Mobile Networks and Computing

DIMACS

Series in Discrete Mathematics
and Theoretical Computer Science

Volume 52

Mobile Networks and Computing

DIMACS Workshop
Mobile Networks and Computing
March 25–27, 1999
DIMACS Center

Sanguthevar Rajasekaran
Panos Pardalos
D. Frank Hsu
Editors

NSF Science and Technology Center
in Discrete Mathematics and Theoretical Computer Science
A consortium of Rutgers University, Princeton University,
AT&T Labs–Research, Bell Labs (Lucent Technologies),
Telcordia Technologies, and NEC Research Institute

American Mathematical Society

This DIMACS volume contains papers from a DIMACS workshop on Mobile Networks and Computing, held March 25–27, 1999.

2000 *Mathematics Subject Classification.* Primary 68M10, 68M12, 90B18.

Library of Congress Cataloging-in-Publication Data

Mobile networks and computing : DIMACS workshop, mobile networks and computing, March 25–27, 1999, DIMACS Center / Sanguthevar Rajasekaran, Panos Pardalos, D. Frank Hsu, editors.
p. cm. — (DIMACS series in discrete mathematics and theoretical computer science, ISSN 1052-1798 ; v. 52)
"NSF Science and Technology Center in Discrete Mathematics and Theoretical Computer Science, a consortium of Rutgers University, Princeton University, AT&T Labs–Research, Bell Labs (Lucent Technologies), Telcordia Technologies, and NEC Research Institute."
Includes bibliographical references.
ISBN 0-8218-1547-4 (alk. paper)
1. Mobile computing—Congresses. 2. Mobile communication systems—Congresses. I. Rajasekaran, Sanguthevar. II. Pardalos, P. M. (Panos M.), 1954– III. Hsu, D. Frank (Derbiau Frank), 1948– IV. NSF Science and Technology Center in Discrete Mathematics and Theoretical Computer Science. V. Series.
QA76.59.M68 2000
006.3–dc21 00-023293

"Learn without flaw what is worthy of learning and lead a life true to what has been learnt"

- Thiruvalluvar (80-20 B.C.) (Thirukkural, Chapter 40: Education, Verse 1)

Foreword

The DIMACS Workshop on Mobile Networks and Computing was held on March 25–27, 1999 at Rutgers University. We would like to express our appreciation to Panos Pardalos, Sanguthevar Rajasekaran, R. Badrinath, and Frank Hsu for their efforts to organize and plan this successful workshop.

The workshop was part of the Special Year on Networks. We extend our thanks to Stuart Haber, David Johnson, and Mihalis Yannakakis for their work as special year organizers.

The workshop brought together theoreticians and practitioners working on various aspects of mobile computing such as sensor networks, smart spaces, field computing, and channel allocation.

DIMACS gratefully acknowledges the generous support that makes these programs possible. The National Science Foundation, through its Science and Technology Centers program, the New Jersey Commission on Science and Technology, the National Security Agency through its support of the Special Year on Networks, and DIMACS's partners at Rutgers, Princeton, AT&T Labs-Research, Bell Labs, NEC Research Institute, and Telcordia Technologies generously supported the special year.

Fred S. Roberts
Director

Robert Sedgewick
Co-Director for Princeton

Preface

In the context of the 1998-1999 DIMACS special year with focus on **Networks**, a three-day workshop entitled "Mobile Networks and Computing" was held on March 25-27, 1999 at the DIMACS Center, Rutgers University. The workshop was organized by Panos Pardalos, Sanguthevar Rajasekaran, R. Badrinath, and Frank Hsu. More than 50 researchers from universities, institutes, governmental agencies, and industrial companies attended the workshop. Participants from Asia, Australia, and Europe gave the workshop an important international component.

Advances in the technologies of networking, wireless communications, and miniaturization of computers lead to the rapid development in mobile communication infrastructure, and have engendered a new paradigm of computing. Users carrying portable devices can freely move around, while still being connected to the network. This provides flexibility in accessing information anywhere and at any time. On the other hand, the price of this flexibility has introduced new levels of complexity that were not encountered in software and protocol design in wired networking.

