Throughput Analysis of TCP on Channels with Memory

Michele Zorzi, *Senior Member, IEEE*, A. Chockalingam, *Senior Member, IEEE*, and Ramesh R. Rao, *Senior Member, IEEE*

***Abstract*—The focus of this paper is to analyze the relative sensitivity of the bulk throughput performance of different versions of TCP, viz., OldTahoe, Tahoe, Reno, and New Reno, to channel errors that are correlated. We investigate the performance of a single wireless TCP connection in a local environment by modeling the correlated packet loss/error process (e.g., as induced by a multipath fading channel) as a first-order Markov chain. A major contribution of the paper is a *unified analytical approach* which allows the evaluation of the throughput performance of various versions of TCP. The main findings of this study are that 1) error correlations significantly affect the performance of TCP, and in particular may result in considerably better performance for Tahoe and NewReno; and 2) over slowly fading channels which are characterized by significant channel memory, Tahoe performs as well as NewReno. This leads us to conclude that a clever design of the lower layers that preserve error correlations, naturally present on wireless links because of the fading behavior, could be an attractive alternative to the development or the use of more complex versions of TCP.**

***Index Terms*—Bursty errors, Markov channel, slow fading, TCP.**

I. Introduction

DUE TO rapid advances in the area of wireless communications and the popularity of the Internet, provision of packet data services for applications like e-mail, web browsing, and mobile computing over wireless channels is gaining importance. Transport Control Protocol (TCP) is a reliable, end-to-end, transport protocol that is widely used to support applications like telnet, ftp, and http [1].

TCP was designed primarily for wireline networks where the channel error rates are very low and congestion is the primary cause of packet loss [2]. Since its original deployment, several modifications to TCP, including Reno, NewReno, and Vegas, have been proposed and their performance analyzed in wireline networks [2]–[4]. Reno's loss recovery algorithm is optimized for the case when a single packet is lost in a window of data. Hence, Reno can suffer performance problems when multiple packets are lost in a window [3]. NewReno addresses this problem by improving the loss recovery phase to handle this situation [3], [5]. However, the performance of these enhanced TCP versions on wireless fading links has not been adequately studied so far.

There have been several recent investigations on different facets of wireless TCP [5]–[13]. Most of these studies do not consider the effect of *correlation in multipath fading* [14], [15]. Errors are often assumed to occur independently and with the same probability on each packet. Yet, because the multipath fading process in a mobile radio environment can be slowly varying for typical values of carrier frequency and user speed, the dependence between errors in the transmissions of consecutive packets of data cannot be neglected. Although motivated by the behavior of the wireless channel, the loss model considered here also applies to any environment where the packet loss process exhibits memory, e.g., due to congestion.

Related work has been presented in [12], where the performance of the OldTahoe and Tahoe versions of TCP is analyzed assuming a two state Markov channel model. A simplified analytical model for the throughput of the Tahoe version in the presence of Markovian packet losses has been presented in [15], where it was shown that correlation has a beneficial effect. An open question in this regard is the relative difference between the performance of various versions of TCP over the fading channel.

In this paper, we compare the performance of OldTahoe, Tahoe, Reno, and NewReno. In each instance, a single TCP connection is assumed to extend over a multipath fading channel modeled as a first-order Markov packet error model, as in [12] and [16]. We assume that a large data file is to be transferred from the base station to a mobile terminal, over a 1.5 Mbps wireless link that is characterized by very low delay-bandwidth product. As in [5], we assume instantaneous ACK's. The assumption of a single TCP connection, with the TCP end points being at the base station and at the mobile terminal, is consistent with IS-99 [17], [18].

Two main findings of this paper are that 1) error correlations significantly affect the performance of TCP, and in particular result in considerably better performance for Tahoe and NewReno; and 2) over slow fading channels, Tahoe performs as well as NewReno. This leads to the conclusion that a clever design of the lower layers that preserves error correlations, naturally present on wireless links because of the fading behavior, could be an attractive alternative to the development or the use of more complex versions of TCP. Another interesting observation that can be drawn from the results presented in this paper is that using different versions of TCP, or even the same version but with different parameters, may lead to significant changes

Manuscript received December 7, 1998; revised December 13, 1999. This work was supported in part by the Center for Wireless Communications, University of California, San Diego.

M. Zorzi is with the Dipartimento di Ingegneria, Università di Ferrara, 44100 Ferrara, Italy (e-mail: zorzi@ing.unife.it).

A. Chockalingam is with the Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore-560012, India (e-mail: achockal@ece.iisc.ernet.in).

R. R. Rao is with the Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093-0407 USA (e-mail: rrao@ucsd.edu).

Publisher Item Identifier S 0733-8716(00)05287-2.

in the energy consumption performance of the protocol, directly affecting the battery life of a portable device. This issue is discussed in detail in [19].

This work provides a *unified analytical approach* to the study of all four versions of TCP considered. This greatly simplifies the assessment of the sensitivity of the various versions to the system parameters in a variety of situations. So far, no analytical approaches have been developed for the *correlated channel* case, except for Tahoe [12]. Also, this study differs from the results reported in [18] in that we examine OldTahoe, Tahoe, Reno, and NewReno without any underlying link protocol, whereas the focus in [18] was on the specific IS-99 system which uses a radio link protocol (RLP) below the TCP layer. TCP with an underlying FEC/ARQ link layer protocol is a topic of ongoing investigation [23].

The paper is organized as follows. In Section II, we describe the OldTahoe, Tahoe, Reno, and NewReno versions of TCP. The system model and the correlated fading channel model considered in this paper are presented in Section III. The analytical approach is introduced in Section IV. Section V provides the results comparing the performance of TCP OldTahoe, Tahoe, Reno, and NewReno on correlated fading channels. Finally, conclusions and topics of future research are provided in Section VI.

II. TCP OldTahoe, Tahoe, Reno, and NewReno

In this section, we provide a description of the receive and transmit processes in TCP OldTahoe, Tahoe, Reno, and NewReno. While the receive processes are the same for all of them, their transmit processes are different in the way the *loss recovery phase* is implemented. The following description of the receive and transmit processes follows that of [5].

The TCP receiver can accept packets out of sequence, but will only deliver them in sequence to the TCP user. During connection setup, the receiver advertizes a maximum window size, $W_{\max}$, so that the transmitter does not allow more than $W_{\max}$ unacknowledged data packets outstanding at any given time. The receiver sends back an acknowledgment (ACK) for every data packet it receives correctly. The ACK's are cumulative. That is, an ACK carrying the sequence number m acknowledges all data packets up to, and including, the data packet with sequence number $m-1$. The ACK's will identify the next expected packet sequence number, which is the first among the packets required to complete the in-sequence delivery of packets. Thus, if a packet is lost (after a stream of correctly received packets), then the transmitter keeps receiving ACK's with the sequence number of the first packet lost (called duplicate ACK's), even if packets transmitted after the lost packet are correctly received at the receiver.

The TCP transmitter operates on a window based transmission strategy as follows. At any given time t, there is a lower window edge $A(t)$, which means that all data packets numbered up to, and including, $A(t)-1$ have been transmitted and acknowledged, and that the transmitter can send data packets from $A(t)$ onwards. The transmitter's congestion window, $W(t)$, defines the maximum amount of unacknowledged data packets the transmitter is permitted to send, starting from $A(t)$. Under normal data transfer, $A(t)$ has nondecreasing sample paths. However, the adaptive window mechanism causes $W(t)$ to increase or decrease, but never to exceed $W_{\max}$. Transitions in the processes $A(t)$ and $W(t)$ are triggered by the receipt of ACK's. The receipt of an ACK that acknowledges some data will cause an increase in $A(t)$ by an amount equal to the amount of data acknowledged. The change in $W(t)$, however, depends on the particular version of TCP and the congestion control process. Each time a new packet is transmitted, the transmitter starts a timer. If such timer reaches the *round-trip timeout* value (derived from a round-trip time estimation procedure [1]) before the packet is acknowledged, timeout timer expiration occurs, and retransmission is initiated from the next packet after the last acknowledged packet. The timeout values are set only in multiples of a timer granularity [1].

The basic window adaptation procedure, common to all TCP versions [20], works as follows. Let $W(t)$ be the transmitter's *congestion window width* at time t, and $W_{th}(t)$ be the *slow-start threshold* at time t. The evolution of $W(t)$ and $W_{th}(t)$ are triggered by ACK's (new ACK's, and not duplicate ACK's) and timeouts as follows.

1) If $W(t) < W_{th}(t)$, each ACK causes $W(t)$ to be incremented by 1. This is the *slow start* phase.
2) If $W(t) \geq W_{th}(t)$, each ACK causes $W(t)$ to be incremented by $1/(W(t))$. This is the *congestion avoidance* phase.
3) If timeout occurs at the transmitter at time t, $W(t^+)$ is set to 1, $W_{th}(t^+)$ is set to $\lceil W(t)/2 \rceil$, and the transmitter begins retransmission from the next packet after the last acknowledged packet.

Note that the transmissions after a timeout always start with the first lost packet. The window of packets transmitted from the lost packet onwards, but before retransmission starts, is called the *loss window*.

