retransmit) for earlier segments. Additionally, a *timeout* is used to trigger retransmission of segments that are not acknowledged. TCP receivers guarantee in-order delivery to the application by buffering segments that arrive out-of-order.

TCP also provides flow control and congestion control semantics. For this, it employs a window-based sending mechanism in which senders limit the maximum number of unacknowledged segments they transmit to a value, namely the *Send Window* ($W$). In order to provide flow-control, $W$ is not allowed to grow more than the flow control limit, $K$, which is the minimum of the sender and receiver-advertised window sizes. TCP implements congestion control using three kinds of mechanisms to maintain $W$: (i) increasing $W$ on receiving indications (ACKs) of successful segment transmissions; (ii) detecting the occurrence of congestion on the path between the sender and receiver; and (iii) responding to congestion indications by reducing $W$. Different versions of TCP differ in the mechanisms and policies that are used for each of these three tasks. Most TCP variants—including Tahoe, Reno [27], SACK, Scalable [10], High-speed [28], and BIC [11]—rely on packet losses to detect congestion. Some versions—such as Vegas [6] and Fast-TCP [9]—additionally rely on increase in segment round-trip times to detect the onset of congestion. Explicit Congestion Notification (ECN) can also be used to signal congestion in the network. An ECN-enabled TCP Reno sender will reduce $W$ upon receiving a congestion signal in the same manner as it would on detecting a segment loss.

TCP versions differ more significantly in how they update $W$ (see Section 3). The general principle, though, is that TCP senders are more aggressive in reducing their send window on detecting congestion than they are in increasing it in the absence of congestion. In addition, most TCP versions define a *Slow Start* phase in order to achieve fast start-up behavior. In Slow Start, $W$ is incremented aggressively until it reaches the slow-start threshold, $S$. When the window is above $S$, a TCP sender is said to be in the *Congestion Avoidance* phase.

### 2.2. Factors that impact TCP performance

The throughput of a TCP session at any given time is governed by the value of $W$ as well as the time it takes for all segments in the window to get acknowledged. A number of factors impact the growth of $W$. First, different *TCP versions* react differently to indications of successful transmission or to indications of congestion. As a result, they differ in the send window they maintain and, consequently, the throughput they achieve. Second, since $W$ is incremented only when ACKs are received, the latency and rate at which ACKs arrive directly impacts throughput—the longer it takes for ACKs to arrive, the slower is the TCP transfer. This implies that TCP throughput depends directly on the path *round-trip times* and *bottleneck transmission capacity*.[1] Third, *packet losses* are used as congestion indicators and result in an aggressive reduction in $W$, and consequently, TCP throughput. Fourth, the *send and receive buffer limits*, and the rate at which the receiving end of an application consumes data, impose a limit on the $W$ and, hence, throughput. Finally, the sender's setting of $S$ impacts the rate at which the send window gets incremented initially, especially for transfers that are short.

It follows that the performance of any TCP session will depend on the exact nature in which it encounters the factors mentioned above. Below, we outline several observations related to the diversity with which the above factors occur in the Internet and use these to derive our modeling objectives.

### 2.3. Modeling objectives

- **Incorporating TCP variants:** TCP congestion control has undergone several enhancements, since it was originally proposed in [29]. While newer versions are getting widely deployed, several different TCP versions may co-exist in the Internet simultaneously [6,7,27]. Furthermore, with the advent of high-speed networks, several researchers are proposing new variants of TCP to enable it to efficiently use network bandwidth [8–11,30]. A TCP model is most useful when it can incorporate several of these variants and, possibly, help in comparing these. This leads to the following modeling objective.

  **Objective 1.** *A TCP model should be applicable to different and newer versions of the protocol.*

- **Incorporating variability:** TCP connections can experience significant statistical variability around average network properties such as round-trip times and packet losses [31,32]. A performance model that estimates only the *expected* transfer time may, therefore, be far off from the *actual* performance experienced by a given TCP connection. It is, therefore, important to also estimate the amount by which the two quantities may differ. The estimation analysis itself, however, should be computationally simple for it to be usable in practice.

  **Objective 2.** *A TCP model should efficiently estimate not just the expected performance of a transfer, but also the amount by which the actual performance may deviate from the expected behavior.*

- **Incorporating network speeds:** The Internet is extremely diverse in the types of edge networking technology used and the end-to-end bottleneck link capacities present. Internet users may sit behind 56 Kbps phone modem lines and engage mostly in text-based email and browsing applications. Other home

---

[1] The bottleneck transmission capacity of a path is defined as the minimum of transmission capacities of all links on the path.

users sitting behind broadband technology, such as ADSL or cable modems, have a bottleneck capacity of a few megabits per second available to them and may engage in the download of large audio and video files. Large organizations and commercial enterprises may have local area networks made of 10/100 Mbps or even Gigabit Ethernet technology. Finally, new networks being deployed for scientific computing with extremely large data sets, have an end-to-end capacity of more than a few gigabits per second [33]. A TCP performance model should not only incorporate the diversity in link technologies present in today's Internet, but also be applicable to future ultra-high-speed networks. This leads to the following modeling objective.

**Objective 3.** *A TCP model should incorporate end-to-end bottleneck capacity and be applicable to paths with different types of edge networking technology—ranging from phone modem lines to high-speed gigabit optical fibers.*

- **Incorporating Transfer Settings:** TCP transfers can be extremely diverse in the size of the transfer [34] as well as the end-host settings for various protocol parameters. For instance, the total number of bytes transmitted in a bulk TCP transfer can vary from as few as 40 B to as much as several megabytes. While short transfers account for a majority of Internet connections, long transfers account for a majority of bytes transferred [35]; hence, it is important to model both types. Furthermore, different operating systems differ in the default initial settings of $S$, the slow-start threshold, and $K$, the maximum limit on congestion window size. This leads to our next modeling objective.

**Objective 4.** *A performance model should be applicable to all types of TCP transfers—independent of the size of transfer and end-host protocol parameter settings.*

A TCP performance model is most useful when it meets all of the above objectives and, thus, caters to the diversity inherent in the Internet.

### 2.4. State of the art

Many analytical models for TCP have been proposed over the last decade [13–25,36]. One way to categorize these is based on whether they conduct *steady-state* or *transient* analysis of TCP connections. We discuss only a few of these below.

A simple formulation for the steady-state throughput of a bulk TCP transfer, as a function of round-trip time and loss rate, was initially presented in [18]. More comprehensive steady-state models were subsequently developed in [13,15,19–24]. One distinguishing feature for some of these is the way in which packet losses are modeled—in [21], the authors conduct fluid analysis of TCP window size behavior by modeling the arrival of packet loss signals as a Poisson process. The model proposed in [13] allows for the incorporation of general and correlated distributions for losses. Perhaps the most prominent steady-state throughput model for TCP was presented in [23], in which the authors modeled the TCP Congestion Avoidance phase using a Markovian model and correlated losses. This work incorporated the impact of retransmission timeouts, fast retransmit, and delayed acknowledgments on TCP throughput. The model was validated by comparing with the actual throughput achieved by several TCP connections instantiated across the Internet. There have also been recent attempts at developing generalized steady-state models that incorporate two or more variants of TCP [14,24,25].

The category of transient TCP analysis has seen less work [16,26,36]. The model in [16] has received much attention, in which the authors extended TCP modeling to transient analysis of the initial Slow Start phase in order to accurately derive the transfer time for short-lived connections. For long transfers, the model incorporated the steady-state throughput formulation from [23] for estimating the remaining transfer time for connections that entered Congestion Avoidance. The authors demonstrated that their formulation was more accurate than past work for not only short transfers, but also when the packet loss rates were very low. It may, therefore, be fair to say that the model in [16] is among the most comprehensive TCP models that exist today.

While many recent models have been well-validated in several realistic scenarios, none of them satisfy simultaneously *all* of the objectives derived in Section 2.3. In particular, several past models mostly incorporate the congestion-control mechanisms in TCP-Reno and are not directly applicable to other TCP versions (Objective 1). Second, to the best of our knowledge, *all* past models predict only the average-case performance for a given TCP transfer—they do not estimate the variability in transfer times (Objective 2). Thus, past models fail to cater to the random dynamics in Internet traffic conditions. Third, *none* of the past models incorporate the impact of bottleneck transmission capacities on TCP performance and hence, are accurate for only limited types of edge-networking technologies (Objective 3). Finally, the analyses in [13–15,17–25] model only the steady-state behavior of long-lived TCP connections and hence, are not applicable to the majority of Internet connections (Objective 4). Furthermore, transient models that switch to steady-state formulations for long transfers, do not adequately deal with the issue of when to switch. We substantiate several of the above observations about past work with analysis and simulations in Sections 5 and 6.

### 2.5. Our approach

In order to simultaneously achieve all of our modeling objectives, we use the following key ideas:

- We incorporate protocol diversity by first defining an abstract class of *Loss-based Congestion Control* (LB-CC) mechanisms. Our definition is fairly general and we show that several TCP variants, including those being proposed for high-speed networks, belong to the LB-CC class. We then develop a parameterized stochastic model that predicts, for any member of the LB-CC class, the time it would take to transfer (TTT) a given number of bytes on a given Internet path.
- We conduct transient analysis of TCP—that does not assume stationarity in TCP window size dynamics—and that predicts the transfer time for a TCP connection accurately, irrespective of the transfer size. Our transient analysis is compute-intensive for long transfers—we enhance it with a simple mechanism for detecting steady-state and switching to a steady-state estimation of transfer time. Our resulting model is both computationally efficient as well as accurate for long and short transfers.
- We estimate the deviation of actual performance from the predicted performance by modeling both the expected value as well as the standard deviation in TTT. Our formulation of TCP window dynamics as a semi-Markov process is fundamental to our ability to compute these performance metrics.
- We explicitly model the impact of the bottleneck transmission capacity on the minimum spacing between ACKs. This allows us to pace TCP "ack-clocking" and, consequently, its throughput.

In what follows, we define the LB-CC class in Section 3 and present our models in Section 4.

## 3. The LB-CC class

Two requirements guide our definition of an abstract framework for different TCP variants. First, the definition should capture as many details of mechanisms *common* to different versions as possible. This will allow analysis conducted using the framework to be accurate. Second, the definition should be generic enough so that it allows the incorporation of the differences between current TCP versions, as well as many new protocol variants.