The new challenges in designing software systems for mobile networks include location and mobility management, channel allocation, power conservation, among others. This workshop is aimed at bringing together researchers from academia as well as the industry who are working on various aspects of mobile computing. Topics will include sensor networks, smart spaces, field computing, channel allocation, etc.

A total of 24 invited talks were presented. Overall, the workshop achieved a great success. This volume contains a collection of refereed papers from the workshop.

The workshop was sponsored by DIMACS, through grants from the National Science Foundation, the New Jersey Commission on Science and Technology, and other sources. We would like to take this opportunity to thank the sponsors, David Johnson for his support, the DIMACS staff, the participants, the authors, the anonymous referees, and the American Mathematical Society, for making the workshop successful and the publication of this volume possible.

Sanguthevar Rajasekaran, Panos Pardalos, and Frank Hsu
November 1999

Contents

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 52, 2000

Stochastic Modeling of a Single TCP/IP Session over a Random Loss Channel

Al-Hussein A. Abou-Zeid, Murat Azizoglu, and Sumit Roy

ABSTRACT. In this paper, we present an analytical framework for modeling the performance of a single TCP session in the presence of random packet loss. This framework may be applicable to communications channels that cause random packet loss modelled by appropriate statistics of the inter-loss duration. It is shown that the analytical model predicts the throughput for LANs/WANs (low and high bandwidth-delay products) with reasonable accuracy, as measured against the throughput obtained by simulation. Random loss is found to severely affect the network throughput, higher speed channels are found to be more vulnerable to random loss than slower channels, especially for moderate to high loss rates.

1. Introduction

TCP/IP has been designed for reliable networks in which most packet losses occur primarily due to network congestion. An important aspect of TCP is its window-based congestion avoidance mechanism [6]. In TCP/IP, when a node successfully receives a packet, it sends an acknowledgment (ACK) back to the source. At all times, the source keeps a record of the number of unacknowledged packets that it has released into the network. This number is called the congestion window size, or simply, *the window size.* The source is allowed to increase its window size as long as packets keep being acknowledged. The source detects a packet loss by either the non-arrival of a packet ACK within a certain time (maintained by a *timer*), or by the arrival of multiple ACKs with the same next expected packet number. A packet loss is interpreted by the source as an indication of congestion, and the source responds by reducing its window size so as not to overload the network with packets, thereby indirectly controlling the data rate. Thus modelling the dynamic behavior of congestion window size is key to analyzing TCP/IP throughput performance in a variety of situations. The system (source, network and receiver) is frequently called a 'self-clocked' system since the arrival of ACKs acts as the clock that increases the window size, while the loss of a packet acts as a 'reset' for the system.

1991 *Mathematics Subject Classification.* Primary 54C40, 14E20; Secondary 46E25, 20C20.

An important system parameter that affects TCP throughput is the ratio β of the buffer size available in the bottleneck links of the network to the link bandwidth-delay product. For LANs, the round-trip delay in a connection is small, so that the bandwidth delay product could be much smaller than the buffer size (large β). WANs, on the other hand, have large round-trip delays, so that the buffer size is typically smaller than the bandwidth-delay product (small β). It is easy to see that, for the same bandwidth delay product, increasing the buffer size at a bottleneck link (increasing β) decreases the amount of congestion. It is worth noting that the bandwidth-delay product determines the maximum number of packets that can be in transit between a source-destination pair.

Network congestion is not the only source of packet loss - random packet loss may be significant in many circumstances. For example, it may arise due to intermittent faults in hardware elements in wired networks. In a wireless network, multipath fading that characterizes many terestrial links may be modelled as leading to random packet loss depending on the fade rates relative to data rate on the channel. While random packet loss on the Internet has been reported in [8], it was not taken into consideration in the congestion control mechanism in TCP/IP. Previous research [2, 3, 4, 5] has shown that random packet loss (which is not due to congestion) may severely decrease the throughput of TCP because TCP interprets random packet loss to be due to congestion and hence lowers the input data rate into the network, and consequently the throughput.