Besides running the basic window adaptation algorithm, the transmitter performs the following tasks which are related to packet losses:

- *loss detection:* a mechanism by which the transmitter concludes (correctly or incorrectly) that a packet was lost
- *loss recovery phase:* a mechanism which allows the protocol to recover lost packets through retransmission
- *window adaptation during loss recovery:* the way window adaptation is handled while lost packets are being recovered (different than the basic window adaptation in general).

The above procedures are implemented in TCP OldTahoe, Tahoe, Reno, and NewReno as follows.

- In the case of OldTahoe, loss detection and recovery is performed only through timeout and retransmission. Window adaptation during loss recovery follows the basic algorithm.
- In the case of Tahoe, in addition to the regular timeout mechanism, a *fast retransmit* procedure is implemented for loss detection. If subsequent to a packet loss, the transmitter receives the Kth duplicate ACK at time t, before the

timer expires, then the transmitter behaves as if a timeout has occured and begins retransmission, with $W(t^+)$ and $W_{th}(t^+)$ as given in the basic window adaptation algorithm.

- In the case of Reno also, the fast retransmit procedure following a packet loss is implemented. However, the subsequent recovery procedure is different. If the Kth duplicate ACK is received at time t, then $W_{th}(t^+)$ is set to $\lceil W(t)/2 \rceil$, and $W(t^+)$ is set to $W_{th}(t^+) + K$ instead of 1 (the addition of K accounts for the K packets that have successfully left the network). The Reno transmitter then retransmits only the first lost packet. As the transmitter waits for the ACK for the first lost packet retransmission, it may get duplicate ACK's for the outstanding packets. The receipt of each of such duplicate ACK causes $W(t)$ to be incremented by 1. If there was only a single packet loss in the loss window, then the ACK for its retransmission will complete the loss recovery; W at this time would be set to W_{th}, and the transmission resumes according to the basic window control algorithm. If there are multiple packet losses in the loss window, then the ACK for the first lost packet retransmission will advance the left edge of the window, A, by an amount equal to 1 plus the number of good packets between the first lost packet and the next one. In this case, if the loss recovery is not successful due to lack of the duplicate ACK's necessary to trigger multiple fast retransmits, then a timeout has to be waited for. The above data loss recovery strategy in Reno is shown to perform better than Tahoe when single packet losses occur in the loss window, but can suffer performance problems when multiple packets are lost in the loss window [3].
- In the case of NewReno, fast retransmit and congestion window adaptation are as in Reno, but the loss recovery mechanism is as in Tahoe [5]. That is, NewReno will not send the first lost packet alone and wait for its ACK like Reno. Instead it will continue with the transmission of subsequent packets like Tahoe. The NewReno version of the protocol can recover from multiple packet losses in some cases.

III. System Model

Data exchange in TCP involves a connection setup phase, a data transfer phase, and a connection tear-down phase. In this paper, we are primarily interested in the bulk throughput performance of TCP. Consequently, we model only the data transfer phase of the protocol which dominates the overall performance. We consider a single transmitter–receiver pair running TCP on a dedicated link characterized by a two-state Markov packet error process, zero propagation delay, and perfect feedback. The transmitter is assumed to have an infinite supply of packets to send.

In particular, in the numerical evaluations, a 1.5 Mbps wireless data link with a negligibly small bandwidth-delay product is assumed. Therefore, the acknowledgments (ACK's) from the mobile receiver arrive instantaneously at the base station. The ACK packets are assumed to arrive error-free. These assumptions may be expected to be reasonable in wireless local environments where the propagation delays are small, and the ACK packets are relatively smaller in size than data packets (40 bytes versus 500–1500 bytes). We checked the validity of these assumptions by running some simulations[1] including ACK delay and ACK errors. As expected, for typical values of the parameters as encountered in a local wireless environment (e.g., ACK delays less than a couple of TCP slot intervals and ACK error probability not exceeding 0.01), the effect of ACK delays and errors on the TCP throughput performance is negligible.

As is usually done in most studies taking an analytical approach, we assume the presence of a single TCP connection over the wireless channel. Unlike in [5], where the base station side TCP/IP stack is placed on a fixed LAN host from which packets are routed through an intermediate system to the mobile terminal, we assume that the TCP/IP stack is directly placed at the base station similar to TCP placement in IS-99 [17]. Consequently, there is no queueing and no delay due to the intermediate system.

A. Channel Model

We model the correlation in the multipath fading process using a first-order Markov model for the process of packet errors, as proposed in [16]. The statistics of the packet errors is then fully characterized by the transition matrix of such process:

$$M_c = \begin{pmatrix} p_{BB} & p_{BG} \\ p_{GB} & p_{GG} \end{pmatrix} \tag{1}$$

where p_{BG} is the transition from bad to good, i.e., the conditional probability that successful transmission occurs in a slot given that a failure occurred in the previous slot, and the other entries in the matrix are defined similarly. The average probability of a packet loss, P_E, can be found from the probabilities in (1), which in turn depend on the physical characterization of the channel, usually expressed in terms of the *fading margin*, F, and the *normalized Doppler bandwidth*, f_dT, where T is the packet duration [16]. Detailed relationships between the Markov transition probabilities and the fading channel parameters are given in [16].

To apply this Markov model at the TCP packet level, we assume a TCP packet size of 1400 bytes. At 1.5 Mbps rate, this corresponds to a packet transmission time T of about 7.5 ms. By choosing different P_E and f_dT values, we can establish fading channel models with different degrees of correlation in the fading process. When f_dT is small, the fading process is very correlated (long bursts of packet errors); on the other hand, for larger values of f_dT, successive samples of the channel are almost independent (short bursts of packet errors). Table I shows the Markov parameters p_{GG} and p_{BB}, and the average length of a burst of erroneous packets, $(1 - p_{BB})^{-1}$, for different values of P_E and f_dT.

It should be stressed here that the parameters given in Table I are computed according to the simplified analytical model of [16]. One may expect to find some discrepancies between these values and the values obtained through actual simulations over

[1]Simulation results presented in the paper have been obtained using Berkeley's network simulator `ns`.

TABLE I
MARKOV PARAMETERS p_{GG} AND p_{BB} AT DIFFERENT VALUES OF P_E AND f_dT

f_dT	P_E	F (dB)	p_{GG}	p_{BB}	$(1-p_{BB})^{-1}$
0.01	0.001	29.9978	0.999329	0.329452	1.49132
	0.01	19.9782	0.997518	0.754308	4.07013
	0.1	9.77322	0.991872	0.926851	13.6708
0.08	0.001	29.9978	0.999007	0.008239	1.00831
	0.01	19.9782	0.990680	0.077286	1.08376
	0.1	9.77322	0.940354	0.463188	1.86285
0.64	0.001	29.9978	0.999000	0.001185	1.00238
	0.01	19.9782	0.990019	0.011834	1.01198
	0.1	9.77322	0.90182	0.116376	1.13170

the fading channel. The characterization of the packet error behavior induced by fading processes would merit a study by itself, and is beyond the scope of this paper. The fully analytical approach adopted here (Markov modeling of the error process with analytically computed parameters, coupled with the analytical throughput evaluation via Markov analysis) has the advantage of providing results quickly and to allow extensive investigation of the effect of system parameters. Moreover, the results and behaviors observed for more complicated simulation models (where we run the protocol over the actual error process produced by a Rayleigh fading trace) are very similar to the ones observed here, and therefore the potential of the analysis to provide useful insight is preserved. Some more discussion about the modeling of the fading channel is given in [19].

The case of independent and identically distributed (i.i.d.) errors will be also considered for comparison. In this case, the error process is memoryless and $p_{BB} = p_{GB} = P_E$.

Although motivated by the behavior of the wireless channel, the loss model considered here applies to any environment where the packet loss process exhibits memory, e.g., due to congestion. I.i.d. loss models do not capture this and Markov models are a natural choice in this case.

IV. Analytical Approach

A. Joint Window/Channel Evolution

The analysis is based on a Markov/renewal reward approach. The following discussion, for the purpose of description and to introduce precisely the methodology, will refer to TCP Reno. With minor changes, the following can be adapted to the other versions of TCP, as will be detailed in later subsections.

The joint evolution of the window parameters and the channel state can be tracked by a random process $\mathcal{X}(t) = (C(t-1), W_{th}(t), W(t))$, where $W(t)$ and $W_{th}(t)$ are the window size and the slow start threshold in slot t, respectively, and $C(t-1)$ is the channel state (bad, B, or good, G, corresponding to an erroneous or correct transmission, respectively) in slot $t-1$. Time is discrete, and the slot (packet transmission time) is the time unit. Unfortunately, this process is not Markov, since its evolution starting from a certain state also depends on other quantities not accounted for in $\mathcal{X}(t)$, such as the number of outstanding packets (i.e., packets whose ACK is being waited for) and the "age" of each, implemented through a timeout timer. Incorporating these quantities into the process description would make it Markov but would produce such a large state space as to make its solution impractical.

An alternative is to sample the process appropriately. Specifically, we look for appropriate instants t_k such that $X(k) \triangleq \mathcal{X}(t_k)$ is a Markov process.

Consider a time slot immediately after a slot in which a timeout timer expired. According to the rules of TCP Reno, at that time the window size shrinks to 1 and all timers are reset, so that, from the point of view of the window adaptation algorithm, no outstanding packets are present. Therefore, at these instants, knowledge of $(C(t-1), W_{th}(t), W(t))$ is all there is to know to characterize the window/channel evolution in the future.

