Accurate Analysis of TCP on Channels With Memory and Finite Round-Trip Delay

Michele Rossi, *Member, IEEE*, Raffaella Vicenzi, and Michele Zorzi, *Senior Member, IEEE*

***Abstract*—In this paper, we present an accurate analytical model for transport control protocol (TCP) over correlated channels (e.g., as induced by multipath fading) taking into account a finite round-trip delay. In particular, we develop models and analysis for studying four versions of TCP, namely, Old Tahoe, Tahoe, Reno, and New Reno. We focus on a single wireless TCP connection by modeling the correlated packet loss/error process as a Discrete Time first-order Markov chain. Our model explicitly incorporates important aspects such as *slow start, congestion avoidance, fast retransmit* and *fast recovery*. The main findings of this study are that: 1) an increasing round-trip time may significantly affect the throughput performance of TCP, especially when an independent channel is considered; 2) New Reno performs better than Reno and Tahoe when the channel is uncorrelated, whereas Tahoe's recovery strategy is the most efficient when the channel correlation is high; and 3) the maximum window size does not play a determinant role in increasing throughput performance in both correlated and independent channels. While some of these conclusions confirm what other authors have observed in simulation studies, our analytical approach sheds some new light on TCP's behavior.**

***Index Terms*—Bursty channel, congestion control algorithm, correlated packet losses, performance analysis, TCP/IP modeling.**

I. Introduction

THE INCREASING popularity of wireless networks indicates that wireless links will play an important role in the future internetworking. transport control protocol (TCP) is a reliable, end-to-end, transport protocol that is widely used to support applications like *e-mail*, *telnet*, *ftp*, and *http*. TCP was designed primarily for wireline networks where packet losses are caused mainly by congestion. Several modifications have been proposed to the TCP loss-recovery and congestion-control mechanism to improve data throughput, including Reno, New Reno, and Vegas.

In the recent literature, many papers have appeared on the topic of TCP performance in wireless systems. Some of those papers have as their main goal to describe new behaviors, usually observed from results obtained by simulation. Some others are more concerned with modeling TCP operation by means of analytical techniques, in order to provide more general tools and to develop more accurate descriptions while controlling the complexity. The goal of this paper is to provide an accurate analytical model for TCP operation.

Manuscript received May 24, 2002; revised October 19, 2002; accepted February 11, 2003. The editor coordinating the review of this paper and approving it for publication is Y.-B. Lin. This work has been supported in part by ERICSSON Research.

The authors are with the Department of Engineering, University of Ferrara, 44100 Italy (e-mail: mrossi@ing.unife.it; rvicenzi@ing.unife.it; mzorzi@ing.unife.it).

Digital Object Identifier 10.1109/TWC.2004.825360

The analytical characterization of TCP has been already investigated in previous studies. In [1], for example, the TCP throughput is obtained for large round-trip delays, but this paper focuses on channel described by means of an independent and identically distributed (i.i.d.) error process. In [2], Hellal *et al.* studied the behavior of TCP Reno and Vegas considering an error-free wireline network, where a bottleneck link is the only cause of packet losses. On the other hand, Padhye *et al.* [3], have derived a useful formula for the TCP Reno throughput over wireline networks, considering the error process as i.i.d. In [3], the correlation among losses is tracked assuming that, considering a window of transmitted packets, whenever a packet is lost, all the other packets in the same window are lost as well (we refer to this model with the term *quasi-static channel*). All these papers consider a *static* (or *quasi-static*) channel and a wired network. A correlated channel has been considered in [4] and in [5], where TCP is studied over a wireless link and a two-state Markov model is used to describe the channel burstiness. An instantaneous feedback is considered, i.e., the acknowledge (ACK) message is received immediately after the completion of the packet transmission. This idealized model was adopted to limit the analytical complexity of the approach. This simplification leads to overestimating the TCP performance as the bandwidth-delay product increases.