In [2, 3], a discrete-time model for random packet loss was used in which any given packet is assumed to be lost with probability q independent of all other packets. This model induces a geometric distribution on the number of packets successfully transmitted between consecutive packet losses, that may or may not be appropriate for specific lossy networks. A different loss model was employed in [5] which assumed that packet loss is characterized by an inhomogeneous Poisson process. The steady-state distribution of the window size was obtained in [5] under the assumption of infinite buffer size. In contrast, in this paper we assume a continuous-time packet loss model governed by a general renewal process and investigate the effect of finite buffer size on the performance.

A basic system model is shown in Figure 1. An infinite source (i.e. one that always has a packet to send) releases packets into a buffer of size B upon receiving ACKs from the destination. The packets are then sent over a single link with capacity μ packets per second and a net delay of τ (propagation delay through the channel, any other processing delays etc.) is assumed. Define $T = \tau + 1/\mu$ to be the time between the start of transmission of a packet and the reception of an ACK for this packet. Then μT is the bandwidth-delay product and the ratio $\beta = \frac{B}{\mu T}$ is the buffer size normalized by the bandwidth-delay product.

2. Ideal Channels without Random Packet Loss

We first briefly review the operation of TCP for the case of ideal channels, and summarize the key results in [1, 3] relevant to our work. There are different versions of TCP - the popular Tahoe version developed by Jacobson [6] (TCP-T) as well as the Reno version (TCP-R) that incorporates a fast retransmit option together with a method for reducing the effect of slow start [7]. While [1, 3] considered both versions, our analysis concentrates on TCP-R only, since our aim is to present

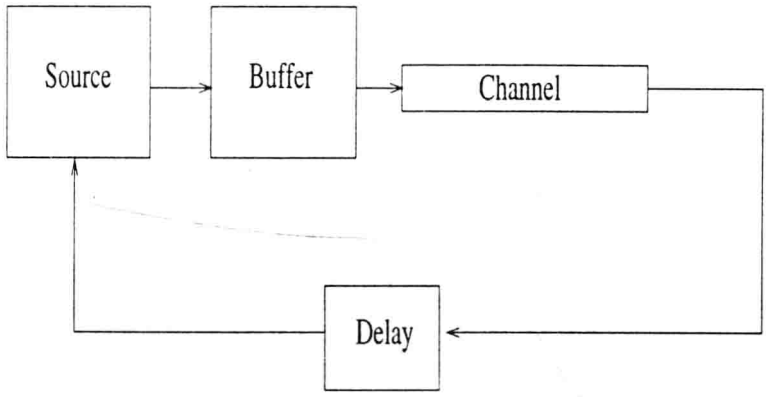

FIGURE 1. Block diagram of the system model

an analytical framework for the analysis of TCP/IP rather than to compare the different versions.

In TCP/IP, there are no explicit ACK or negative ACK signals. When the source sends a packet, it initializes a timer with an expiry time that is set depending on a current estimate of the delay τ. When a destination correctly receives a packet, it sends a signal to the source with the 'next expected' packet number. If this is received by the source prior to timer expiry, it is considered an ACK; otherwise, the packet is considered lost. In some implementations of TCP such as TCP-R, if the source receives multiple ACK signals with the same 'next expected' packet number, it interprets this as a packet loss.

Let $t' = 0$ denote the time of establishment of the TCP session under consideration, and let $W(t')$ denote the congestion window size at time t'. Then the algorithm followed by a TCP session (assuming 'ideal' operation of TCP) can be described as follows:

TCP-Tahoe:

1. Every time a packet ACK is received,
   if $W(t') < W_{th}$, set $W(t') = W(t') + 1$ ... **Slow Start Phase**
   elseif$((ACKcount++) == W(t'))$ { $ACKcount = 0, W(t') = W(t') + 1$ } ... **Congestion Avoidance Phase**

2. Every time a packet loss is detected,
   set $W_{th} = W(t')/2$ and set $W(t') = 1$. (**Go back to Slow Start Phase**)

TCP-Reno:

1. Every time a (non-repeated) packet ACK is received,
   if $W(t') < W_{th}$, set $W(t') = W(t') + 1$ ... **Slow Start Phase**
   elseif$((ACKcount++) == W(t'))$ { $ACKcount = 0, W(t') = W(t') + 1$ } ... **Congestion Avoidance Phase**

2. Every time a packet loss is detected via duplicate ACK,
   set $W_{th} = W(t')/2$, then set $W(t') = W(t')/2$ (the algorithm doesn't initiate

slow start)

3. Every time a packet loss is detected via timer expiry, the algorithm goes to slow start (as in TCP-T)
set $W_{th} = W(t')/2$ and set $W(t') = 1$.