Likewise, consider a time slot immediately following a slot in which the loss recovery phase was successfully completed. At this time, by definition, all outstanding packets have been acknowledged, and therefore no timeouts are active. Again, $(C(t-1), W_{th}(t), W(t))$ is all there is to know to characterize the window/channel evolution in the future.

Therefore, if we sample the process $\mathcal{X}(t) = (C(t-1), W_{th}(t), W(t))$ by choosing as sampling instants t_k's the slots immediately following those in which either a timeout timer expires or a loss recovery phase is successfully completed, we obtain a process $X(k) = \mathcal{X}(t_k)$ which is Markov. As an additional benefit, we note that, again from the protocol rules, the value of $W(t_k)$ can only be equal to 1 (timeout case) or to $W_{th}(t)$ (successful loss recovery). Finally, note that the channel state at time $t_k - 1$ can be either erroneous or correct in the timeout case, but can only be correct for a successful loss recovery, which must be ended by a successful transmission. Therefore, the state space of the process $X(k)$ is given by

$$\Omega_X = \left\{(C, W_{th}, 1), C = B, G, 1 \le W_{th} \le \left\lceil \frac{W_{\max}}{2} \right\rceil \right\} \cup \left\{(G, W_{th}, W_{th}), 1 \le W_{th} \le \left\lceil \frac{W_{\max}}{2} \right\rceil \right\} \tag{2}$$

where the first set corresponds to timeout and the second set corresponds to successful recovery phase. Note that the total number of states is in this case $3\lceil W_{\max}/2 \rceil - 1$.

B. Counting of Transmissions and Successes

For the purpose of evaluating metrics of interest, such as throughput, the above information is not sufficient, since it does not track when transmission attempts and successful transmissions occur. Also, it does not track time, since no information is given about the span between successive sampling instants.

In order to be able to characterize these quantities, we consider a semi-Markov process (as defined in [21, Ch. 10]) which admits $X(k)$ as its embedded Markov chain. That is, we label transitions of the chain $X(k)$ with *transition metrics*, which track the (possibly random) events which determine time delay, transmissions, and successes. For a given transition, let N_d be the associated number of slots, N_t the number of transmissions, and N_s the number of successful transmissions (the transition in question will be explicitly identified only when needed).

For this model to be semi-Markov, one must guarantee that the probability distribution of the transition metrics is uniquely determined by the origin i and the destination j of the transition itself and, once conditioned on the pair i, j, they are independent of past and future evolution. One should observe that while this is rigorously verified for time delays and transmission attempts, it is not for successes, since whether or not a successful transmission is to be accounted for toward throughput depends on whether or not it had been already transmitted and counted in the past.[2] One way around this problem, which we will adopt in this paper, is to consider two ways of counting successes. Whenever there is uncertainty about whether or not a successful transmission should be counted as a true success, counting it will lead to an optimistic throughput evaluation (upper bound), whereas not counting it would lead to a pessimistic throughput evaluation (lower bound). Note that in both cases, the number of successes counted for a transition only depends on when transmissions occur and on the channel state during those slots. Since both quantities evolve according to a semi-Markov processes, the two proposed counting strategies result in semi-Markov processes as well, and provide rigorous stochastic bounds. Tightness of these bounds has been verified to be always very good, so that this technique is very accurate.

As a final comment, we remark that bounding techniques will also be used in some cases to simplify the computation of some metrics. In fact, even though precise computation of all quantities would be possible in some cases, it is much better to replace it with an optimistic and a pessimistic approximations, which give tight bounds with the advantage of being simpler to compute.

C. Semi-Markov Analysis

Let t_k be the kth sampling instant determined according to the above rules (without loss of generality, assume $t_1 = 1$.) We define *cycle* k as the time evolution of the system between the two consecutive sampling instants t_k and t_{k+1}. The statistical behavior of a cycle only depends on the channel state at time $t_k - 1$ and on the slow start threshold and window size at time t_k. Also, we assume that the process is stationary, so that everything is independent of k.

When we look at the evolution of the process during a generic cycle k, it is implicit in what follows that system variables are conditioned on $X(k) = (C(t_k - 1), W_{th}(t_k), W(t_k)) \in \Omega_X$, where Ω_X is the set of all possible values of $X(k)$ (state space of the sampled process). For simplicity of notation, let $C(t_k - 1) = C$.

Let $n \geq 1$ be the first slot of the cycle to contain an erroneous transmission. The probability distribution of n, conditioned on the state at the beginning of the cycle, is given by

$$\begin{aligned}\alpha_C(n) &= P[\text{first error at } t = n \mid C(0) = C] \\ &= \begin{cases} p_{CB} & n = 1 \\ p_{CG} p_{GG}^{n-2} p_{GB} & n > 1. \end{cases}\end{aligned} \tag{3}$$

Let $Y(k)$ be the system state at time n in cycle k. Since transmissions in slots $1, 2, \cdots, n-1$ were successful, there are no outstanding packets except for the one transmitted at time n. Also, by definition we know that the channel state at time n is B. Finally, according to the protocol rules during the loss recovery phase, the value of the slow start threshold at time n, $W_{th}(n)$, does not play any role in determining the state at the beginning of cycle $k+1$.[3] Therefore, the state space Ω_Y only consists of the possible values of the window size at time n, which is the only quantity to be tracked. The size of Ω_Y is $3W_{\max}/2 - 1$ (for $W_{\max}$ even), which results from appropriate quantization of the actual window size (which is a real number in general), as discussed in the Appendix.

The system evolution during a cycle can then be separated into two parts. In the first part, the system makes a transition from a state $X(k) \in \Omega_X$ to a state $Y(k) \in \Omega_Y$; whereas in the second part, the system makes a transition from a state $Y(k) \in \Omega_Y$ to a state $X(k+1) \in \Omega_X$. Due to the feedforward structure of the transitions, we can consider the two parts separately, and then combine the results to obtain the complete description of a full cycle.

More specifically, let $\mathbf{\Phi}^{(1)}(z)$ be a matrix whose ijth entry is the transition function associated to the transition from state $i \in \Omega_X$ to state $j \in \Omega_Y$, and analogously let $\mathbf{\Phi}^{(2)}(z)$ be a matrix whose $j\ell$th entry is the transition function associated to the transition from state $j \in \Omega_Y$ to state $\ell \in \Omega_X$ [21]. The statistics of the system evolution during a cycle is fully characterized by the matrix

$$\mathbf{\Phi}(z) = \mathbf{\Phi}^{(1)}(z)\mathbf{\Phi}^{(2)}(z) \tag{4}$$

whose entries are the transition functions associated to transitions from Ω_X to itself.

The variable z is in general a vector of transform variables, each of which tracks a quantity of interest. In our case, we set $z = (z_d, z_t, z_s)$, to track the delay, number of transmissions and number of successes. More precisely, let $\xi_{ij}(N_d, N_t, N_s)$ be the probability that the system makes a transition to state j in exactly N_d slots, and that in $\{1, 2, \cdots, N_d\}$ N_t transmission attempts are performed and N_s transmission successes are counted, given that the system was in state i at time 0. Then, we have

$$\Phi_{ij}(z_d, z_t, z_s) = \sum_{N_d, N_t, N_s} \xi_{ij}(N_d, N_t, N_s) z_d^{N_d} z_t^{N_t} z_s^{N_s}. \tag{5}$$

In particular, the transition matrix of the embedded Markov chain is just given by $\boldsymbol{P} = \mathbf{\Phi}(1, 1, 1)$. The matrix of average delays can be found as

$$\begin{aligned}\boldsymbol{D} &= \left.\frac{\partial \mathbf{\Phi}(z_d, z_t, z_s)}{\partial z_d}\right|_{z_d, z_t, z_s = 1} \\ &= \boldsymbol{D}_1 \mathbf{\Phi}^{(2)}(1, 1, 1) + \mathbf{\Phi}^{(1)}(1, 1, 1)\boldsymbol{D}_2\end{aligned} \tag{6}$$

[2]In fact, it is possible that a packet transmission performed with good channel conditions is not acknowledged because older packets were incorrectly received and await retransmission. Further evolution may lead to this packet being transmitted again. Ignoring these cases would lead to incorrect double-counting of some packets.

[3]According to the rules of the algorithm, $W_{th}(t_{k+1})$ is determined based on $W(n)$ and on what happens during the loss recovery phase (if any), and $W(t_{k+1})$ is either 1 or $W_{th}(t_{k+1})$. Finally, the channel state is not influenced by the window evolution.

where

$$\boldsymbol{D}_1 = \left.\frac{\partial \boldsymbol{\Phi}^{(1)}(z_d, z_t, z_s)}{\partial z_d}\right|_{z_d, z_t, z_s = 1}$$
$$\boldsymbol{D}_2 = \left.\frac{\partial \boldsymbol{\Phi}^{(2)}(z_d, z_t, z_s)}{\partial z_d}\right|_{z_d, z_t, z_s = 1}. \tag{7}$$

The averages of the other quantities, $\boldsymbol{T}$, $\boldsymbol{S}$, can be found similarly.