Recall from the discussion in Section 2.1 that many TCP variants share mechanisms such as use of Fast Retransmit, a flow-control limit, and the Slow Start phase. They may, however, differ in techniques used to detect congestion (segment loss or increase in delays), as well as the window updating functions. In this paper, we focus on TCP versions that use only segment losses to detect congestion. We generalize the window-updating functions of all such protocols by defining the LB-CC class below. In the rest of this paper, we denote the slow-start threshold by $S$, the flow-control limit on window size by $K$, and the send window by $W$.

**Definition 1.** A transport protocol is said to belong to the class of loss-based congestion control (LB-CC) protocols, if the sender employs the following policies for updating its send window and slow-start threshold:

- On receiving the acknowledgment for the successful transmission of a segment, the sender updates its send window as follows:

$$W = \begin{cases} \min\{W + f_1(W), K\}, & \text{if } W < S \\ \min\{W + f_2(W), K\}, & \text{if } W \geqslant S \end{cases} \quad (1)$$

where $(W + f_i(W))$, for $i = 1, 2$, are non-decreasing functions of $W$. $S$ does not get updated on the receipt of an ACK for a successful transmission.
- On receiving the indication of a packet loss through multiple duplicate acknowledgments, the sender reduces $W$ and $S$ as follows:

$$W = \begin{cases} \max\{W - g_1(W), 1\}, & \text{if } W < S \\ \max\{W - g_2(W), 1\}, & \text{if } W \geqslant S \end{cases} \quad (2)$$

$$S = \begin{cases} \max\{W - g_3(W), 1\}, & \text{if } W < S \\ \max\{W - g_4(W), 1\}, & \text{if } W \geqslant S \end{cases} \quad (3)$$

where $g_i(W)$ are positive functions such that for all $W$, $g_3(W) \geqslant g_1(W)$ and $g_4(W) \geqslant g_2(W)$, and $(W - g_i(W))$ are non-increasing functions of $W$, for $i = 1, 2, 3, 4$.
- On receiving the indication of a packet loss through retransmission timeouts, the sender reduces $W$ and $S$ as follows:

$$W = \begin{cases} \max\{W - h_1(W), 1\}, & \text{if } W < S \\ \max\{W - h_2(W), 1\}, & \text{if } W \geqslant S \end{cases} \quad (4)$$

$$S = \begin{cases} \max\{W - h_3(W), 1\}, & \text{if } W < S \\ \max\{W - h_4(W), 1\}, & \text{if } W \geqslant S \end{cases} \quad (5)$$

where $h_i(W)$ are positive functions such that for all $W$, $h_3(W) \geqslant h_1(W)$ and $h_4(W) \geqslant h_2(W)$, and $(W - h_i(W))$ are non-increasing functions of $W$, for $i = 1, 2, 3, 4$.

Observe that our definition of the LB-CC class is quite generic and can incorporate many new protocol designs in addition to those existing today. Below, we illustrate how several TCP variants map to the LB-CC class.

### 3.1. TCP-Reno

Reno senders alternate between the two stages—Slow Start and Congestion Avoidance—of congestion control [27]. Reno senders employ a *multiplicative-increase multiplicative-decrease* (MIMD) window updating policy during Slow Start, and an *additive-increase multiplicative-decrease* policy during Congestion Avoidance [37]. Specifically, when segment losses are detected using triple duplicate ACKs, Reno senders use Fast Recovery to effectively reduce their window size by half. Reno maps to the LB-CC class with the following parameters:

$$f_1(W) = 1; \quad f_2(W) = \frac{1}{W}$$
$$g_1(W) = g_2(W) = \frac{W}{2}$$
$$h_1(W) = h_2(W) = W - 1$$
$$g_3(W) = g_4(W) = h_3(W) = h_4(W) = \frac{W}{2}$$

### 3.2. TCP-Tahoe

The design of TCP-Tahoe [29] predates that of Reno. Reno employs an *additive-increase* policy during Slow Start and a *multiplicative-increase* policy during Congestion Avoidance. In response to packet losses detected by duplicate ACKs as well as retransmission timeouts, Tahoe senders reduce their window size to 1 segment. Tahoe uses Fast Retransmit, but not Fast Recovery. Tahoe, therefore, maps to the LB-CC class with the following parameters:

$$f_1(W) = 1; \quad f_2(W) = \frac{1}{W}$$
$$g_1(W) = g_2(W) = h_1(W) = h_2(W) = W - 1$$
$$g_3(W) = g_4(W) = h_3(W) = h_4(W) = \frac{W}{2}$$

### 3.3. Scalable TCP (S-TCP)

Scalable TCP proposes to achieve high utilization in high-speed networks by adding an MIMD window update region (when the congestion window is above a threshold, $L$) to the Reno SS and CA phases [10]. It maps to the LB-CC class with the following parameters:

$$f_1(W) = 1$$
$$g_1(W) = \frac{W}{2}$$
$$h_1(W) = h_2(W) = W - 1$$
$$g_3(W) = h_3(W) = \frac{W}{2}$$
$$g_4(W) = h_4(W) = \min\left\{\frac{W}{2}, L\right\}$$
$$f_2(W) = \begin{cases} \frac{1}{W}, & \text{if } S \leqslant W < L \\ 0.01, & \text{if } W \geqslant L \end{cases}$$
$$g_2(W) = \begin{cases} \frac{W}{2}, & \text{if } S \leqslant W < L \\ 0.875W, & \text{if } W \geqslant L \end{cases}$$

### 3.4. High-speed TCP (HSTCP)

High-speed TCP is a generalized form of Scalable TCP, in which the MIMD increment and decrement functions are parameterized as follows [28]:

$$f_2(W) = \begin{cases} \frac{1}{W}, & \text{if } S \leqslant W < L \\ \gamma(W), & \text{if } W \geqslant L \end{cases}$$

$$g_2(W) = \begin{cases} \frac{W}{2}, & \text{if } S \leqslant W < L \\ \lambda(W)W, & \text{if } W \geqslant L \end{cases}$$

where $\gamma$ and $\lambda$ are functions of the current window size, the loss probability, as well as several parameters. In real implementations, it is proposed that these functions be looked up from a pre-computed table. A recommended set of parameters is specified in [28], which yields the following forms for these functions:

$$\lambda(W) = 0.5 + 0.4 \frac{log(W) - log(L)}{log(W_{high}) - log(L)}$$

$$\gamma(W) = 2Wp(W) \frac{1 - \lambda(W)}{1 + \lambda(W)}$$

where $p(W) = 0.078/W^{1.2}$ and $W_{high} = 83,000$. We use the above forms in our evaluations in Section 7.

### 3.5. Square root fair (SRF) TCP

SRF TCP is also designed for achieving high-utilization in high-speed networks, and additionally aims to achieve good TCP-friend-liness and minimize RTT-unfairness [12]. Specifically, it advocates a square-root function for window-dynamics in the high-speed region and maps to the LB-CC class with the following parameters:

$$f_1(W) = 1$$

$$g_1(W) = \frac{W}{2}$$

$$h_1(W) = h_2(W) = W - 1$$

$$g_3(W) = h_3(W) = \frac{W}{2}$$

$$g_4(W) = h_4(W) = \min\left\{\frac{W}{2}, L\right\}$$

$$f_2(W) = \begin{cases} \frac{1}{W}, & \text{if } S \leqslant W < L \\ aW^{\alpha}, & \text{if } W \geqslant L \end{cases}$$

$$g_2(W) = \begin{cases} \frac{W}{2}, & \text{if } S \leqslant W < L \\ bW^{\beta}, & \text{if } W \geqslant L \end{cases}$$

where $\alpha = -\frac{1}{2}$, $\beta = \frac{1}{2}$, and $a$, $b$ are positive constants. The recommended values for these are $a = 1.25$ and $b = 15$ [12].

## 4. A transient model for the LB-CC class

In this section, we first formulate the behavior of an LB-CC sender as a semi-Markov process, and then compute performance metrics—specifically, the mean and variance in time to transfer $n$ bytes in a bulk transfer. Below, we describe each of these steps in detail.

### 4.1. Formulating a semi-Markov process

We assume that in a bulk transfer, the TCP sender always transmits packets of the same size $B$, which is equal to the *maximum segment size*.

### 4.1.1. State variables

Recall that one of our modeling objectives is to develop a model that is simple to use for deriving TCP properties. One of the first hurdles in achieving this objective is that the state of an LB-CC TCP sender at any time $t$ is represented using two quantities: the send window $W(t)$ and the Slow Start threshold $S(t)$ (see Definition 1). Keeping track of two variables is significantly more complex

than keeping track of only one variable. The challenge then is: *how can we reduce the complexity of tracking two variables?*

We meet this challenge by exploiting the fact that the state variable $S(t)$ is used only when $W(t) < S(t)$. Furthermore, $S(t)$ is updated to a function of only $W(t)$, and one which does not depend on past values of $S(t)$. These two facts collectively imply that we need not keep track of $S(t)$ for any time instants $t$ at which $W(t) \geqslant S(t)$. This results in a significant gain in efficiency, since the only situations in which $S(t)$ needs to be modeled is either at the beginning of a session, or after the occurrence of a timeout.[2] We rely on standard TCP terminology below, in which the TCP session is said to be in *slow-start* phase when $W(t) < S(t)$, and in *congestion avoidance* phase when $W(t) \geqslant S(t)$.

### 4.1.2. Modeling discrete state updates

Since TCP is an event-driven protocol, state updates can be modeled using discrete time steps. Most past models do so by using the approximation that state variables get updated once every *flight*, where a flight is typically defined as the time interval between the transmission of the first packet of the current window and the receipt of its acknowledgment. The flight duration is approximated by the *mean* RTT of the path between the TCP sender and receiver.

In practice, however, TCP senders do not use the notion of flights, but update their state on the receipt of *every* ACK or on detecting packet losses or time-outs. We accurately model such behavior as described below.