A TCP-T session typically evolves as follows. A packet is released from the source into the buffer just after the session is established and the transmitter enters 'slow start' phase. Each time an ACK is received, the window size is incremented according to slow start until it reaches W_{th} at which time the algorithm switches to congestion avoidance phase. The source subsequently increases its window size by one only after every window's worth of acknowledgments (congestion avoidance phase). This continues until a packet loss is detected, whence the window size is reset to one and the algorithm re-enters the slow start phase. This cycle repeats itself until all the packets at the source are transmitted and acknowledged at which time the TCP/IP session is terminated. It is worth noting that during slow start, the window size actually increases rapidly; in order to increase the window size by one when an ACK is received, the source releases two packets simultaneously into the buffer, while, to keep the window size constant, the source releases only one packet (to replace the one just acknowledged). In slow start, the source always releases two packets at each ACK reception, while in congestion avoidance, the source releases one packet for each ACK reception (to keep the window size constant) except when a window's worth of acknowledgment is received, in which case it releases two packets.

In a TCP-R session, the first cycle after session establishment is identical to that of TCP-T. However, after the first packet loss, the algorithm does not go back to slow start. Instead, the algorithm reduces the window size to half its value and continues in the congestion avoidance phase. In this description, we have assumed that the algorithm implements Selective Acknowledgments (SACKs) so that loss of multiple packets does not lead to time out phenomena, and hence the algorithm never enters Step 3 in the pseudo-code above.

Finally, we assume that fast retransmit option is used [**11**] in the event of a packet loss detection to avoid lengthy stoppage of transmission.

Denote $w_p = \mu T + B = \mu\tau + B + 1$, and note that when the window size reaches w_p, the bit pipe (the combination of the channel and the transmit buffer) is fully utilized, i.e, the buffer is fully occupied and the maximum number of packets allowable are in transit. A further increase in window size at this stage causes buffer overflow, at which point the window size is reset (for TCP-T) or halved (for TCP-R) and W_{th} is set to $w_p/2$.

Sample functions of the window size obtained from simulations for TCP-Tahoe and TCP-Reno are shown in Figure 2 when the only source of packet loss is buffer overflow.

A note about buffer overflow is in order. As can be noticed in Figure 2, the buffer build up rate in the slow start phase is very high, due to the 'fast' nature of slow start. Hence, one would expect that for small values of β, buffer overflow can take place in slow start. This is actually true, and has been noted in [**3**]. In such a case, for TCP-T, the window size is set back to one and another slow start phase follows with a lower threshold. For TCP-R, the window size is set to half its value