According to the theory presented in [21] and [22], from the above quantities we can compute a number of steady-state performance parameters. In particular, we can evaluate the average throughput as

$$\text{throughput} = \frac{\sum_{i \in \Omega_X} \pi_i \sum_{j \in \Omega_X} P_{ij} S_{ij}}{\sum_{i \in \Omega_X} \pi_i \sum_{j \in \Omega_X} P_{ij} D_{ij}} \tag{8}$$

where $\pi_i, i \in \Omega_X$, are the steady-state probabilities of the Markov chain with transition matrix $\boldsymbol{P}$. The average number of transmissions per slot can be similarly computed, by using T_{ij} instead of S_{ij}, and is related to the energy consumption of the protocol [19].

Note that, in order to compute steady-state performance from this analysis, knowledge of $\boldsymbol{P} = \boldsymbol{\Phi}(1,1,1)$ and of $\boldsymbol{D}, \boldsymbol{T}$, and $\boldsymbol{S}$ is sufficient. On the other hand, wherever possible, we will give the full form of the transition functions, which may be useful in further elaborations (e.g., computation of higher moments [21]).

In the following, we evaluate the transition functions $\boldsymbol{\Phi}^{(1)}(z)$ and $\boldsymbol{\Phi}^{(2)}(z)$. The major task here is to correctly identify all possible transitions and the associated transition functions. We proceed in the following way. For each possible system state (transition origin):

1) a set of mutually exclusive events is identified, exhausting all possibilities;
2) for each of those events, based on the origin state and on the protocol rules:
    a) the destination of the corresponding transition is identified
    b) the transition function is computed;
3) transitions corresponding to distinct events but leading to the same destination state are combined (i.e., the corresponding transition functions are added), to obtain $\boldsymbol{\Phi}^{(1)}(z)$ and $\boldsymbol{\Phi}^{(2)}(z)$.

D. Computation of $\boldsymbol{\Phi}^{(1)}(z)$

Since the first part of a cycle consists of error-free transmissions, and since all versions of TCP considered here have the same window adaptation mechanism as long as there are no errors, the computation of $\boldsymbol{\Phi}^{(1)}(z)$ described here applies to all versions of TCP.

Let $X = X(k) = (C, W_{th}, W)$. The first part of the cycle has a duration of $N_d = n$ slots with probability $\alpha_C(n), n \geq 1$. Conditioned on the value of n, the value of the window size at time n is a deterministic function $w(n, W, W_{th})$ of W_{th} and W that can be easily tabulated, and is assumed here to be known. Therefore, the window size at time n is denoted by

$$Y = W(n) = w(n, W, W_{th}). \tag{9}$$

Also, conditioned on the value of n, $N_d = n$ slots, $N_t = n$ packet transmissions and $N_s = n - 1$ packet successes. Therefore,

$$\boldsymbol{\Phi}^{(1)}_{XY}(z_d, z_t, z_s) = \sum_{n \in \mathcal{C}(X,Y)} \alpha_C(n) z_d^n z_t^n z_s^{n-1} \tag{10}$$

where $\mathcal{C}(X,Y) = \{n: w(n, W, W_{th}) = Y\}$.

E. Computation of $\boldsymbol{\Phi}^{(2)}(z)$ for TCP Reno

The second part of the cycle can also be fully characterized by appropriately labeling transitions and counting events. Unlike in the previous case, $\boldsymbol{\Phi}^{(2)}(z)$ does depend on the way the different TCP versions handle packet loss recovery, and therefore it must be computed separately in the various cases. We address the case of TCP Reno in this subsection.

For simplicity of notation, in what follows we let $n = 0$, so that the first slot in the second phase corresponds to time 1. Define $\varphi_{ij}(k, x)$ as the probability that there are k successes in slots 1 through x and that the channel is in state j at time x, given that the channel was in state i at time 0. Both a recursive technique and explicit expressions for these probabilities are given in [24]. Note that the functions ϕ in that paper are defined in a slightly different way. However, it is straightforward to relate them by noting that $\varphi_{BG}(j, x) = \phi_{10}(j-1, x)$ and $\varphi_{GB}(j, x) = \phi_{01}(j+1, x)$, whereas in the other two cases they are the same.

1) The Case of $\lfloor Y \rfloor \leq K$: If $\lfloor Y \rfloor \leq K$, fast retransmit cannot be triggered, since after the packet lost at time 0, only $\lfloor Y \rfloor - 1 < K$ more packets can be transmitted, and K duplicate ACK's will never be received. In this case, timeout timer will expire and the lost packet will be retransmitted in slot $t_{k+1} = T_o$. Note that the value of the window size at timeout will still be equal to Y (recall that duplicate ACK's do not advance the window), so that after timeout the algorithm will set $W_{th} = \lceil Y/2 \rceil, W = 1$. This event will therefore lead to state $X(k+1) = (C, \lceil Y/2 \rceil, 1)$ with transition function

$$p_{BC}(T_o - 1) z_d^{T_o - 1} z_t^{\lfloor Y \rfloor - 1} z_s^{N_s}. \tag{11}$$

Note that in (11), the quantity N_s is a random variable. In order to find the specific expression for the transition function, we would have to further condition on the random events involved. This would be possible, but very tedious. Instead, we use here a simpler approach, recalling that for our purposes we only need the average of the reward functions on each transition, i.e., in this case, $E[N_s]$. This is also tedious to compute, but can be easily bounded[4] by noting that $N_s \geq 0$ cannot exceed the number of successful slots in $\{1, 2, \cdots, \lfloor Y \rfloor - 1\}$, so that $0 \leq E[N_s] \leq \sum_{i=1}^{\lfloor Y \rfloor - 1} p_{BG}(i)$, where $p_{BG}(i)$ is the i-step transition probability of the Markov channel, computed as

$$M_c(i) = \begin{pmatrix} p_{BB}(i) & p_{BG}(i) \\ p_{GB}(i) & p_{GG}(i) \end{pmatrix} = \begin{pmatrix} p_{BB} & p_{BG} \\ p_{GB} & p_{GG} \end{pmatrix}^i. \tag{12}$$

[4]Bounds obtained in this way proved to be very tight.

2) The Case of $\lfloor Y \rfloor > K$: Let us now assume that $\lfloor Y \rfloor > K$. The following two cases can occur.

Case 1—Fast Retransmit is Not Triggered: If fewer than K slots in $\{1, 2, \cdots, \lfloor Y \rfloor - 1\}$ are successful, fast retransmit will not be triggered. The destination values of W_{th} and W will be as in the previous case. Let $\mathcal{A}(C, j)$ be the event that there are j successful slots in $\{1, 2, \cdots, \lfloor Y \rfloor - 1\}$ and that the channel state in slot $\lfloor Y \rfloor - 1$ is C. From the Markov channel characterization, the following probabilities can be derived: $P[\mathcal{A}(G, j)] = \varphi_{BG}(j, \lfloor Y \rfloor - 1)$ and $P[\mathcal{A}(B, j)] = \varphi_{BB}(j, \lfloor Y \rfloor - 1)$. The transition function leading from Y to state $X(k+1) = (C, \lceil Y/2 \rceil, 1)$ is then given by

$$p_{BC}(T_o - \lfloor Y \rfloor) z_d^{T_o - 1} z_t^{\lfloor Y \rfloor - 1} \sum_{j=0}^{K-1} P[\mathcal{A}(B, j)] z_s^{N_s(B,j)} + p_{GC}(T_o - \lfloor Y \rfloor) z_d^{T_o - 1} z_t^{\lfloor Y \rfloor - 1} \sum_{j=1}^{K-1} P[\mathcal{A}(G, j)] z_s^{N_s(G,j)} \tag{13}$$

where $0 \leq N_s(B, j), N_s(G, j) \leq j$ and the two terms account for the two possibilities for the channel state at time $\lfloor Y \rfloor - 1$. The sums are limited to $K - 1$ rather than $\lfloor Y \rfloor - 1$ since the number of successes must be less than K for the considered case of fast retransmit not triggered.

Case 2—Fast Retransmit is Triggered: If the Kth duplicate ACK is received, fast retransmit is triggered right after the Kth successful slot in $\{1, 2, \cdots, \lfloor Y \rfloor - 1\}$. Let $\mathcal{B}(K)$ be the event that the packet failure at time 0 is followed by K consecutive successes, and let $\mathcal{B}(i, \ell_1), K < i < \lfloor Y \rfloor, 0 < \ell_1 \leq K$ be the event that the Kth success occurs at time i and the first loss after the loss in 0 occurs at time ℓ_1 (note that since $i > K$, there must be a packet loss before the Kth success). The probabilities of these events are given as follows:

$$P[\mathcal{B}(K)] = p_{BG} p_{GG}^{K-1} \tag{14}$$

$$P[\mathcal{B}(i, \ell_1)] = \begin{cases} p_{BB}\varphi_{BG}(K, i-1), \\ \quad \ell_1 = 1;\ i = K+1, \cdots, \lfloor Y \rfloor - 1 \\ p_{BG} p_{GG}^{\ell_1 - 2} p_{GB} \varphi_{BG}(K - \ell_1 + 1, i - \ell_1), \\ \quad \ell_1 = 2, \cdots, K;\ i = K+1, \cdots, \lfloor Y \rfloor - 1. \end{cases} \tag{15}$$

Case 2a: Consider first the occurrence of the event $\mathcal{B}(K)$. Since at time K the Kth duplicate ACK is received, retransmission of that packet is performed in slot $K + 1$.