Let $U_n$ be the time when the $n$th acknowledgment is received. Let $W_n = W(U_n-)$ be the window-size of the session just before the $n$th acknowledgment is received. Similarly, define $S_n = S(U_n-)$. We assume window-size dependent packet losses: we denote $p_L(W_n)$ as the probability for a sender to receive a third duplicate ACK in the $n$th ACK event, and $p_T(W_n)$ as the probability for a timeout to follow immediately after the $n$th ACK is received. Thus, if $p_L$ and $p_T$ do not depend on the window-size, the model reduces to independent and identically distributed packet losses.

Note that these probabilities do not depend on the value of $S$. To summarize, we assume that the $n$th acknowledgment indicates a packet loss via multiple duplicate ACKs with probability $p_L(W_n)$, a packet loss via timeout with probability $p_T(W_n)$ and a successful transmission with probability $1 - p(W_n)$, where $p(W_n) = p_L(W_n) + p_T(W_n)$.

With the above assumptions and notation, we can model the $\{(W_n, S_n), n \geqslant 0\}$ process as a *Discrete-time Markov chain*. Recall that we do not need to keep track of $S_n$ during the congestion avoidance phase. We exploit this fact by setting $S_n = W_n$ as long as the process stays in the congestion avoidance phase.

For brevity, we use the following notation:

$$\bar{f}_i(w) = \min\{w + f_i(w), K\}, \quad i = 1, 2,$$
$$\underline{g}_i(w) = \max\{w - g_i(w), 1\}, \quad i = 1, \dots, 4,$$
$$\underline{h}_i(w) = \max\{w - h_i(w), 1\}, \quad i = 1, \dots, 4.$$

Clearly $\bar{f}_i, \underline{g}_i$, and $\underline{h}_i$, are bounded below by 1 and above by $K$. We further assume that $\underline{g}_i$ and $\underline{h}_i$ are bounded above by $L \leqslant K$.[3] Thus the state space of the $\{(W_n, S_n), n \geqslant 0\}$ process is $\{(w, s) : 1 \leqslant w < s \leqslant L\} \cup \{(w, s) : 1 \leqslant w = s \leqslant K\}$. This state space grows as the square of $L$, but only linearly in $K$. This allows computing efficiencies in

---

[2] It is important to note that although in practice, $S(t)$ gets updated on detecting losses through multiple duplicate ACKs, we need not model it. This is because $W(t)$ and $S(t)$ are set to the *same* value in response to such events. Since $W(t)$ is not less than $S(t)$, therefore, we need not track the latter quantity.

[3] This formulation fits in nicely both with current protocols, for which $L = K$, as well as for high-speed protocols, for which $L$ is the low threshold [28,10].

the analysis of connections with large $K$, especially in high-speed networks.

Using the above framework, the transition equations for the $\{(W_n, S_n), n \geqslant 0\}$ process are given below.

*Congestion avoidance (CA) phase:* For $1 \leqslant w \leqslant K$

$$(W_n = w, S_n = w) \rightarrow (W_{n+1}, S_{n+1}) = \begin{cases} (\bar{f}_2(w), \bar{f}_2(w)), & \text{w. p. } 1 - p(w) \\ (\underline{g}_2(w), \underline{g}_2(w)), & \text{w. p. } p_L(w) \\ (\underline{h}_2(w), \underline{h}_4(w)), & \text{w. p. } p_T(w) \end{cases}$$

(6)

*Slow-start (SS) phase:* For $1 \leqslant w < s \leqslant L$

$$(W_n = w, S_n = s) \rightarrow (W_{n+1}, S_{n+1})$$
$$= \begin{cases} (\bar{f}_1(w), s), & \text{w. p. } 1 - p(w), \text{ if } \bar{f}_1(w) < s \\ (\bar{f}_1(w), \bar{f}_1(w)), & \text{w. p. } 1 - p(w), \text{ if } \bar{f}_1(w) \geqslant s \\ (\underline{g}_1(w), \underline{g}_1(w)), & \text{w. p. } p_L(w) \\ (\underline{h}_1(w), \underline{h}_3(w)), & \text{w. p. } p_T(w) \end{cases}$$

(7)

Note that the functions $g_3$ and $g_4$ do not play any role in these transition equations. This implies that we can restrict our attention to LB-CC protocols with $g_3 = g_1$ and $g_4 = g_2$.

### 4.1.3. Modeling the time between updates

In order to compute time-related performance metrics such as the transfer time of a session, we also need to model $U_n$, or more specifically, the time between the receipt of acknowledgments: $U_{n+1} - U_n$. If we assume that acknowledgments are uniformly distributed within a flight, then the time between acknowledgments at time $t$ can be approximated as: $RTT/W(t)$, where $RTT$ is the mean RTT. Indeed, this is precisely what is done in past work, where the instantaneous throughput is modeled as the inverse of this quantity, namely $W(t)/RTT$. Unfortunately, this formulation ignores the impact of bottleneck transmission capacity on the spacing between ACKs. In particular, if the minimum transmission capacity among all links on the paths between the sender and receiver is $C$, then the segments will be spaced on an average at least $B/C$ time units apart when delivered to the receiver, where $B$ is the segment size. Consequently, ACKs received at the sender will also have the same minimum spacing. Another way to describe this behavior is that when the window size grows beyond the *delay-bandwidth product* ($RTT * C/B$) of the path, the bottleneck transmission capacity of the path will limit TCP throughput.

The time between two acknowledgments is, therefore, estimated by:

$$\alpha(W_n) = \max\{RTT/W_n, B/C\}$$

Let $t_{TO}$ be the average timeout duration for the TCP session. We incorporate the impact of timeouts by assuming that if the $n$th acknowledgment indicates a timeout, the next acknowledgment is delayed by an additional time $t_{TO}$. This is a crude but satisfactory method of accounting for the fact that no new segments are transmitted during a timeout. Also, during Fast Retransmit, the sender retransmits the lost segment, before sending out new packets at its reduced window. Thus, we see that:

$$U_{n+1} - U_n = \begin{cases} \alpha(W_n) & \text{w. p. } 1 - p(W_n) \\ \alpha(W_n) + RTT & \text{w. p. } p_L(W_n) \\ \alpha(W_n) + t_{TO} & \text{w. p. } p_T(W_n) \end{cases}$$

(8)

With this formulation we see that $\{(W(t), S(t)), t \geqslant 0\}$ is a semi-Markov process. This probabilistic structure allows us to compute many desired performance measures in an easy fashion. In particular, we are interested in $T_n$, the time to transfer $n$ segments successfully. Below, we compute the mean and variance of $T_n$ for an LB-CC TCP session.

### 4.2. Computing mean transfer time

Define, for the CA and SS phases, respectively:

$$\tau_n(w) = E(T_n|W(0) = w, S(0) = w), \quad 1 \leqslant w \leqslant K$$
$$\tau_n(w, s) = E(T_n|W(0) = w, S(0) = s), \quad 1 \leqslant w < s \leqslant L$$

Note that the expected transfer time while in CA mode, $\tau_n(w)$, does not depend upon the value of the $S$. This greatly simplifies the computation. Also note that since the TCP session starts in state $(W(0) = 1, S(0) = L)$, the time to send $n$ packets successfully is given by $\tau_n(1, L)$.

Now let $u(w)$ be the expected time until the next acknowledgment, given that current window is $w$.

From Eq. (8), we get

$$u(w) = \alpha(w) + t_{TO}p_T(w) + RTT p_L(w), \quad 1 \leqslant w \leqslant K.$$

Now condition on the time to receive this acknowledgment. It takes an expected amount given by $u(w)$. When it arrives, the state of the session changes according to Eqs. (6) and (7). If the acknowledgment indicates a success, we need to transmit $n - 1$ more packets; else we need to transmit $n$ more packets. Putting all these events together, we get the following equations:

$$\tau_n(w) = u(w) + (1 - p(w))\tau_{n-1}(\bar{f}_2(w)) + p_T(w)\tau_n(\underline{h}_2(w), \underline{h}_4(w)) + p_L(w)\tau_n(\underline{g}_2(w)), \quad 1 \leqslant w \leqslant K$$

(9)

$$\tau_n(w, s) = u(w) + (1 - p(w))\tau_{n-1}(\bar{f}_1(w), s) + p_T(w)\tau_n(\underline{h}_1(w), \underline{h}_3(w)) + p_L(w)\tau_n(\underline{g}_1(w)), 1 \leqslant \bar{f}_1(w) < s \leqslant L$$

(10)

$$\tau_n(w, s) = u(w) + (1 - p(w))\tau_{n-1}(s) + p_T(w)\tau_n(\underline{h}_1(w), \underline{h}_3(w)) + p_L(w)\tau_n(\underline{g}_1(w)), 1 \leqslant \bar{f}_1(w) = s \leqslant L$$

(11)

We have the following initial conditions:

$$\tau_0(w) = 0, \quad \tau_0(w, s) = 0.$$

Since $\tau_n$ appears on both sides of Eqs. (9)–(11), we need an efficient method of computing the above quantities. One such method is to use iterations, which we explain for Eq. (9). Assume that $\tau_{n-1}(w)$ is known for all $1 \leqslant w \leqslant K$. Let $\tau_{n,0}(w) = 0$ for all $1 \leqslant w \leqslant K$ and compute

$$\tau_{n,k+1}(w) = u(w) + (1 - p(w))\tau_{n-1}(\bar{f}_2(w)) + p_T(w)\tau_{n,k}(\underline{h}_2(w), \underline{h}_4(w)) + p_L(w)\tau_{n,k}(\underline{g}_2(w)), \quad 1 \leqslant w \leqslant K$$

(12)

It is easy to see that the above iteration is a contraction mapping and as $k \rightarrow \infty, \tau_{n,k}(w)$ approaches $\tau_n(w)$ geometrically at the rate $\max(p(w))$. This is a very rapid convergence, especially when the loss probabilities are small. Finally, we can recursively obtain $\tau_n$ starting with the initial condition $\tau_0 = 0$.