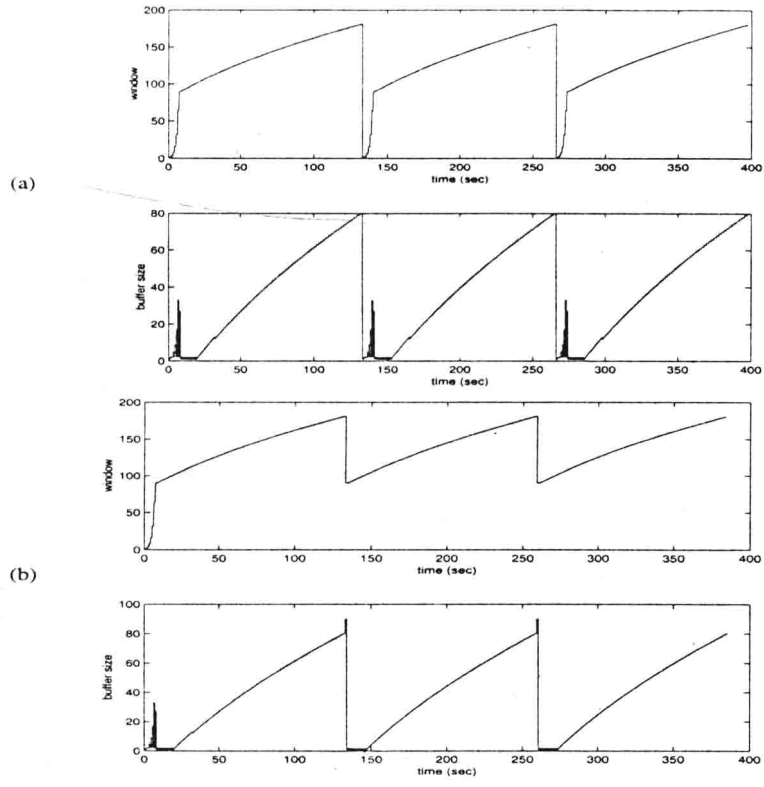

FIGURE 2. Window and buffer size evolution without random loss of packets for $\mu = 100$, $\tau = 1.0$, $\beta = 0.8$. (a)TCP-T, (b) TCP-R.

and the window evolution continues as depicted earlier. The effect of this 'double' slow start is significant for TCP-T but negligible for TCP-R.

The algorithms outlined above, defining the window size evolution during a TCP session have been studied in [**1, 3**] and mathematical expressions have been obtained for the envelope of the window size, denoted, for convenience, also by $W(t')$, for each of the phases (slow start and congestion avoidance). We summarize those expressions in a convenient way (so as to be able to use them in the following section) as follows (refer to Figure 3). Let n denote the number of packets acknowledged during a time interval t. Then the window evolution can be expressed as follows:

1. **Slow Start** $(1 < W(t') < W_{th})$. Consider two instants t'_0, $t'_0 + t$ in a slow start phase of any of the TCP cycles. Choose t'_0 such that $W(t'_0) = 1$.Then,

$$W(t'_0 + t) = 2^{t/T} \tag{2.1}$$

$$n = W(t'_0 + t) - 1 \tag{2.2}$$

2. **Congestion Avoidance - Phase I** $(W_{th} < W(t') < \mu T)$. Consider two instants t'_0, $t'_0 + t$ in a congestion avoidance phase of any of the TCP cycles.

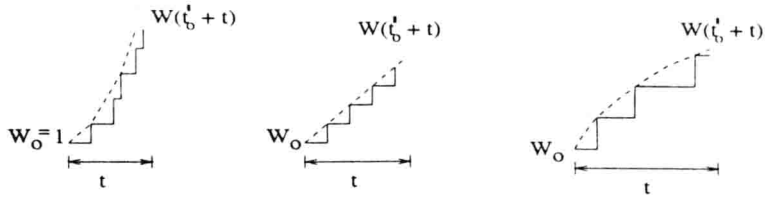

FIGURE 3. Sketch of the exponential, linear and sub-linear $O(\sqrt{t})$ phases for window evolution. Solid lines indicate the actual window size evolution while dotted lines indicate the envelope.

Choose t'_0 such that $W(t'_0) = W_0$.Then,

$$W(t'_0 + t) = W_0 + t/T \tag{2.3}$$

$$n = \frac{1}{T}(W_0 t + t^2/(2T)) \tag{2.4}$$

3. **Congestion Avoidance - Phase II** ($\mu T < W(t') < w_p$). Consider two instants t'_0, $t'_0 + t$ in a congestion avoidance phase of any of the TCP cycles. Choose t'_0 such that $W(t'_0) = W_0$.Then,

$$W(t'_0 + t) = \sqrt{W_0^2 + 2\mu t} \tag{2.5}$$

$$n = \mu t \tag{2.6}$$

Note that in the sequel, we focus on time instants t' where $W(t')$ is discrete. Equation (1) follows from considering the time instances at which packets are sent and packet ACKs are received, and noting that two packets are sent for each ACK received. Equation 2 follows from the fact that each time an ACK is received, $W(t')$ is incremented by one; since $W(t')$ is initially equal to one, the total number of packets acknowledged at the end of the slow start phase is one less than equal to the window size at the end of the slow start phase. Equations (3)-(6) follow from a continuous time approximation of the window size. Specifically in the congestion avoidance phase, if we approximate the window size by a continuous time function, then