- If this retransmission is successful, the loss recovery phase is successfully completed, and a new cycle starts at time $K + 2$. In this case, the destination state is $X(k+1) = (G, \lceil Y/2 \rceil, \lceil Y/2 \rceil)$[5] and the transition function is given by

$$P[\mathcal{B}(K)] p_{GG} z_d^{K+1} z_t^{K+1} z_s^{K+1}. \tag{16}$$

- If, on the other hand, the retransmission is a failure, the protocol will stop and wait for an ACK which will never be transmitted, and timeout will eventually resolve the deadlock. In this case, according to the TCP Reno rules, upon receiving the Kth duplicate ACK the window size will be updated to $W' = \min\{\lceil Y/2 \rceil + K, W_{\max}\}$, so that the new state after timeout is $X(k+1) = (C, \lceil W'/2 \rceil, 1)$, with transition function

$$P[\mathcal{B}(K)] p_{GB} p_{BC}(T_o - K - 2) z_d^{T_o - 1} z_t^{K+1} z_s^K. \tag{17}$$

Note that the total number of successful transmissions to be counted in this case is K, since according to the TCP Reno rules, the K successful transmissions will be eventually acknowledged (when the failed packet is finally transmitted successfully) without being retransmitted (in fact, as long as the failed packet keeps failing, the window size remains equal to 1 and no other packets are transmitted).

Case 2b: Consider the occurrence of the event $\mathcal{B}(i, \ell_1)$. Successful loss recovery is not possible here, since in TCP Reno, multiple losses in a congestion window lead to deadlock and consequent timeout.

- If the retransmission at time $i + 1$ is a failure, a behavior similar to the previous case can be observed, i.e., the next cycle will start in state $X(k+1) = (C, \lceil W'/2 \rceil, 1)$, with transition function given by

$$P[\mathcal{B}(i, \ell_1)] p_{GB} p_{BC}(T_o - i - 2) z_d^{T_o - 1} z_t^{i+1} z_s^{N_s} \tag{18}$$

where $\ell_1 - 1 \leq N_s \leq K$. Note, in fact, that the $\ell_1 - 1$ successful packets consecutively transmitted after the loss at time 0 will be acknowledged (without ever being retransmitted) when that lost packet is eventually received successfully, so that $\ell_1 - 1 \leq N_s$. Also, since K packets were successfully transmitted, in the best case they will all be acknowledged without being retransmitted, i.e., $N_s \leq K$.

- On the other hand, if the retransmission at time $i + 1$ is successful, all packets preceding the one transmitted at time ℓ_1 are acknowledged, and the system will timeout at the end of slot $T_o + \ell_1 - 1$. In this case, the window size at that time will be $W'' = \min\{\lceil Y/2 \rceil + K + 1, W_{\max}\}$ (the ACK for the successful retransmission causes the window to be further increased by one with respect to the previous case), so that the next cycle will start in state $X(k+1) = (C, \lceil W''/2 \rceil, 1)$ with transition function

$$P[\mathcal{B}(i, \ell_1)] p_{GG} p_{GC}(T_o + \ell_1 - i - 2) \times z_d^{\ell_1 + T_o - 1} z_t^{i+1} z_s^{N_s} \tag{19}$$

where $\ell_1 \leq N_s \leq K + 1$. Note in fact that, in this case, the number of successes to be counted is at least one more than before, accounting for the successful retransmission at time $i + 1$, but cannot be larger than $K + 1$.

3) Transition Functions and $\Phi^{(2)}(z)$: Let $\mathcal{E}(Y)$ be an exhaustive set of mutually exclusive events associated with transitions from state $Y(k) = Y$. Also, let $d_Y(\cdot)$: $\mathcal{E}(Y) \rightarrow \Omega_X$ be a well-defined function, giving the destination state corresponding to each event in $\mathcal{E}(Y)$. Note that this requires that events be defined so that the destination is uniquely specified. However, distinct events may lead to the same destination. Also,

[5]Note in fact that according to Reno's rules, after the Kth duplicate ACK is received, W_{th} is set to half the window size Y, and never changed until at last at successful completion of the loss recovery phase the window size is set equal to $W_{th} = \lceil Y/2 \rceil$.

let the function ψ map each event to the corresponding transition function. We can then formally find the YX-th entry of the transition function matrix $\mathbf{\Phi}^{(2)}$ as

$$\mathbf{\Phi}_{YX}^{(2)}(z) = \sum_{\mathcal{A}\in\mathcal{D}(X,Y)} \psi(\mathcal{A}) \tag{20}$$

where $\mathcal{D}(X,Y) = \{\mathcal{A} \in \mathcal{E}(Y)\colon d_Y(\mathcal{A}) = X\}$ is the set of all events leading from $Y \in \Omega_Y$ to $X \in \Omega_Y$ during phase two.

F. TCP OldTahoe and Tahoe

In the case of TCP OldTahoe and Tahoe, there is no fast recovery. In Tahoe, once fast retransmit is triggered, regular transmission is resumed, starting from the first unacknowledged packet. In OldTahoe there is no fast retransmit. The sampling rule for the window evolution therefore needs to be changed in these cases. We still consider the instants immediately following timeout. In addition (for Tahoe only), we also consider the instants in which a retransmission due to fast retransmit is performed.

Consider OldTahoe first. $\mathbf{\Phi}^{(1)}(z)$ is found as for TCP Reno. For $\mathbf{\Phi}^{(2)}(z)$, note that the second part of the cycle, initiated by the loss at time 0, has duration which is deterministically equal to T_o, since every loss can only be recovered by timeout. The next cycle then starts in state $X(k+1) = (C, \lceil Y/2\rceil, 1)$ with transition function

$$p_{BC}(T_o-1) z_d^{T_o-1} z_t^{\lfloor Y\rfloor-1} z_s^{N_s} \tag{21}$$

where $N_s \geq 0$ is upper-bounded by the number of successful slots in $\{1, 2, \cdots, \lfloor Y\rfloor - 1\}$, so that, in particular, $0 \leq E[N_s] \leq \sum_{i=1}^{\lfloor Y\rfloor-1} p_{BG}(i)$.

Let us now focus on Tahoe. In this case, again, $\mathbf{\Phi}^{(1)}(z)$ is the same as before. Regarding the computation of $\mathbf{\Phi}^{(2)}(z)$, we first observe that all the events considered for TCP Reno and corresponding to no fast retransmit still apply in the case (note, in fact, that Tahoe and Reno differ only *after* fast retransmit is triggered). Therefore, we only need consider here the case in which fast retransmit is triggered, i.e., the case in which the Kth successful transmission occurs at time $i < \lfloor Y\rfloor$. The next cycle then starts in slot $i+1$ and in state $X(k+1) = (G, \lceil Y/2\rceil, 1)$, with transition function

$$\varphi_{BG}(K,i) z_d^i z_t^i z_s^{N_s} \tag{22}$$

where $0 \leq N_s \leq K$.

G. TCP NewReno

Finally, let us consider TCP NewReno. The only case in which it is different from TCP Reno is when there are multiple losses within the same congestion window and the retransmission performed due to fast retransmit is successful (second bullet of Case 2b above). All other cases are the same as in Reno.

Let us then consider the case in which the loss at time 0 is not followed by K consecutive successful transmission (Reno and NewReno are different only when *multiple losses* occur). Recall the definition of $\mathcal{B}(i,\ell_1), K < i < \lfloor Y\rfloor, 0 < \ell_1 \leq K$ as the event that the Kth success occurs at time i and the first loss after the loss in 0 occurs at time ℓ_1. If the Kth duplicate ACK is received at time $i > K$, then besides the first lost packet (at time 0) there are $i-K$ losses in the window.

1) Successful Loss Recovery: If after slot i we have $i+1-K$ consecutive successes, loss recovery is successfully completed, since at time $2i+1-K$ the last packet to be lost is successfully retransmitted and the corresponding ACK will acknowledge all outstanding packets. A new cycle will then start at time $2i+2-K$ in state $X(k+1) = (G, \lceil Y/2\rceil, \lceil Y/2\rceil)$, since the slow start threshold W_{th}, after being set to $\lceil Y/2\rceil$ upon reception of the Kth duplicate ACK, remains unchanged, and W is set to W_{th} upon completion of the loss recovery phase. The transition function corresponding to this event is

$$P[\mathcal{B}(i,\ell_1)] p_{GG} 1^{i+1-K} z_d^{2i+1-K} z_t^{2i+1-K} z_s^{i+1} \tag{23}$$

since all the i packets transmitted until the Kth duplicate ACK is received, plus the very first loss, are acknowledged when the loss recovery phase is completed.

2) Unsuccessful Loss Recovery: Consider now the case in which the loss recovery phase does not end successfully, i.e., after the Kth duplicate ACK there are less than $i-K$ consecutive successes. It would be possible to extend the analysis by tracking the exact location of the losses by means of a vector $(\ell_1, \ell_2, \cdots, \ell_{i-K})$, but this leads potentially to a very complicated analysis. Instead, we propose a bounding technique which has about the same complexity of the analysis previously proposed for Reno.