### 4.3. Computing variance in transfer time

Next we derive the second moment of $T_n$. Define

$$\sigma_n(w) = E(T_n^2|W(0) = w, S(0) = w), \quad 1 \leqslant w \leqslant K$$
$$\sigma_n(w, s) = E(T_n^2|W(0) = w, S(0) = s), \quad 1 \leqslant s < w \leqslant L$$

The variance of the time to send $n$ packets is then given by

$$V(n) = \sigma_n(1, L) - (\tau_n(1, L))^2$$

By doing the same type of first step analysis as for the first moment, we get

$$
\begin{aligned}
\sigma_n(w) &= E[T_n^2|W(0) = w, S(0) = w] \\
&= (1 - p(w))E[(\alpha(w) + T_{n-1})^2|W(0) = \bar{f}_2(w), S(0) \\
&= \bar{f}_2(w)] + p_L(w)E[(\alpha(w) + RTT + T_n)^2|W(0) \\
&= \underline{g}_2(w), S(0) \\
&= \underline{g}_2(w)] + p_T(w)E[(\alpha(w) + t_{TO} + T_n)^2|W(0) \\
&= \underline{h}_2(w), S(0) = \underline{h}_4(w)] \, 1 \leqslant w \leqslant K
\end{aligned}
\tag{13}
$$

$$
\begin{aligned}
\sigma_n(w,s) &= E[T_n^2|W(0) = w, S(0) = s] \\
&= (1 - p(w))E[(\alpha(w) + T_{n-1})^2|W(0) = \bar{f}_1(w), S(0) \\
&= s] + p_L(w)E[(\alpha(w) + RTT + T_n)^2|W(0) \\
&= \underline{g}_1(w), S(0) \\
&= \underline{g}_1(w)] + p_T(w)E[(\alpha(w) + t_{TO} + T_n)^2|W(0) \\
&= \underline{h}_1(w), S(0) = \underline{h}_3(w)], \, 1 \leqslant \bar{f}_1(w) < s \leqslant L
\end{aligned}
\tag{14}
$$

$$
\begin{aligned}
\sigma_n(w,s) &= E[T_n^2|W(0) = w, S(0) = s] \\
&= (1 - p(w))E[(\alpha(w) + T_{n-1})^2|W(0) = s, S(0) \\
&= s] + p_L(w)E[(\alpha(w) + RTT + T_n)^2|W(0) \\
&= \underline{g}_1(w), S(0) \\
&= \underline{g}_1(w)] + p_T(w)E[(\alpha(w) + t_{TO} + T_n)^2|W(0) \\
&= \underline{h}_1(w), S(0) = \underline{h}_3(w)] \, 1 \leqslant \bar{f}_1(w) = s \leqslant L
\end{aligned}
\tag{15}
$$

After tedious algebra and using Eqs. (9)–(11), and using the notation

$$
v(w) = RTT^2 p_L(w) + t_{TO}^2 p_T(w) - \alpha(w)^2, \quad 1 \leqslant w \leqslant K,
$$

the above equations reduce to

$$
\begin{aligned}
\sigma_n(w) &= v(w) + 2\alpha(w)\tau_n(w) + 2t_{TO}p_T(w)\tau_n(\underline{h}_2(w), \underline{h}_4(w)) \\
&\quad + 2RTTp_L(w)\tau_n(\underline{g}_2(w)) + (1 - p(w))\sigma_{n-1}(\bar{f}_2(w)) \\
&\quad + p_L(w)\sigma_n(\underline{g}_2(w)) + p_T(w)\sigma_n(\underline{h}_2(w), \underline{h}_4(w)), 1 \\
&\leqslant w \leqslant K
\end{aligned}
\tag{16}
$$

$$
\begin{aligned}
\sigma_n(w,s) &= v(w) + 2\alpha(w)\tau_n(w,s) \\
&\quad + 2t_{TO}p_T(w)\tau_n(\underline{h}_1(w), \underline{h}_3(w)) \\
&\quad + 2RTTp_L(w)\tau_n(\underline{g}_1(w)) + (1 - p(w))\sigma_{n-1}(\bar{f}_1(w), s) \\
&\quad + p_L(w)\sigma_n(\underline{g}_1(w)) + p_T(w)\sigma_n(\underline{h}_1(w), \underline{h}_3(w)), 1 \\
&\leqslant \bar{f}_1(w) < s \leqslant L
\end{aligned}
\tag{17}
$$

$$
\begin{aligned}
\sigma_n(w,s) &= v(w) + 2\alpha(w)\tau_n(w,s) \\
&\quad + 2t_{TO}p_T(w)\tau_n(\underline{h}_1(w), \underline{h}_3(w)) \\
&\quad + 2RTTp_L(w)\tau_n(\underline{g}_1(w)) + (1 - p(w))\sigma_{n-1}(s) \\
&\quad + p_L(w)\sigma_n(\underline{g}_1(w)) + p_T(w)\sigma_n(\underline{h}_1(w), \underline{h}_3(w)), 1 \\
&\leqslant \bar{f}_1(w) = s \leqslant L
\end{aligned}
\tag{18}
$$

These equations have the same structure as the equations for the mean transfer time, and hence can be solved by the same iterative, recursive fashion.

### 4.4. Modeling $p_T$ and $t_{TO}$

Retransmission timeouts are a TCP-specific mechanism, the design of which impacts $p_T$, the probability of a timeout event, and $t_{TO}$, the average duration of a timeout. In [23], the following formu-

lation is suggested for estimating $p_T$ and $t_{TO}$ from the packet loss probability, $p$:[4]

$$
p_T(w) = \min\left\{1, \frac{(1 - p_i^3)(1 + p_i^3(1 - p_i^{w-3}))}{1 - p_i^w}\right\}
\tag{19}
$$

$$
t_{TO} = TO \frac{1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6}{1 - p}
\tag{20}
$$

where $p_i = 1 - p$ and $TO$ is the value of the average single timeout. The probability of receiving loss indication via multiple duplicate ACKs is then computed as: $p_L(w) = p - p_T(w)$.

The above formulation has been used in other TCP modeling efforts (see for example [16]). Our experimental evaluations in Section 5 indicate, however, that this formulation is inaccurate when packet loss rates are very high ($p$ is greater than 1–3%). In the absence of a more accurate formulation in the literature, we too adopt the above for our experimental validations in Section 5—however, we emphasize that any improved formulation can be directly applied to our model since it does not make any restrictive assumptions about $p_T$. For example, to model TCP SACK (selective acknowledgements), we could replace $p_T$ and $p_L$ with those from [38].

### 4.5. Modeling ECN

It is important to mention that TCP behavior in an ECN-enabled network is not explicitly addressed in this paper. However, extending the LB-CC framework to incorporate ECN would be a fairly straightforward exercise. In the extended framework, (i) on receiving an ECN congestion-signal, the TCP sender would reduce $W$ exactly as in Eqs. (2) and (3); (ii) $p_E(W_n)$ would be the probability of receiving an ECN congestion-signal; and (iii) the analysis would reformulate to include: $p(W_n) = p_L + p_E + p_T$, and $U_{N+1} - U_N = \alpha(W_n)$ w.p. $1 - p(W_n) + p_E(W_n)$.

## 5. Model validation

We have implemented the model presented in Section 4 using the Matlab programming environment [39]. We use the Matlab implementation to validate our model, henceforth referred to as the LB-CC model, in two different settings: (i) validation against simulation of a single TCP connection with carefully-controlled network settings; and (ii) validation against TCP connections simulated using a comprehensive HTTP workload. We use NS-2 for our simulations [40]. We compare the accuracy of our model to the one proposed in [16], henceforth referred to as the "Cardwell" model.

### 5.1. Single-connection simulations

#### 5.1.1. Validation methodology

We validate the ability of our model to accurately capture the impact of five factors—namely, $C$ (bottleneck), $p$ (loss), RTT, $K$ (window size limit), and the protocol version—on the transfer time of a bulk TCP transfer. For each combination of these factors, we run $N_{sim}$, where $N_{sim} \geqslant 100$, simulations of a TCP connection that transfers 1000 segments, each of size 1460 B, over a linear 2-hop path between the sender and receiver. We set all link capacities equal to the desired $C$ and the sum of link propagation latencies to the desired minimum RTT. Note that the actual RTTs will be variable due to buffering at the router. We subject the TCP connection, referred to as $A_1$, to independent random packet losses with the desired probability $p$. Router buffers are well-provisioned to avoid

---

[4] It is assumed in [23] that $p(w)$ is independent of $w$; hence, we denote it simply as $p$ in the formulation.

**Table 1**
Comparison of computed and observed values of $p_T$.

| Figure | Observed $p$ (%) | Computed $p_T$ (%) | Observed $p_T$ (%) |
|---|---|---|---|
| 1 | 0.994 | 0.059 | 0.983 |
| 2 | 0.994 | 0.059 | 0.077 |
| 3 | 0.989 | 0.058 | 0.063 |
| 4 | 0.991 | 0.058 | 0.186 |
| 5 | 0.093 | 0.001 | 0.002 |
| 6 | 2.938 | 0.479 | 0.370 |
| 7 | 4.961 | 1.306 | 0.960 |

additional packet drops due to buffer overflow. The maximum window size limit is set to the desired $K$.

At the end of the $N_{sim}$ simulations, we compute the average value of the per-connection $p$ (see Table 1). We then feed this quantity into the LB-CC and Cardwell models and compute $\tau_n$ for both. We also compute the variance $V(n)$ in transfer time using our model LB-CC. We then compare these quantities to quantities $E[T_n]$ and $Var[T_n]$ estimated from the simulations. Unless explicitly mentioned, all validations are conducted using TCP Reno ($f$, $g$, and $h$ functions defined in Section 3).

### 5.1.2. Impact of bottleneck capacities

We simulate five kinds of networking technologies: 56 Kbps (phone modems), 1.54 Mbps (broadband ADSL), 10 Mbps (Ethernet), 54 Mbps (VDSL, 802.11), and 100 Mbps (fast Ethernet). For each kind of network, we subject the single TCP connection $A_1$ to a round-trip propagation latency of 100 ms and a packet loss probability of 0.01 (these choices will be justified later in this section). $K$ is set equal to the delay-bandwidth product.