$$\frac{dW(t')}{dt'} = min\{\frac{1}{T}, \frac{\mu}{W(t')}\},$$

the solution to which yields (3),(5). Note that (3) indicates a linear growth of the window size with time with a slope of $1/T$, while (5) specifies a growth that is $O(\sqrt{t})$. This can be explained as follows - for $W(t') < \mu T$, the channel is able to transmit $W(t')$ packets in T seconds (linear growth with slope $1/T$), until the window size reaches μT, at which time buffer build-up set in. Thus it takes more than T seconds to send $W(t')$ packets subsequently and hence the window increase is slower than linear.

Using (2.1) - (2.6) and taking into consideration the periodic evolution of the window size outlined in the previous discussion, it is straightforward to compute the average packet transmission rate R as the ratio of the number of packets sent in one cycle of the TCP session to the time duration of the cycle. Note that for the case of TCP-R, we neglect the first cycle since it is different from the rest of the cycles (it is the only one that contains slow start). The average transmission rate is given by

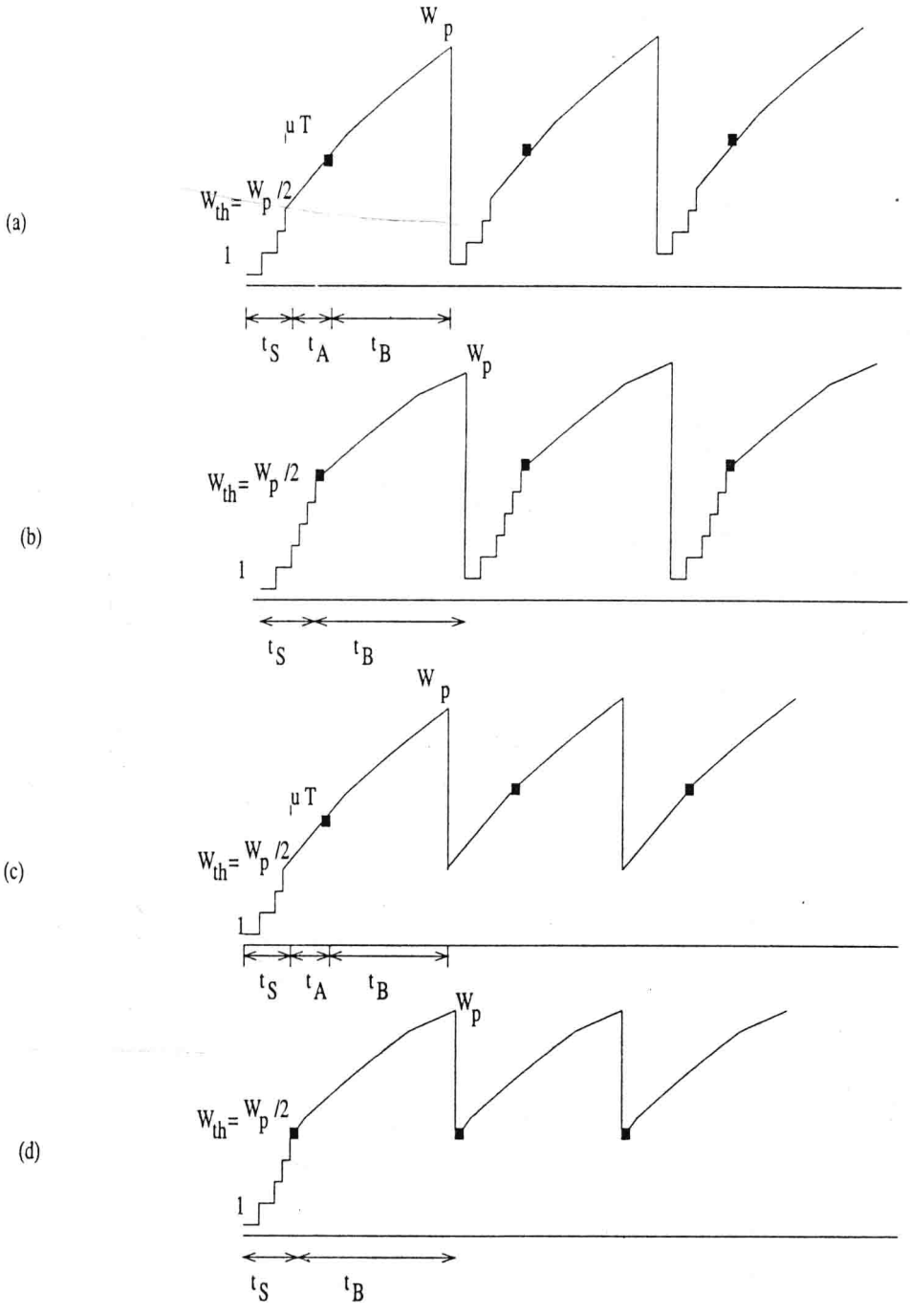

FIGURE 4. Sketch of the TCP window evolution without random loss ; (a) TCP-T with $\beta < 1$ (b) TCP-T with $\beta > 1$, (c) TCP-R with $\beta < 1$, (d) TCP-R with $\beta > 1$.Note that for $\beta < 1$, $w_p/2 < \mu T$, and hence the window evolution has a linear phase, while for $\beta > 1$, $w_p/2 > \mu T$ and hence the window evolution doesn't have a linear phase.

TCP-T:

$$\beta < 1 : R = \frac{n_S + n_A + n_B}{t_S + t_A + t_B} \tag{2.7}$$

$$\beta > 1 : R = \frac{n_S + n_B}{t_S + t_B} \tag{2.8}$$

TCP-R:

$$\beta < 1 : R = \frac{n_A + n_B}{t_A + t_B} \tag{2.9}$$

$$\beta > 1 : R \simeq \mu \tag{2.10}$$

The average throughput is then calculated as

$$\rho = \frac{R}{\mu} \tag{2.11}$$