With probability $p_{GG}^j p_{GB}$, after the successful retransmission of the first loss, we have exactly j consecutive good slots, followed by a failure. After the jth success, the value of the window is given by $W''' = \min\{\lceil Y/2\rceil + K + 1 + j, W_{\max}\}$. The initial state of the next cycle corresponds to timeout of the $(j+2)$nd loss in the window, since the first $j+1$ have been successfully retransmitted, and is $X(k+1) = (C, \lceil W'''/2\rceil, 1)$. In order to precisely determine the statistics of the exact time at which the cycle starts, t_{k+1}, and channel state in the slot $t_{k+1}-1$, C, we would need to know in which slot the $(j+2)$nd loss was originally transmitted. Instead, we take two bounding approaches.

- Assume that $C = B$ and that the cycle duration is maximum, i.e., $N_d = \lfloor Y\rfloor - 1 + T_o - 1$ (i.e., the loss which times out occurred right before the Kth success). Also, let $N_t = 2i - K$ (upper bound) and $N_s = \ell_1$ (lower bound). This situation clearly corresponds to the worst case for all quantities. Note that the window parameters of the destination state are precisely determined, whereas assuming $C = B$ certainly leads to a pessimistic estimate. In this case, the transition function is expressed as

$$P[\mathcal{B}(i,\ell_1)] p_{GG}^{j+1} p_{GB} z_d^{\lfloor Y\rfloor-1+T_o-1} z_t^{2i-K} z_s^{\ell_1}. \tag{24}$$

- On the other hand, we can assume that $C = G$, that the cycle duration is minimum, i.e., $N_d = \ell_1 + T_o - 1$, and that N_s and N_t are maximum and minimum, respectively, i.e., $N_s = i$ and $N_t = i+1$. This corresponds to the best case, and therefore provides an upper bound to the performance. The transition function in this case is

$$P[\mathcal{B}(i,\ell_1)] p_{GG}^{j+1} p_{GB} z_d^{\ell_1+T_o-1} z_t^{i+1} z_s^{i}. \tag{25}$$

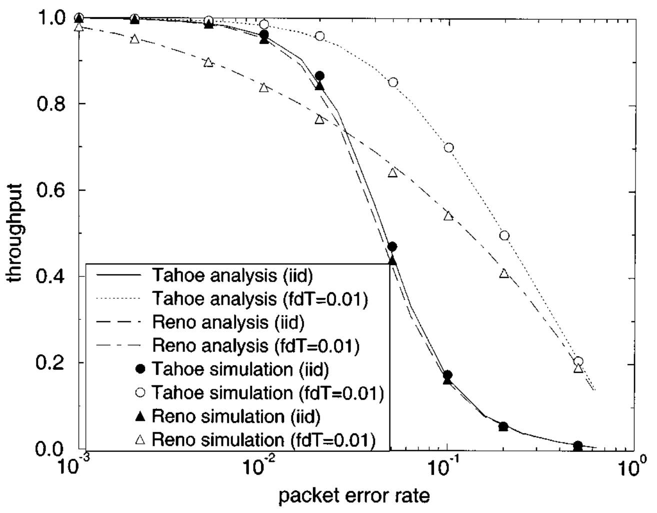


Fig. 1. Throughput performance of TCP Reno and Tahoe for i.i.d. and $f_dT = 0.01$. $W_{\max} = 6$. $K = 3$. $MTO = 100$. Analysis and simulation.

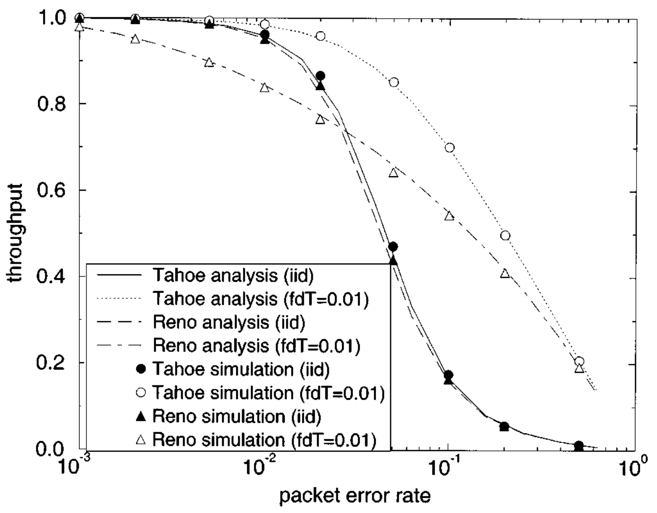


Fig. 2. Throughput performance of TCP Reno and Tahoe for i.i.d. and $f_dT = 0.01$. $W_{\max} = 24$. $K = 3$. $MTO = 100$. Analysis and simulation.

The methodology described in Section IV-C can be applied here to find performance bounds.

V. Results

We now compare the performance of TCP Tahoe, Reno, and NewReno. The OldTahoe version is not considered here because its performance is found to be significantly inferior to that of the other versions [5]. As a concrete example of a channel with memory, we consider the flat Rayleigh fading channel [14], whose error process is approximated by a two-state Markovian chain as described in [12] and [16]. In particular, the degree of memory of the channel directly depends on the normalized value of the Doppler bandwidth, f_dT, where f_d is the maximum Doppler shift [14] and T is the packet duration. For comparison, we also consider a memoryless channel.

The values of the normalized Doppler bandwidth, f_dT, considered are 0.01, 0.08, and 0.64. At a carrier frequency of 900 MHz and the considered packet duration of about 7.5 ms, an f_dT value of 0.01 corresponds to a user moving at a speed of about 1.5 km/h (pedestrian user), and an f_dT value of 0.64 corresponds to a user speed of about 100 km/h (vehicular user). When $f_dT = 0.01$, the fading process is very much correlated (e.g., average packet error burst length of 13 packets for $f_dT = 0.01$ and $P_E = 0.1$ in Table I) However, when $f_dT = 0.64$, the fading is fast and the performance is virtually the same as in the memoryless case (i.i.d. errors).

Fig. 1 shows the throughput of both Reno and Tahoe as a function of the marginal packet error probability P_E, for the two cases of $f_dT = 0.01$ and i.i.d. errors. A maximum advertized window size $W_{\max}$ of 6 packets, minimum timeout MTO of 100 packets, and a fast retransmit threshold K of 3 are used. Simulation points are also given, showing the accuracy of the analytical approach.

From Fig. 1 we note that for the chosen advertized window size of 6 packets, Reno performs better when packet errors are i.i.d. (i.e., mostly single packet errors) than correlated packet errors (e.g., $f_dT = 0.01$) at small values of P_E. This is because at low values of P_E, i.i.d. errors are dealt with effectively by Reno's congestion window adaptation algorithm, which is optimized for single packet errors.

In the presence of bursty packet errors, however, two conflicting effects influence the performance. For a given value of the average packet error rate, P_E, clustered errors correspond to fewer error *events* (each comprising multiple packet errors), and therefore the congestion window is shrunk less frequently than for i.i.d. errors. A second effect takes place depending on the relationship between channel error burstiness and size of the advertized window. In order to trigger a fast retransmit, $K = 3$ duplicate ACK's must be successfully received after a packet loss. If packet n is corrupted by the channel at time t, only $W(t) - 1$ more packets can be transmitted, with $W(t) - 1 \leq W_{\max} - 1$ (=5 in this case). Therefore, if $W(t) - K$ or more packets are also corrupted, then the congestion window will be exhausted before K duplicate ACK's can be generated. The transmitter will then stop and wait for a timer expiration, i.e., the fast retransmit feature is not triggered. This undesirable event is fairly likely if the channel error burstiness is comparable with the congestion window size.

As can be seen in Fig. 1, for high values of P_E, the beneficial effect overweighs the negative effect, resulting in better performance for correlated errors than for i.i.d. errors. Also the performances of Reno and Tahoe are almost identical at high P_E values. On the other hand, for small values of P_E, the effect of fast retransmit failures may dominate the performance, resulting in degraded performance due to error correlation. This effect can be significant if the channel error burstiness is comparable with the congestion window size (which cannot exceed $W_{\max}$).

For example, in Fig. 1, with a $W_{\max}$ value of 6 the correlation benefit is not fully exploited for $f_dT = 0.01$, where packet error bursts span 13 packets on average for $P_E = 0.1$ (as given in Table I). In this case, providing a $W_{\max}$ larger than 13 is beneficial. For Tahoe, this is clearly confirmed by the performance plots for $W_{\max} = 24$ in Fig. 2, where the $f_dT = 0.01$ case is found to perform significantly better than the i.i.d. case at all values of P_E. However, Reno performance is seen to be severely degraded at low P_E clustered errors, and significantly

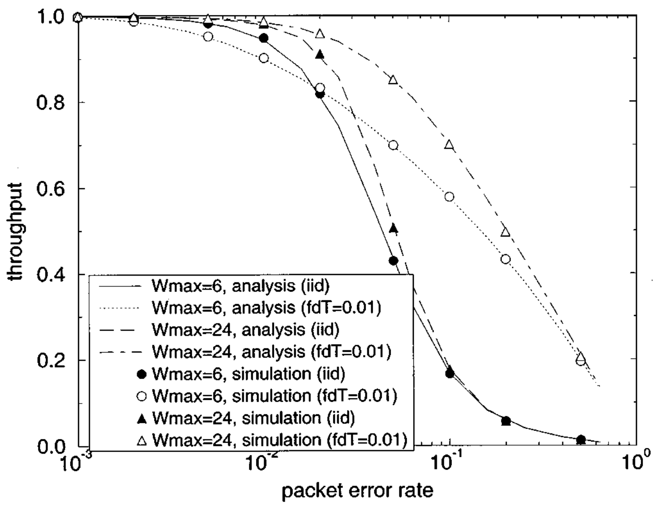


Fig. 3. Throughput performance of TCP NewReno for i.i.d. and $f_dT = 0.01$. $W_{\max} = 6$ and 24. $K = 3$. $MTO = 100$. Analysis and simulation.