Figs. 1–3 plot the transfer time metrics as a function of the number of segments transmitted in topologies with capacities of 56 Kbps, 1.54 Mbps, and 10 Mbps. (In all of the figures, the LB-CC model is labeled M1, and the Cardwell model is labeled M2.) The results with 54 Mbps and 100 Mbps were similar to the 10 Mbps experiment and have been omitted due to space constraints. We find that both LB-CC and Cardwell track the average transfer time of connections quite well at high link capacities. At low link capacities, however, Cardwell is unable to track the impact of bottleneck capacity on ACK spacing, and hence under-predicts the expected transfer time.

Recall from the discussion in Section 4 that the impact of small bottleneck capacities on TCP throughput increases when the window grows larger than the delay-bandwidth product. To better illustrate this effect, we re-use the 56 Kbps topology, but simulate a TCP connection with $K = 44$ segments.[5] This window size far exceeds the delay-bandwidth product of the topology, but represents a likely scenario, in which TCP connections use default operating system settings. Fig. 4 plots the results of this experiment. We find that the ability of Cardwell to estimate the transfer time accurately worsens even further in this case. The LB-CC model is, however, able to estimate the transfer time fairly accurately for all settings of $C$ and $K$.

The LB-CC model also tracks the standard deviation in $T_n$ reasonably well. In some cases, though, the deviation in the simulation transfer times are higher—we expect these to reduce if we increase $N_{sim}$. It is interesting to note that the deviation with $K = 44$ (Fig. 4) is much lower for both simulations and the LB-CC model, than in Fig. 1. We believe that this is because with $K = 44$, $A_1$ has a greater likelihood of receiving three duplicate ACKs and, hence, suffers a lower number of timeouts (see Table 1). Timeout events are likely to add significant variability to transfer times.

[5] The maximum window that can be advertised without using extra options is 64$KB$. With 1460-byte segments and no extra options, the window size can be at most 44 segments.
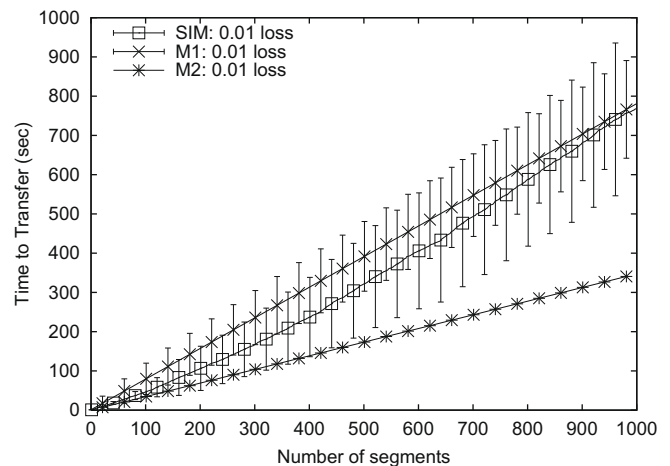


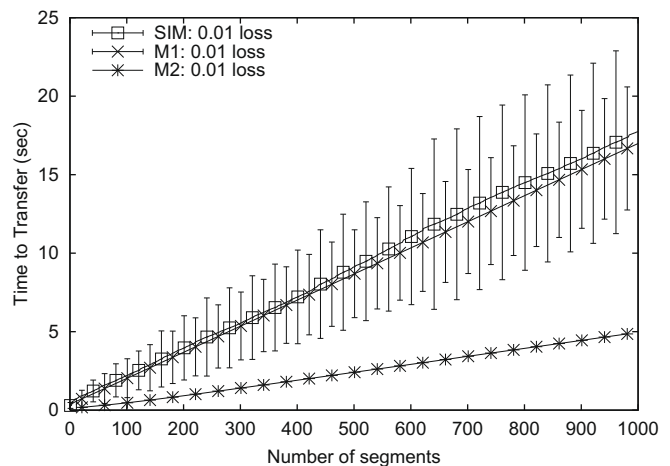**Fig. 1.** 56 Kbps, 100 ms RTT, 0.01 loss.



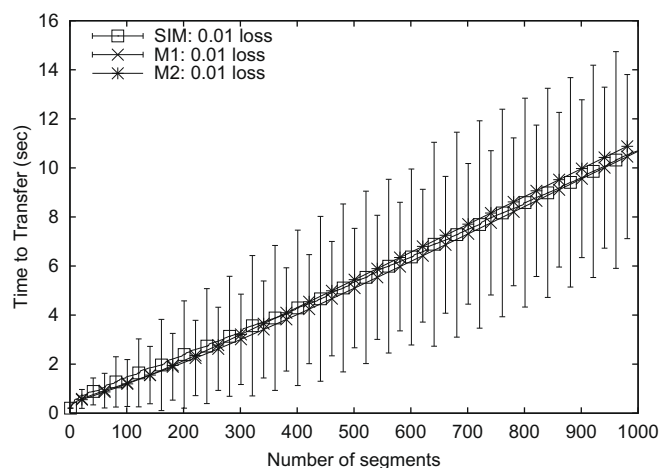**Fig. 2.** 1.54 Mbps, 100 ms RTT, 0.01 loss.



**Fig. 3.** 10 Mbps, 100 ms RTT, 0.01 loss.

In our validation experiments below with different loss rates and RTTs, we restrict our attention to a 10 Mbps topology and set $K$ to the delay-bandwidth product. This helps ensure a fair comparison of LB-CC and Cardwell—using a smaller $C$ or larger $K$ is likely to bias the results against Cardwell.
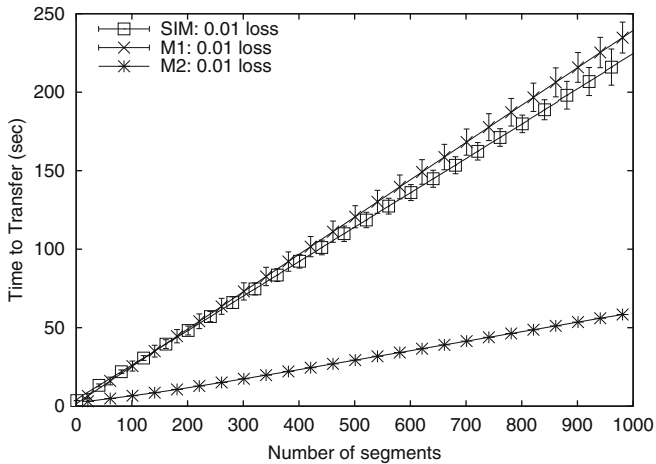
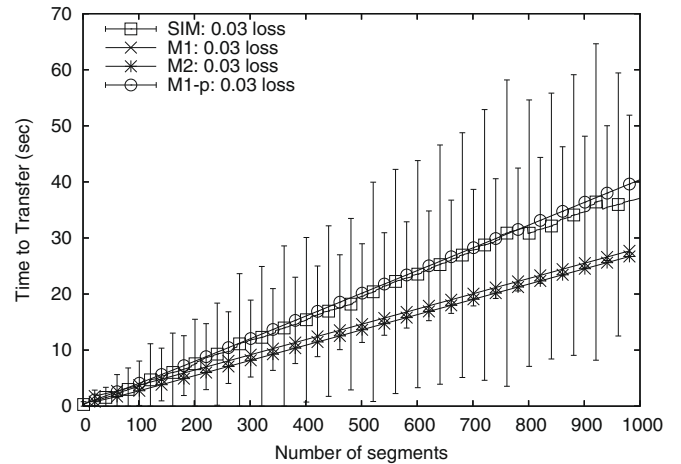**Fig. 4.** 56 Kbps, $K = 44$, 100 ms RTT, 0.01 loss.



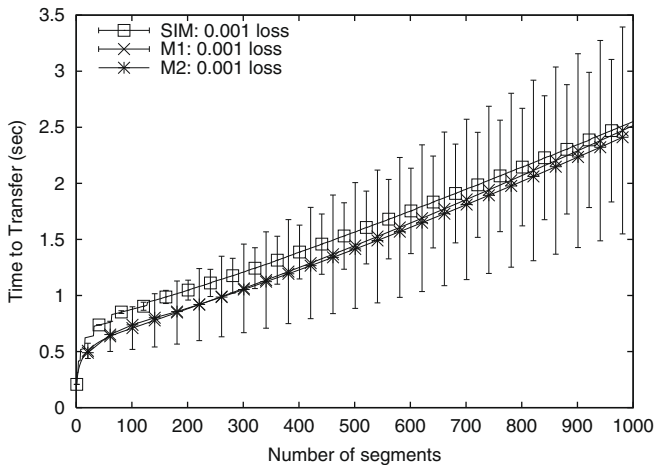**Fig. 6.** 0.03 loss, 10 Mbps, 100 ms RTT.



**Fig. 5.** 0.001 loss, 10 Mbps, 100 ms RTT.

model is fairly accurate even at high loss rates. We emphasize again that this indicates only a need for more accurate modeling of $p_T$ at high loss rates and does not say anything about the relative accuracy of LB-CC and Cardwell at different loss rates.

We find that the LB-CC model tracks the deviation in simulation transfer times well for loss probabilities lower than 0.01. We also find that the end-to-end RTTs do not influence the accuracy of our model, except at the high loss rates noted above. In the remaining validation experiments, we restrict our attention to a loss probability of no more than 0.01.

*5.1.4. Validation with different protocols*

All validations presented so far have been conducted with TCP Reno. We also validate our model for four other LB-CC protocols, namely, Tahoe, Scalable TCP, High-speed TCP, and Square Root Fair TCP. We simulate a Tahoe TCP connection on a 10 Mbps topology— the latter three protocols are also simulated at 100 Mbps and 1 Gbps topologies (since these protocols are designed for high-speed networks). We subject each topology to different loss rates ranging from 0.00001 to 0.01, and RTTs ranging from 10 ms to 200 ms.