The values above n_S, n_A, n_B, t_S, t_A and t_B are obtained by substituting for W_0 and $W(t')$ (see Figure 3) in (1)-(6) by the initial and final values of the slow start and congestion avoidance phases as indicated in Figure 4. For example, to compute n_S and t_S for TCP-T, set $W(t_S) = w_p/2$ in (1); to compute n_B and t_B for TCP-R and $\beta < 1$, let $W_0 = \mu T, W(t_B) = w_p$ in (3); to compute n_B and t_B for TCP-R and $\beta > 1$, let $W_0 = w_p/2, W(t_B) = w_p$ in (3) and so on.

Note the difference in the expressions for $\beta < 1$ and $\beta > 1$. Consider TCP-T; for $\beta < 1$, $w_p/2 < \mu T$ and hence the cyclical evolution of the window size consists of an exponential growth (the slow start phase from 1 to $w_p/2$) described by (1), then a linear growth (congestion avoidance phase from $w_p/2$ to μT) expressed by (3) and then $O(\sqrt{t})$ growth (congestion avoidance phase from μT to w_p) described by (5). On the other hand, for $\beta > 1$, $w_p/2 > \mu T$, and hence, the cyclical evolution consists of exponential growth (slow start from 1 to $w_p/2$) followed by $O(\sqrt{t})$ growth (congestion avoidance from $w_p/2$ to w_p) without having a linear phase. Similar observations apply for TCP-R; refer to Figure 4 for a schematic diagram of the window size evolution for each of the two ranges of β.

3. Channels with Random Packet Loss

3.1. Random Loss Model. Let S_i denote the time of the i^{th} packet loss, for $i = 1, 2, \ldots$. Let $X_i = S_i - S_{i-1}$ denote the time between $(i-1)^{\text{th}}$ and i^{th} packet losses with $X_1 = S_1$ by convention. As stated earlier, we will consider $\{X_1, X_2, \ldots\}$ to be a set of IID random variables with probability density function $f(x)$ and distribution function $F(x)$. Thus, the process (pdf) defined by the loss occurrence times $\{S_1, S_2, \ldots\}$ is a renewal process with interrenewal pdf $f(x)$.