In [6], a TCP mean throughput analysis over finite round-trip delay channels is reported considering both independent and correlated losses. In that paper, the delay between the instant where a packet loss occurs and the instant in which it is detected by the TCP sender is neglected. This assumption leads to accurate results where the round-trip time (RTT) is small. However, this is not true for large RTTs, e.g., as the ones envisioned in 3G cellular networks. In these scenarios end-to-end TCP RTTs can grow up to RTT$= 0.5$ s and the delay between error events and error detection can be estimated as RTT$+3\mu$, where μ is the TCP packet transmission time. Moreover, in [6] it is assumed that the congestion window restarts from 1 after multiple timeout events and that, in the correlated error case and when the timeout timer-expiry period is greater than the bad state duration, the error event is detected by the Fast Recovery algorithm with probability one (without entering timeout). In [7], the TCP throughput performance in correlated channels with finite round-trip delay is also investigated. This analysis enables qualitative and meaningful comparison between different schemes, but the accuracy of the model is not addressed. The distribution of the number of consecutive packet drops in a bad state is imposed, regardless of the time duration between the transmission of these packets. This assumption is only valid when packet transmission is continuous. However, TCP is a bursty protocol that introduces (sometimes considerables) intermittent idle durations during the lifetime of a connection. This assump-

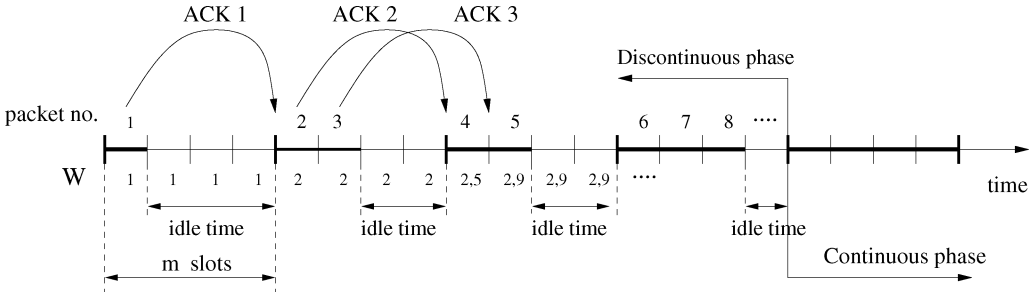

Fig. 1. Discontinuous transmissions during TCP slow-start phase.

tion holds when the window size is always equal to or greater than the bandwidth delay product of the link. This is not true in general and the assumption above in these cases may lead to inaccurate results.

The merit of the present paper is to give a very accurate analytical characterization of the TCP throughput behavior over wireless links characterized by a finite round-trip delay. The analysis we propose uses a semi-Markov chain to find throughput performance. However, it is worth noting that the analysis has the potential to be extended to find energy consumption metrics as well. To the best of our knowledge, our analysis is more accurate than any other analysis previously proposed in the literature. All the simplifying assumptions discussed above have been removed in the model presented in this paper, by constructing a semi-Markov chain able to jointly track window evolution, successes, transmissions and underlying channel state. Moreover, the assumption made for analytical tractability by previous works (see [5] and [6] as examples) that a timeout timer is associated with each TCP packet has been removed. Instead, a single timeout is considered for the whole connection (as in a real TCP data transfer).

In order to take into consideration the bursty nature of the channel, a two-state discrete time Markov channel (DTMC) is considered, as in [5]. The time is slotted, the single slot duration is equal to the TCP packet transmission time and the channel process (DTMC) evolves slot by slot according to its transition probabilities. Moreover, we extend the work presented in [5] considering a noninstantaneous feedback, i.e., ACKs arrive exactly $m-1$ slots after the packet transmission, where m is the RTT value. The introduction of this feedback delay leads to what we call a "discontinuous transmission phase" (see Fig. 1). In fact, after the transmission of the first packet of the connection, the TCP sender enters an idle time waiting for the reception of an ACK message. When this ACK arrives, the TCP window size, W, is incremented by one (*slow-start phase*), allowing the receiver to transmit two consecutive packets. According to the protocol rules, W is incremented by one segment for each new received ACK until W reaches the *slow-start threshold*, $W_{\rm th}$. When W becomes larger than $W_{\rm th}$ the sender enters the so called *congestion-avoidance* phase and W is incremented by $1/W$ for each received ACK.[1]

During this phase, TCP transmits packets in bursts of length equal to W and enters an idle time waiting for the ACK reception. This discontinuous phase finishes when W reaches the round-trip value, i.e., when W becomes equal to m. It is important to note that the assumption of an RTT greater than zero implies that the TCP sender takes more time to fill the bandwidth-delay product of the link, since the window grows more slowly, and this may significantly affect the TCP performance.

The main findings of this study are that; 1) an increasing RTT may significantly affect the throughput performance of TCP, especially when an independent channel is considered; 2) New Reno performs better than Reno and Tahoe when the channel is uncorrelated, whereas the Tahoe recovery strategy is the most efficient when the channel correlation is high; and 3) the maximum window size does not play a determinant role in increasing throughput performance in both correlated and independent channels. While some of these conclusions confirm what other authors have observed in simulation studies, our analytical approach sheds some new light on TCP's behavior.