We find that the LB-CC model tracks the simulation results quite well in all of these experiments (we omit the plots due to space constraints).
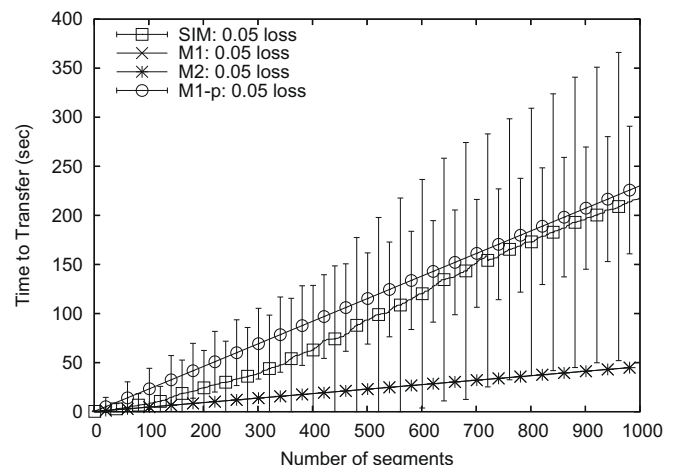
*5.1.3. Impact of loss rates and RTTs*

Internet loss probabilities can range from less than $10^{-6}$ (medium errors) to more than 0.05 (congestion and wireless links). End-to-end RTTs can also vary from a few to hundreds of milliseconds. In order to validate our model under a diverse set of loss rates and RTTs, we simulate a set of 10 Mbps topology with three kinds of end-to-end propagation latencies—10 ms (metropolitan networks), 100 ms (cross-country transfers), and 200 ms (inter-continental transfers). (Note that the actual RTTs will vary due to buffering at the router.) We then run different experiments, in which we subject $A_1$ to different loss probabilities—specifically, 0.0001, 0.001, 0.01, 0.03, and 0.05.

Figs. 3, 5, and 6 plot the transfer metrics for a 100 ms topology with loss probabilities of 0.01, 0.001, and 0.03, respectively. We find that both LB-CC and Cardwell are equally good at modeling loss probabilities of 0.01 or lower in a 10 Mbps topology. At higher loss probabilities, however, both models under-predict TCP transfer time. Table 1, which lists the values of $p_T$ observed in the simulations against those computed using Eq. (19), shows that at high loss rates, the computed values of $p_T$ can be fairly inaccurate. We believe that this inaccuracy is responsible for the under-estimation by both models. In order to validate our conjecture, we use as input to our model the observed values of $p_T$ from Table 1—Fig. 6 also plots the resultant predictions of transfer time (labelled as $M_1 - p$)—we find that an accurate value of $p_T$ ensures that the



**Fig. 7.** 0.05 loss, 10 Mbps, 100 ms RTT.

## 5.2. HTTP workload simulations

The validations conducted in Section 5.1 do not incorporate the impact of competing cross-traffic on the performance of a given TCP connection. In this section, we present validation results from several simulations conducted with an empirically-derived HTTP workload model [41].

### 5.2.1. Experimental methodology

We simulate extensive two-way HTTP traffic workload generated on the topology depicted in Fig. 8. Each circle and hexagon in the figure represents *five* "clouds" of HTTP clients or servers (i.e., end systems sharing an aggregation link). The dashed lines represent the direction of the flow of data traffic in the network. Regular traffic is generated by circles 0 and 5 and traverses all routers. Cross-traffic is generated by circles 1–4 and shares only one link with regular traffic. This topology, first proposed in [42], allows us to simulate end-to-end paths with multiple congested links and different offered loads. In addition to the link propagation delays, the routers have been modified to delay segments by a fixed amount on a per-connection basis—this allows us to simulate TCP connections with different minimum RTTs and, thus, represent large networks.

We use the PackMime model [41] to generate synthetic web traffic. We also use PackMime to generate an empirical minimum RTT distribution. We run several experiments in which we simulate 1–3 bottleneck links and generate HTTP workloads ranging from 50% to 90% of the bottleneck capacity. These experiments help us sample a very diverse set of TCP connections, with loss probabilities ranging from 0.0008 to more than 0.04 and round-trip times ranging from 60 ms to 400 ms.

### 5.2.2. Validation

We consider all TCP connections simulated as regular traffic above and for each, record $n$ (total number of segments transferred) and $T_n$ (transfer time) and compute the values of $p$, $p_T$, and mean $t_{TO}$. We then feed these quantities to the LB-CC model and compute the predicted transfer time, $T_n^{LB-CC}$, for each connection. Fig. 9 plots the cumulative distribution of the relative errors—computed as $\left( \frac{T_n - T_n^{LB-CC}}{T_n} \right)$. We find that our model tracks $T_n$ reasonably well. For instance, the prediction accuracy is within 0.1 of the simulations for 80% of the connections.
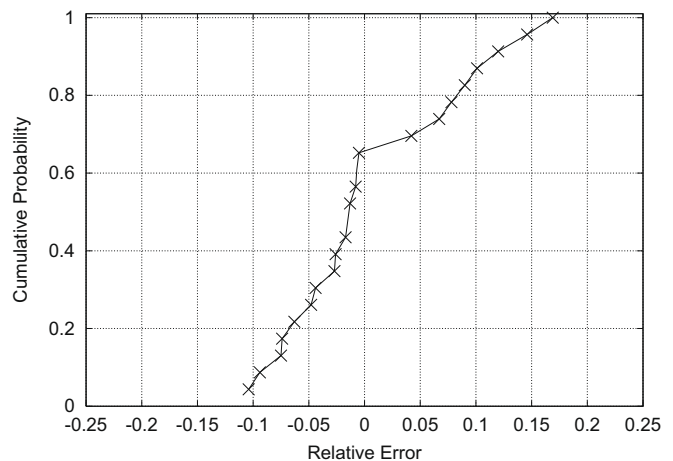


**Fig. 9.** Relative error with HTTP simulations.

## 6. Achieving computational efficiency

One limitation of our iterative model is that the complexity of computing $\tau_n$ is linear in $n$, the number of segments to be transferred. This is not the case with most past TCP models, since they rely on *steady-state* analysis for computing the (constant) mean TCP throughput for long transfers. However, a TCP transfer may transmit quite a few segments before it attains steady-state throughput—this is especially true in high-speed networks. For accurately modeling short transfers, therefore, it is important to conduct *transient* analysis of the kind presented in Section 4. In order to achieve simultaneously modeling accuracy as well as computational efficiency, we use the approach of: *(i) detecting when a TCP transfer has attained steady-state, and (ii) using steady-state throughput to predict its remaining transfer time.* The basic idea here has been used even in past work—indeed, the model in [16] switches to a steady-state prediction model as soon as the sender leaves the initial Slow Start phase. The key difference, however, is in deciding when to switch.

Our approach is based on the following key insight. The slope of the $\tau_n$ curve converges as $n$ increases; Fig. 11, which plots the instantaneous throughput ($B/\tau_n - \tau_{n-1}$) for the $M_1$ curve in Fig. 3,
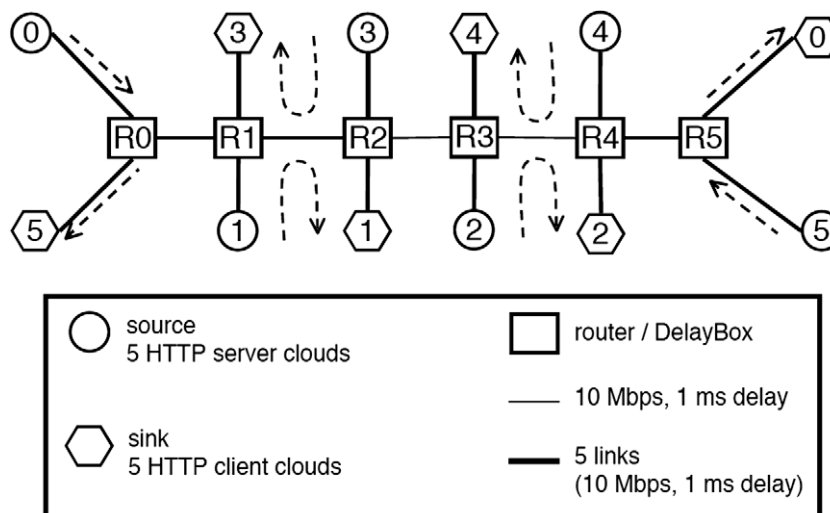


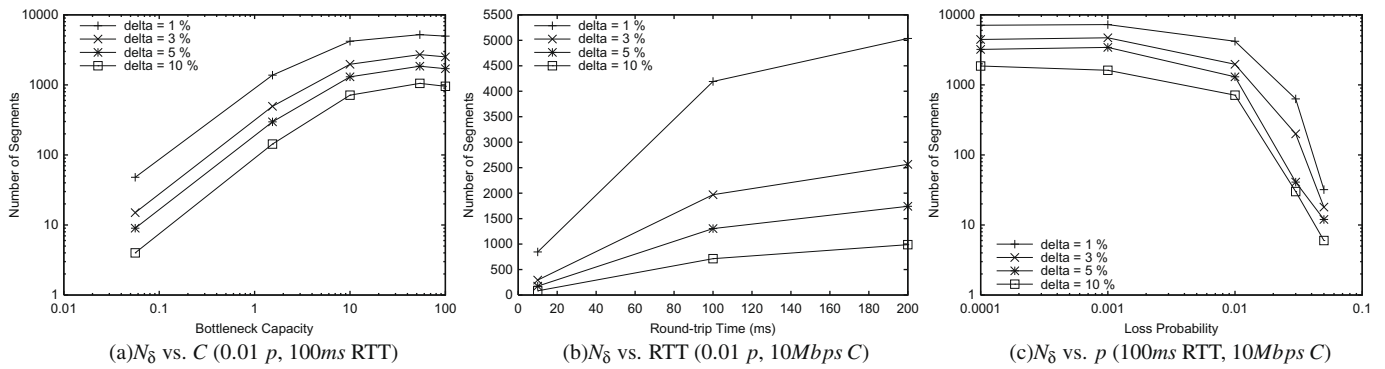**Fig. 8.** HTTP workload simulations.