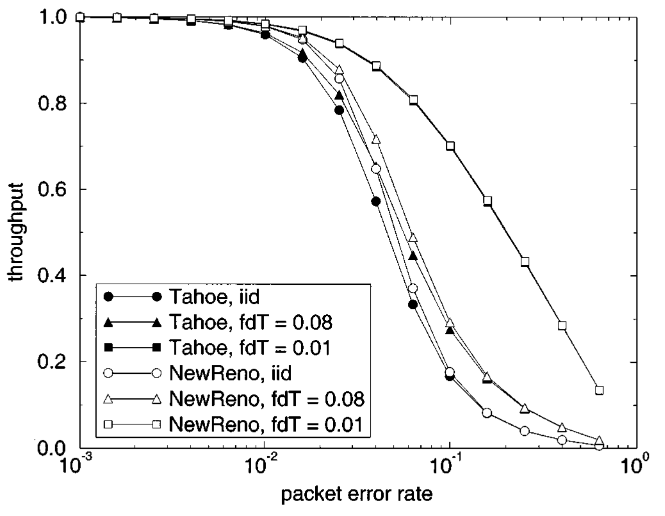


Fig. 4. Throughput comparison of TCP Tahoe and NewReno at different values of f_dT. $W_{\max} = 24$. $K = 3$. $MTO = 100$. Analysis only.

worse than Tahoe in general. This poor performance of Reno when errors are bursty (multiple errors) is in agreement with earlier *qualitative* predictions in [3].

Fig. 3 gives the performance of NewReno for i.i.d. errors as well as at $f_dT = 0.01$ for $W_{\max} = 6$ and 24, $K = 3$ and $MTO = 100$ (again, simulation checks have been performed and reported in the graph). As in the case of Tahoe, NewReno also benefits from correlation in packet errors ($f_dT = 0.01$ performance is better than i.i.d. performance), more so when $W_{\max}$ is large.

Since TCP Reno is not appropriate for use on correlated fading channels, it is therefore not considered further. A performance comparison of TCP NewReno with TCP Tahoe under different degrees of correlation in the fading process for $W_{\max} = 24$, $MTO = 100$, and $K = 3$, is given in Fig. 4 (for clarity, simulation marks are not included in this and the following figures). Since both Tahoe and NewReno follow loss recovery procedures different from that of Reno, they both exhibit substantial increase in the mean congestion window width at loss instants when $f_dT = 0.01$ compared to the i.i.d.

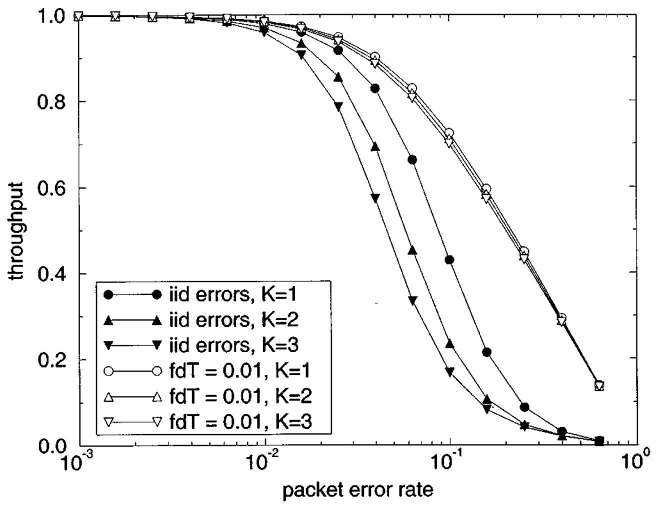


Fig. 5. Throughput performance of TCP Tahoe at different values of fast retransmit threshold, $K(1, 2, 3)$ at $f_dT = 0.01$ and i.i.d. $W_{\max} = 24$. $MTO = 100$. Analysis only.

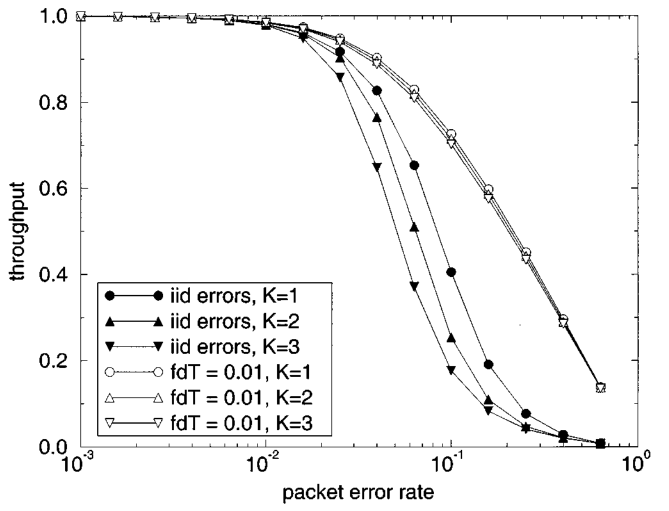


Fig. 6. Throughput performance of TCP NewReno at different values of fast retransmit threshold, $K(1, 2, 3)$ at $f_dT = 0.01$ and i.i.d. $W_{\max} = 24$. $MTO = 100$. Analysis only.

case. This essentially translates into increased throughput in slow fading compared to i.i.d. fading as can be seen from Fig. 4. Also, conforming to the results reported in [5], with i.i.d. packet errors, TCP NewReno is found to perform better than TCP Tahoe at low and medium values of P_E, because of its recovery optimization to single packet errors. At high values of P_E, however, Tahoe and NewReno exhibit similar performance. As in i.i.d. errors, NewReno is found to perform better than Tahoe in correlated errors as well, but only at large values of f_dT (e.g., $f_dT = 0.08, 0.64$). However, as the value of f_dT is decreased, the difference between NewReno and Tahoe diminishes. For example, when $f_dT = 0.01$ NewReno and Tahoe exhibit almost identical performance.

The effect of varying the fast retransmit threshold, K, on the Tahoe and NewReno versions of TCP for i.i.d. and $f_dT = 0.01$ is shown in Figs. 5 and 6. Throughput curves are plotted for $K = 1, 2, 3$. It is observed that, in i.i.d. errors, the performance with $K = 1$ is better than with $K = 3$. However, when

$f_dT = 0.01$, there is practically no benefit from going from $K = 3$ to $K = 1$, both for Tahoe and NewReno. Hence newer versions of TCP like TCP Vegas, which propose to retransmit at the instance of the receipt of the first duplicate ACK itself (i.e., $K = 1$), may not benefit from such modifications on highly correlated wireless fading channels. The use of selective ACK's (SACK TCP [3]) has been proposed as a more promising variation. However, it is worth noting that in the environment under consideration, where instantaneous feedback is assumed, SACK TCP and TCP NewReno behave identically, and therefore the results presented for TCP NewReno apply to SACK TCP as well.

As previously mentioned, the proposed analysis could be used to assess the energy consumption performance of various TCP versions. For example, by looking at Fig. 2 in the horizontal direction, one can note that Tahoe and Reno under correlated fading conditions achieve throughput 0.9 at average error rates which are about one order of magnitude apart. Since from Table I we see that a decrease of an order of magnitude of the average error rate corresponds to a 10 dB increase of the fading margin, these results indicate that in these conditions Tahoe may be ten times more energy efficient than Reno, which is a very significant result. The energy efficiency issue is addressed in detail in [19].

Finally, the impact of ACK delays and losses has also been investigated by ns simulation in order to assess the accuracy of our assumption of instantaneous and error-free feedback. What we found is that for ACK delays of up to 2 slots and for ACK error rates better than 0.01, there is essentially no difference in performance compared to the ideal case. In a local wireless environment, the roundtrip times are such that one or two slots cover all practical cases. Also, ACK's are usually protected against errors, so that assuming an error rate better than 0.01 does not seem too restrictive. For more general scenarios, including the important case of connections with large bandwidth-delay product, the analysis presented in this paper is to be extended.

VI. Summary

We proposed a new analytical approach to computing the performance of TCP, which enabled us to study the bulk throughput of TCP Tahoe, Reno, and NewReno over wireless fading links with memory. We considered a scenario where large blocks of data are to be transferred from a base station to a mobile terminal over a 1.5 Mbps wireless link having negligible bandwidth-delay product. We showed that, as long as sufficiently large advertized window sizes are used, the burstiness in the packet errors on the wireless link caused by slow multipath fading significantly affects the throughput performance of TCP compared to i.i.d. packet errors. More specifically, in correlated errors, TCP Tahoe and New Reno have larger throughput, whereas the throughput of TCP Reno (which is not recommended in this environment) may be significantly worse. We further showed that in such slow fading scenarios, NewReno performs no better than Tahoe in general, mainly due to the high degree of correlation in the fading process. Based on the results for different values of the fast retransmit parameter K, it may also be argued that other enhanced versions of TCP (such as Vegas) may not offer any significant improvement on highly correlated wireless fading channels. This fact, together with the observation that TCP performance depends significantly on the channel error correlation, leads us to our final conclusion. A clever design of the lower layers that preserves error correlations (naturally present on wireless links because of the fading behavior) could be an attractive alternative to the development or the use of more complex TCP algorithms. Results for the energy consumption performance of the protocols show that, unlike throughput, this metric may be very sensitive to the TCP version used and on the protocol parameters [19].