Now, suppose that at a certain time instant X_1 $(=S_1)$, the first *random* packet loss event occurs. Denote the window size at that instant by W_1. When the source detects this loss (by the arrival of duplicate ACKs for the case of TCP-R), the window size is halved. The window size now increases as depicted earlier (the window size starts from $W_1/2$ and increases till w_p, at which time a buffer overflow takes place and $W(t')$ is set to $w_p/2$, and so on) until another random packet loss takes place at a random time instant $S_2 = X_1 + X_2$. Denote the window size at this time (the time of the second loss) by W_2.

In what follows, we call one period from $w_p/2$ till w_p the free-running period or the 'typical' cycle (i.e. free from random loss effects). Note that the second

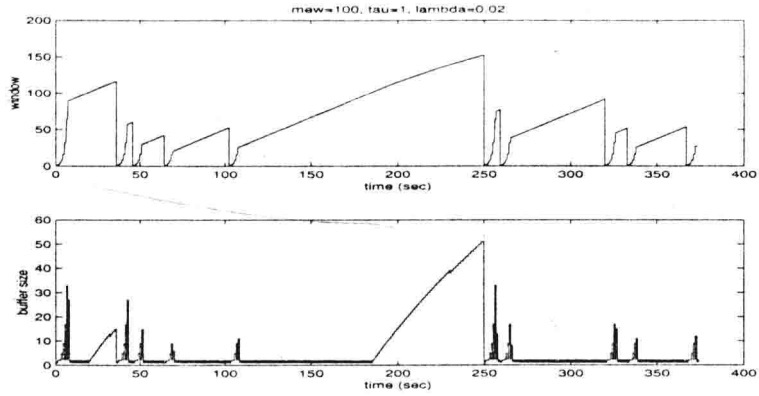

FIGURE 5. Window and buffer size evolution in with random loss of packets for a TCP-T session with $\mu = 100$, $\tau = 1.0$, $\beta = 0.8$ and $\lambda = 0.02$

random loss event can happen before the occurrence of any 'typical' cycles. The window size $W(t')$ is a semi-Markovian stochastic process, because the window size evolution after a random loss (except for its starting value which is half of that just before the random loss) is statistically independent from the window size evolution before the random loss. Further, since $\{X_1, X_2, \dots\}$ are independent and identically distributed (IID), the window sizes $\{W_1, W_2, \dots\}$ (window sizes just before the random loss) form a finite state Markov Chain (i.e. the embedded Markov Chain of the semi-Markov process $W(t')$) [**9, 10**].

3.2. Analysis Assumptions and Notations. With the above model of random loss, two quantities associated with the just defined Markov Chain are of interest

(1) $E[N|W_1 = w_1]$, the expected number of packets successfully transmitted before another random packet loss occurs, given that the most recent random loss took place at w_1; (2) The conditional probability $P[W_2 = w_2|W_1 = w_1]$ (denoted for convenience by P ; the probability that the next *random* loss takes place at $W_2 = w_2$ given that the previous *random* loss took place at $W_1 = w_1$.

Before we attempt to evaluate the above two quantities of interest, we make an approximation for TCP-R, similar to the approximation previously invoked for the channels without random loss. We ignore the first cycle of TCP-R and assume that the TCP session starts with window size $w_p/2$ instead of starting with a window size of 1. This approximation should have a negligible effect on the average throughput, due to two reasons ; (1) A source with an infinite infinite number of packets was assumed; hence the transient behavior (slow start) at the beginning of the connection is expected to be negligible, even for the case of random loss; (2) The duration as well as the number of packets sent during this slow start phase is low (recall that slow start is actually 'fast'). In other words, this transient behavior disappears very fast. We comment on the effect of this approximation in the results section following the analysis.

In the analysis that follows, two ranges of β are considered separately, $\beta < 1$ and $\beta > 1$, and expressions for $E[N|W_1]$ and $P[W_2|W_1]$ are found for each of the two ranges.