(a)$N_\delta$ vs. $C$ (0.01 $p$, 100$ms$ RTT)

(b)$N_\delta$ vs. RTT (0.01 $p$, 10$Mbps$ C)

(c)$N_\delta$ vs. $p$ (100$ms$ RTT, 10$Mbps$ C)

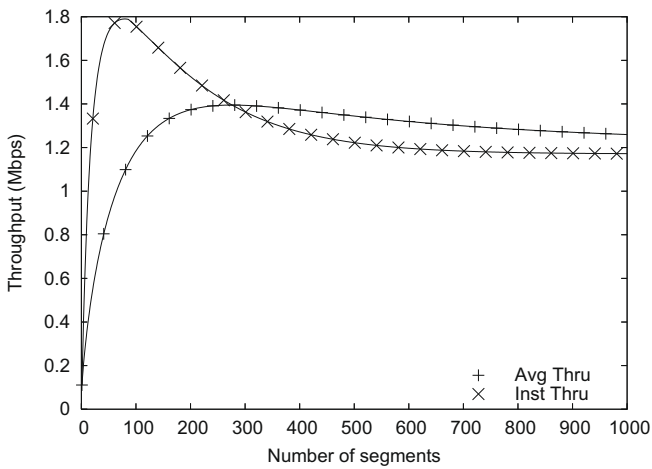**Fig. 10.** $N_\delta$ as a function of $C$, RTT, and $p$.



**Fig. 11.** Throughput.

illustrates this. This implies that after some value of $n$, $\tau_n$ can be approximated by a linear function of $n$. We use this insight in improving the efficiency of our model as follows. For each value of $n$, we compute the slope of the transfer time curve as: $\psi_n = \tau_n - \tau_{n-1}$. The above observation on convergence of the slope implies that $\psi_n - \psi_{n-1}$ converges to 0. Let $N_\epsilon$ denote the smallest value of $n$ such that $\psi_n - \psi_{n-1} < \epsilon$. Then, for any specifiable $\epsilon$, we approximate the curve for all values of $n > N_\epsilon$ by a straight line of slope $\psi_{N_\epsilon}$, such that the line passes through the point $(N_\epsilon, \tau_{N_\epsilon})$. Using this idea, we have reduced the complexity of computing $\tau_n$ to a tunable value: $O(min(n, N_\epsilon))$. In fact, the graphs plotted in Section 5 have been computed using $\epsilon = 10^{-12}$.

### 6.1. When is steady-state-only analysis usable?

Observe that models that rely only on steady-state analysis of TCP work fairly well for bulk transfers that are long. A natural question to ask is: *how long does a TCP transfer have to be before a steady-state-only analysis can accurately predict its transfer time?*

Fig. 11 also plots the average throughput, $\Theta_n = nB/\tau_n$, as a function of $n$ for the experiment depicted in Fig. 3 (10 Mbps $C$, 100 ms RTT, 0.01 $p$). $\Theta_n$ converges to the steady-state throughput, $\Theta_\infty$, as $n$ increases. It follows that steady-state analysis can be used for all connections large enough, such that $\Theta_n$ is reasonable close to $\Theta_\infty$. With this understanding, and using $\Theta_{10000}$ as a reasonable approximation of $\Theta_\infty$, we answer the question raised above as follows.

For any given $\delta$, we find the smallest value of $n$—denoted by $N_\delta$—such that $\Theta_i - \Theta_\infty < \delta$, for all $i \geqslant N_\delta$. In Fig. 10(a)–(c), we plot $N_\delta$ as a function of $C$, RTT, and $p$, respectively. In each figure, we plot curves for $\delta = 0.01, 0.03, 0.05, 0.1$. We find that $N_\delta$ increases with $C$ and RTT. To put this observation in the proper perspective, recall that we set $K$ equal to the bandwidth-delay product for all of these experiments. Thus, $K$ is higher for topologies with larger $C$ and $RTT$, and it is expected that a connection will take longer to grow up to a window size of $K$. $N_\delta$ decreases as $p$ increases. We expect this to be the case because the steady-state average window size is likely to be lower at high loss rates, and hence, is attained faster.

Perhaps the most surprising observation is that $N_\delta$ can be as large as several thousands of segments, even for $\delta = 0.1$. This implies that models that rely only on steady-state analysis are likely to be accurate only for transfers larger than several megabytes. Internet traffic analysis in [35] shows that such transfers may account for less than 1% of HTTP transfers in the Internet. Our results, thus, highlight the importance of using transient analysis to model short connections.

## 7. Example model application: comparison of high-speed protocols

In this section, we illustrate the usefulness of the LB-CC framework by using it to evaluate the relative performance of several recently-proposed "high-speed" variants of TCP congestion-control. Our aim is not to provide a comprehensive evaluation of these protocols, but simply to illustrate in some example settings how the LB-CC framework can help draw fundamental observations about the behavior of the protocol. We consider a set of diverse network topologies in which: (i) the bottleneck transmission capacity is set to either 100 Mbps, 1 Gbps, or 2.5 Gbps, (ii) the RTT is set to either 10 ms or 100 ms, and (iii) the packet loss rate can take on values ranging from $10^{-7}$ to $10^{-3}$. We model three prominent high-speed protocols—namely, HighSpeed TCP (HSTCP), Scalable TCP (S-TCP), and Square Root Fair (SRF) TCP—and compute the time to transfer 100,000 segments (100 MB worth of data). Fig. 12(a), (c), and (e) plot the computed values for bottleneck capacities of 100 Mbps, 1 Gbps, and 2.5 Gbps, respectively. Each figure is a three-dimensional plot of the transfer time as a function of the packet loss probability and the path RTT.

We also compute the value of the instantaneous throughput, as defined in Section 6, attained by each of the protocols after transmitting 100,000 segments. Fig. 12(b), (d), and (f) plot this quantity using the three-dimensional view. Note that the direction of both the x- and y-axes is reversed from the three corresponding plots for transfer time.

(a) 100 Mbps

(b) 100 Mbps

(c) 1 Gbps

(b) 1 Gbps
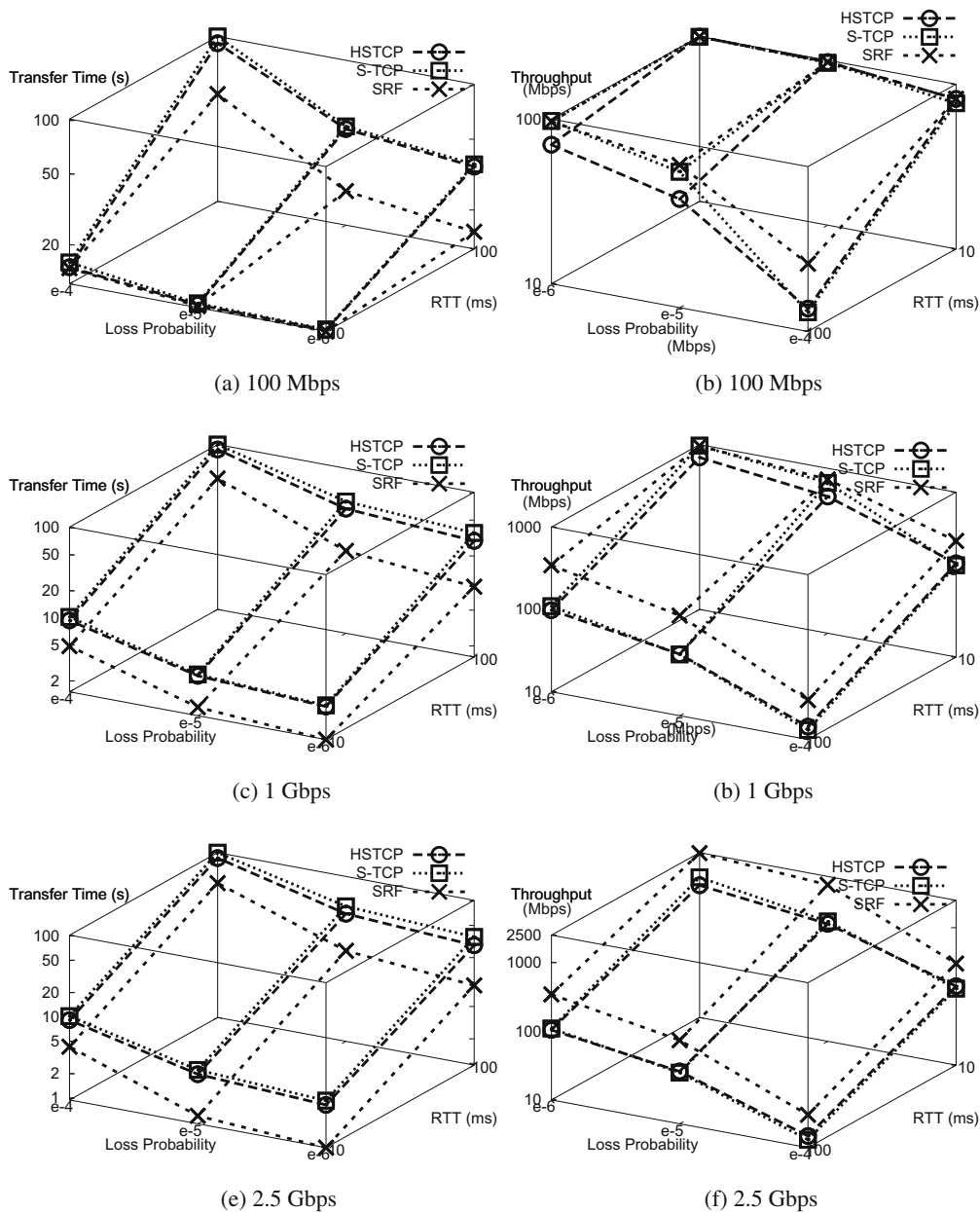
(e) 2.5 Gbps

(f) 2.5 Gbps

**Fig. 12.** Time-to-transfer and throughput achieved after transmitting 100,000 segments.

We find that:

- As expected, a higher packet loss rate as well as a larger path RTT increase the transfer-time and decrease the instantaneous throughput level attained after transmitting the 100,000 segments.
- With large RTTs, high link capacity does not have a significant impact on the transfer-time or the instantaneous throughput level achieved. This is because the link capacity impacts the performance of a transfer only after the congestion-window has reached a value equal to the bandwidth-delay product. This product is quite large on large-RTT networks (larger than the congestion-window value that is attained after sending only 100,000 segments). For instance, we find that the throughput level achieved in quite similar in a 100 ms RTT network across the three values of bottleneck capacity studied.
- Under all network conditions, SRF always outperforms HSTCP and S-TCP (in both the transfer-time as well as instantaneous throughput metrics).