Topics of ongoing investigation include the study of the effect of using a link-layer FEC/ARQ scheme, and the effect of other physical layer parameters (e.g., packet size) on the overall TCP performance. Other performance metrics besides throughput (e.g., delay) are also being considered [23].

Appendix
Quantization of the Window Size

The window size at time n, denoted with Y, is a real number in general, due to the $1/W$ increment rule during the congestion avoidance phase. In order to select an appropriate set to represent the value of Y, observe that what matters for the future evolution of the protocol are the two quantities $\lfloor Y \rfloor$ (which counts how many more packets can be transmitted) and $\lceil Y/2 \rceil$ (which determines the value of the window size and window threshold after loss recovery).

Therefore, any two values of Y which result in the same values of the above two quantities do not need to be distinguished in Ω_Y. It can be seen that the following partition on the real numbers between 1 and $W_{\max}$ (the possible values of the window size) satisfies this constraint: $[1, 2); \{2\}; (2, 3); [3, 4); \{4\}, \cdots, (W_{\max} - 2, W_{\max} - 1)$, $[W_{\max} - 1, W_{\max}), \{W_{\max}\}$, where $W_{\max}$ has been assumed even. In this case, the size of Ω_Y is equal to $3W_{\max}/2 - 1$.

References

[1] W. R. Stevens, *TCP/IP Illustrated, Volume 1*. Reading, MA: Addison-Wesley, 1994.

[2] T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss," *IEEE/ACM Trans. Networking*, pp. 336–350, June 1997.

[3] K. Fall and S. Floyd, "Comparisons of Tahoe, Reno, and Sack TCP," manuscript, ftp://ftp.ee.lbl.gov, Mar. 1996.

[4] O. Ait-Hellal and E. Altman, "Analysis of TCP Vegas and TCP Reno," in *Proc. IEEE ICC'97*, 1997.

[5] A. Kumar, "Comparative performance analysis of versions of TCP in a local network with a lossy link," *ACM/IEEE Trans. Networking*, Aug. 1998.

[6] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *ACM/IEEE Trans. Networking*, Dec. 1997.

[7] A. DeSimone, M. C. Chuah, and O. C. Yue, "Throughput performance of transport layer protocols over wireless LANs," in *Proc. IEEE Globecom'93*, Dec. 1993, pp. 542–549.

[8] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global internet," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 1465–1480, Oct. 1995.

[9] R. Caceres and L. Iftode, "Improving the performance of reliable transport protocols in mobile computing environments," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 850–857, June 1995.

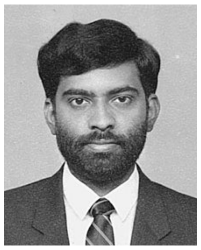
[10] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP performance over wireless networks," in *Proc. 1st Int. Conf. Mobile Computing Networking*, Nov. 1995.

[11] P. Manzoni, D. Ghosal, and G. Serazzi, “Impact of moblity on TCP/IP: An integrated performance study,” *IEEE J. Select. Areas Commun.*, vol. 13, pp. 858–867, June 1995.
[12] A. Kumar and J. Holtzman, “Comparative performance analysis of versions of TCP in a local network with a lossy link—Part II: Rayleigh fading mobile radio link,” Tech. Rep. WINLAB-TR-133, Nov. 1996.
[13] P. Bhagwat, P. Bhattacharya, A. Krishna, and K. Tripathi, “Using channel state dependent packet scheduling to improve TCP throughput over wireless LANs,” *Wireless Networks*, pp. 91–102, 1997.
[14] W. C. Jakes, Jr., *Microwave Mobile Communications*. New York: Wiley, 1974.
[15] M. Zorzi and R. R. Rao, “Effect of correlated errors on TCP,” in *Proc. 1997 CISS*, Mar. 1997, pp. 666–671.
[16] M. Zorzi, R. R. Rao, and L. B. Milstein, “On the accuracy of a first-order Markov model for data transmission on fading channels,” in *Proc. IEEE ICUPC’95*, Nov. 1995, pp. 211–215.
[17] TIA/EIA/IS-99, “Data services option standard for wideband spread spectrum digital cellular system,”, 1995.
[18] A. Chockalingam and G. Bao, “Performance of TCP/RLP protocol stack on correlated fading DS-CDMA wireless links,” in *IEEE Trans. Veh. Tech.*, vol. 49, Jan. 2000, pp. 28–33.
[19] M. Zorzi and R. R. Rao, “Energy efficiency of TCP,” Mobile Networks Appl., submitted for publication.
[20] V. Jacobson, “Congestion avoidance and control,” in *Proc. ACM Sigcomm’88*, Aug. 1988, pp. 314–329.
[21] R. A. Howard, *Dynamic Probabilistic Systems*. New York: Wiley, 1971.
[22] M. Zorzi and R. R. Rao, “Energy constrained error control for wireless channels,” *IEEE Personal Commun. Mag.*, vol. 4, pp. 27–33, Dec. 1997.
[23] A. Chockalingam, M. Zorzi, and V. Tralli, “Wireless TCP performance with link layer FEC/ARQ,” in *Proc. IEEE ICC’99*, June 1999.
[24] M. Zorzi and R. R. Rao, “Lateness probability of a retransmission scheme for error control on a two-state Markov channel,” *IEEE Trans. Commun.*, vol. 47, Oct. 1999.

Michele Zorzi (S’89–M’95–SM’98) was born in Venice, Italy, in 1966. He received the Laurea degree and the Ph.D. degree in electrical engineering from the University of Padova, Italy, in 1990 and 1994, respectively.

During the academic year 1992/1993, he was on leave at the University of California, San Diego (UCSD), attending graduate courses and doing research on multiple access in mobile radio networks. In 1993, he joined the faculty of the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy. After spending three years with the Center for Wireless Communications at UCSD, in 1998, he joined the School of Engineering of the Università di Ferrara, Italy, where he is currently an Associate Professor. His present research interests include performance evaluation in mobile communications systems, random access in mobile radio networks, and energy constrained communications protocols.

Dr. Zorzi currently serves on the Editorial Boards of the IEEE PERSONAL COMMUNICATIONS MAGAZINE and of the *ACM/URSI/Baltzer Journal of Wireless Networks*. He is also guest editor for special issues in the IEEE PERSONAL COMMUNICATIONS MAGAZINE (Energy Management in Personal Communications Systems) and the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (Multi-media Network Radios).



A. Chockalingam (S’92–M’95–SM’98) was born in Rajapalayam, Tamilnadu State, India. He received the B.E. (Honors) degree in electronics and communication engineering from the P. S. G. College of Technology, Coimbatore, India, in 1984, the M.Tech. degree with specialization in satellite communications from the Indian Institute of Technology, Kharagpur, India, in 1985, and the Ph.D. degree in electrical communication engineering (ECE) from the Indian Institute of Science (IISc), Bangalore, India, in 1993.

During 1986 to 1993, he worked with the Transmission R&D division of the Indian Telephone Industries Ltd., Bangalore. From December 1993 to May 1996, he was a Postdoctoral Fellow and an Assistant Project Scientist at the Department of Electrical and Computer Engineering, University of California, San Diego (UCSD), where he conducted research in DS-CDMA wireless communications. From May 1996 to December 1998, he served Qualcomm, Inc., San Diego, CA, as a Staff Engineer/Manager in the systems engineering group. In December 1998, he joined the faculty of the Department of ECE, IISc, Bangalore, India, where he is an Assistant Professor, working in the area of wireless communications, and directing research at the Wireless Research Lab (WRL), IISc. He was a visiting faculty to UCSD during summer 1999. His research interests lie in the area of DS-CDMA systems and wireless networks and protocols.

Dr. Chockalingam is a recipient of the CDIL (Communication Devices India Ltd.) award and Prof. S. K. Chatterjee research grant award.

Ramesh R. Rao (SM’90) was born in Sindri, India, in 1958. He received the Honors Bachelor’s degree in electrical and electronics engineering from the University of Madras in 1980. He did his graduate work at the University of Maryland, College Park, Maryland, receiving the M.S. degree in 1982 and the Ph.D. degree in 1984.

Since then he has been on the faculty of the Department of Electrical and Computer Engineering at the University of California, San Diego (UCSD). His research interests include architectures, protocols, and performance analysis of computer and communication networks. He is currently the Director of UCSD Center for Wireless Communications.

Dr. Rao is the Editor for Packet Multiple Access of the IEEE TRANSACTIONS COMMUNICATIONS. He was the Guest Editor of the JSAC special issue on Multimedia Network Radios. He is currently serving his second term as a member of the IEEE Information Theory Society Board of Governors.