- S-TCP and HSTCP provide similar performance in several cases. However, HSTCP provides lower transfer times in high-speed networks, especially when the RTTs are high. In contrast, the instantaneous throughput achieved by S-TCP is higher than that of HSTCP in topologies with neither too large nor too low bandwidth-delay products (100 Mbps capacity and 100 ms RTT, or Gbps capacity and 10 ms RTT).
- In a 100 Mbps network, all protocols are able to attain a 100 Mbps worth of throughput when loss rates are low and RTTs are small—however, when both of these quantities have large values, none of the protocols attain that throughput after transmitting only 100,000 segments. In higher-speed networks, even with low loss rates and small RTTs, only SRF is successful in attaining a throughput equal to the bottleneck capacity after transmitting 100,000 segments.

We reiterate again that our purpose is not to provide a comprehensive evaluations of these three protocols, but merely to illustrate the power of the LB-CC framework in drawing several

fundamental insights (such as those listed above) about protocol performance.

## 8. Concluding remarks

In this paper, we present the design of a class of performance models that predict the transfer time for bulk TCP transfers under diverse settings of loss rates, round-trip times, end-to-end bottleneck capacities, protocol parameter settings, and protocol versions. We do so in two steps. First, we define the general class of Loss-based Congestion Control (LB-CC) protocols and demonstrate that many TCP variants, including those being proposed for high-speed networks, belong to this class. We then develop a stochastic framework to compute the mean and variance in transfer time for any LB-CC protocol. We validate our model against extensive simulations and show that it is accurate under more diverse settings than past models.

Our work leads to a number of useful modeling guidelines. First, our evaluations indicate that the bottleneck transmission capacity can have a significant impact on TCP performance in low-speed networks. It should, therefore, be incorporated in TCP analysis. Second, unlike what was previously assumed, the probability and impact of retransmission timeouts can take a range of values for a given packet loss rate. Since timeouts impact TCP performance significantly, this implies that either accurate techniques should be developed to relate timeout probability to packet loss probability, or the two should be treated independently in TCP analysis. We use the latter approach in this paper. Finally, our computations indicate that models that rely only on steady-state analysis may be applicable only to connections that transfer more than several megabytes. This underscores the importance of analyzing TCP's transient behavior.

We believe that the generalized LB-CC framework is a powerful tool that can be used in the design and analysis of next-generation transport protocols. In particular, our model provides the opportunity of evaluating the impact of different combinations of $f_i$, $g_i$, $h_i$ on TCP performance in high-speed networks. Furthermore, the stochastic framework developed in Section 4 facilitates the derivation of additional metrics, such as the distribution of $W_n$. As part of future work, we plan to systematically evaluate the impact of each model parameter on such quantities. Finally, we plan to derive empirical models of per-connection losses and round-trip times and use our framework to study their impact on real-world TCP performance.

## References

[1] CAIDA, Workload characterization. Available from: <http://www.caida.org/analysis/workload>.
[2] D. Anderson, H. Balakrishnan, M. Kaashoek, R. Morris, Resilient overlay networks, in: Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP), 2001.
[3] Global grid forum. Available from: <http://www.gridforum.org/>.
[4] S. Floyd, M. Handley, J. Padhye, J. Widmer, Equation-based congestion control for unicast applications, in: Proceedings of ACM SIGCOMM, 2000.
[5] J. Mahdavi, S. Floyd, TCP-friendly unicast rate-based flow control, note sent to end2end-interest mailing list, January 1997.
[6] L. Brakmo, S. O'Malley, L. Peterson, TCP Vegas: new techniques for congestion detection and avoidance, in: Proceedings of ACM SIGCOMM, 1994.
[7] K. Fall, S. Floyd, Simulation-based comparisons of Tahoe, Reno, and Sack TCP, ACM Computer Communication Review 26 (3) (1996) 5–21.
[8] S. Floyd, Highspeed TCP for large congestion windows, Internet Draft, June 2003.
[9] C. Jin, D. Wei, S. Low, G. Buhrmaster, J. Bunn, D. Choe, R. Cottrell, J. Doyle, H. Newman, F. Paganini, S. Ravot, S. Singh, FAST kernel: background theory and experimental results, in: First International Workshop on Protocols for Fast Long-distance Networks, 2003.
[10] T. Kelly, Scalable TCP: improving performance in high-speed wide area networks, in: First International Workshop on Protocols for Fast Long-distance Networks, 2003.
[11] L. Xu, K. Harfoush, I. Rhee, Binary increase congestion control for fast, long distance networks, in: Proceedings of IEEE INFOCOM, 2004.
[12] M. Fukuhara, F. Hirose, T. Hatano, H. Shigeno, K. Okada, SRF TCP: a TCP-friendly and fair congestion control method for high-speed networks, in: Proceedings of the International Conference on Principles of Distributed Systems (OPODIS), Lecture Notes in Computer Science, vol. 3544, Springer, Berlin, 2005, pp. 169–183.
[13] E. Altman, K. Avrachenkov, C. Barakat, A stochastic model of TCP/IP with stationary random losses, in: Proceedings of ACM SIGCOMM, 2000, pp. 231–242.
[14] E. Altman, K. Avrachenkov, A.A. Kherani, B.J. Prabhu, Performance analysis and stochastic stability of congestion control protocols, in: Proceedings of IEEE INFOCOM, 2005, pp. 1316–1327.
[15] A. Budhiraja, F. Hernandez-Campos, V. Kulkarni, F. Smith, Stochastic differential equation for TCP window size: analysis and experimental validation, Journal of Probability in Engineering and Information Sciences 18 (1) (2004) 111–140.
[16] N. Cardwell, S. Savage, T. Anderson, Modeling TCP latency, in: Proceedings of IEEE INFOCOM, 2000.
[17] C. Casetti, M. Meo, A new approach to model the stationary behavior of TCP connections, in: Proceedings of IEEE INFOCOM, 2000.
[18] S. Floyd, Connections with multiple congested gateways in packet-switched networks. Part 1: one-way traffic, Computer Communication Review 21 (5) (1991) 30–47.
[19] T. Lakshman, U. Madhow, The performance of TCP/IP for networks with high bandwidth-delay products and random loss, IEEE/ACM Transactions on Networking (1997).
[20] M. Mathis, J. Semske, J. Mahdavi, T. Ott, The macroscopic behaviour of the TCP congestion avoidance algorithm, ACM Computer Communication Review 27 (3) (1997) 67–82.
[21] V. Misra, W. Gong, D. Towsley, Stochastic differential equation modeling and analysis of TCP windowsize behavior, in: Proceedings of IFIP Performance'99, 1999.
[22] T. Ott, J. Kemperman, M. Mathis, Window size behavior in TCP/IP with constant loss probability, in: Proceedings of IEEE High Performance Communications Systems Workshop, 1997.
[23] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, Modeling TCP throughput: a simple model and its empirical validation, in: Proceedings of ACM SIGCOMM, 1998.
[24] B. Sikdar, S. Kalyanaraman, K. Vastola, Analytic models for the latency and steady-state throughput of TCP Tahoe, Reno and SACK, in: Proceedings of IEEE Globecom, 2001.
[25] B. Sikdar, S. Kalyanaraman, K.S. Vastola, Analytic models for the latency and steady-state throughput of TCP Tahoe, Reno, and SACK, IEEE/ACM Transactions on Networking 11 (2) (2003) 959–971.
[26] M. Mellia, I. Stoica, H. Zhang, TCP model for short lived flows, IEEE Communications Letters 6 (2) (2002) 85–87.
[27] W. Stevens, in: TCP/IP Illustrated, The Protocols, vol. 1, Addison-Wesley, 1994.
[28] S. Floyd, Highspeed TCP and quick-start for fast long-distance networks, in: Plenary Talk at the First International Workshop on Protocols for Fast Long-distance Networks, February 2003. Available from: <http://datatag.web.cern.ch/datatag/pfldnet2003/slides/floyd.pdf>.
[29] V. Jacobson, Congestion avoidance and control, in: Proceedings of ACM SIGCOMM, 1988, pp. 314–329.
[30] D. Katabi, M. Handley, C. Rohrs, Congestion control for high bandwidth-delay product networks, in: Proceedings of ACM SIGCOMM, 2002.
[31] J. Aikat, J. Kaur, D. Smith, K. Jeffay, Variability in TCP round-trip times, in: Proceedings of the ACM SIGCOMM Internet Measurement Conference, 2003.
[32] V. Paxson, End-to-end internet packet dynamics, in: Proceedings of ACM SIGCOMM, 1997.
[33] T. DeFanti, C.D. Laat, J. Mambretti, K. Neggers, B.S. Arnaud, Translight: a global-scale lambdagrid for e-science, Communications of the ACM 46 (11) (2003) 34–41.
[34] Internet2 netflow weekly reports. Available from: <http://netflow.internet2.edu/weekly/>.
[35] F. Smith, F. Hernandez-Campos, K. Jeffay, D. Ott, What TCP/IP protocol headers can tell us about the web, in: Proceedings of ACM SIGMETRICS, 2001, pp. 245–256.
[36] C. Barakat, E. Altman, Peformance of short TCP transfers, in: Proceedings of Networking, 2000.
[37] D. Chiu, R. Jain, Analysis of the increase and decrease algorithms for congestion avoidance in computer networks, Computer Networks and ISDN Systems 17 (1989) 1–14.
[38] A. Wierman, T. Osogami, J. Olsen, A unified framework for modeling TCP-Vegas, TCP-SACK, and TCP-Reno, in: Proceedings of IEEE MASCOTS, 2003.
[39] Matlab: high-performance numeric computation and visualization software. User's Guide, T.M.W. Inc., 1992.
[40] S. McCanne, S. Floyd, ns Network Simulator. Software available from: <http://www.isi.edu/nsnam/ns/>.
[41] J. Cao, W.S. Cleveland, Y. Gao, K. Jeffay, F.D. Smith, M.C. Weigle, Stochastic models for generating synthetic HTTP source traffic, in: Proceedings of IEEE INFOCOM, 2004, pp. 1547–1558.
[42] S. Floyd, Connections with multiple congested gateways in packet-switched networks. Part 2: two-way traffic, 1991, unpublished. Available from: <http://www.icir.org/floyd/papers/gates2.ps.Z>.