This paper is organized as follows. In Section II, the channel and the TCP models are presented in detail and Section III reports the analysis. Results are shown in Section IV and in Section V, some conclusions are given.

II. System Model

A TCP session involves three phases that are the connection setup phase, the data transfer phase and the connection tear-down phase. Since we are primarily interested in the bulk throughput performance of TCP, in this paper, we consider only the data transfer phase because it is the one that dominates the overall performance. We consider a pair of nodes, the sender and the receiver, exchanging data over a dedicated link, characterized by a two-state Markov packet error process, finite propagation delay and perfect feedback. It is assumed that the transmitter always has an infinite supply of packets to send (*Heavy Traffic* assumption). As usually done in studies taking an analytical approach, we focus on a single TCP connection between a wireless terminal and a terminal placed in the terrestrial network. In this scenario, TCP packets experiences an RTT (m) that is given by the sum of the contributes due to the wireless and the terrestrial network. We consider the wired link to be error-free, while we model the errors over the wireless channel by means of the Markov model introduced above. A detailed description of such model will be given in Section II-B.

A. TCP Algorithms Description

In this section, we first describe the transmitter and receiver processes in TCP OldTahoe, Tahoe, Reno and New Reno. In the following, we proceed with our description of the various

[1]In Fig. 1 $W_{\rm th}$ is set to 1 segment.

TCP algorithms. The TCP receiver is common for all TCP versions, while the TCP transmitter depends on the version considered. *The receiver* accepts packets out of sequence, but will only deliver them in sequence to the user's application. Moreover, the receiver sends back one ACK for every packet correctly received (no-delayed ACK receiver).[2] The ACKs are cumulative, that is, an ACK carrying the sequence number i ACKs all data packets up to, and including, the packet with sequence number $i-1$. Each ACK will identify the next expected packet sequence number, which is the first among the packets necessary to complete the in-sequence delivery. Hence, when a packet is lost, the transmitter keeps receiving the so called duplicate ACKs, that is, the ACK with the sequence number of the first packet lost.

The TCP transmitter operates using a sliding window-based strategy as follows. At any given time t, the TCP transmitter maintains the status variables $A(t)$, $W(t)$ and $W_{\rm th}(t)$. The lower window edge, $A(t)$, represents all data numbered up to and including $A(t)-1$ that has been transmitted and acknowledged. This variable is nondecreasing, and for each received ACK with sequence number $n > A(t)$, $A(t)$ increases up to n. The congestion window, $W(t)$, defines the maximum number of unacknowledged packets the transmitter is allowed to send starting from $A(t)$. At any time $W(t)$ is limited by its maximum value $W_{\max}$.[3] The slow-start threshold triggers the increment of $W(t)$ to realize the flow control imposed by the sender, i.e., to slow-down or speed-up the transmission rate of packets into the network. In particular, if $W(t) < W_{\rm th}$, each ACK causes $W(t)$ to be incremented by one. This is called the *slow-start* phase. On the other hand, if $W(t) \geq W_{\rm th}$, $W(t)$ is incremented by $1/W(t)$ every time a new ACK is received. This phase is called *congestion avoidance*.

Each TCP connection has a timeout timer that is updated at each round-trip delay. If this timeout timer expires, i.e., no more ACKs are received, then a retransmission occurs. To maintain the analytical tractability we assume this timer to be fixed at a constant value, T_o.[4] Calling t the time in which this happens, $W(t^+)$ and $W_{\rm th}(t^+)$ are set to 1 and $W(t)/2$ respectively. In this way the connection is restarted in slow-start phase. Note, that there exists another flow control parameter imposed by the receiver called receiver advertised window $awnd$; the protocol rules impose that at any time t the transmitter is allowed to inject into the network a number of unacknowledged packets equal to the minimum between $W(t)$ and $awnd$. By setting $awnd$, the receiver can slow-down the sender transmission rate when it is not able to read data at the same speed at which the sender is transmitting.

When a loss occurs, *TCP-OldTahoe* [9] uses only the above explained timeout recovery mechanism.

TCP-Tahoe [8], [10] instead, implements a further recovery technique called *fast retransmit*. When a packet is lost and when the transmitter gets the Kth duplicate ACK[5] at time t, it behaves as if a timeout had occurred and begins retransmission setting $W(t^+)$ and $W_{\rm th}(t^+)$ as above.

In the case of *TCP-Reno* [8], [11], [12], the Fast Retransmit procedure is implemented as in Tahoe, but the subsequent recovery phase is different. In more detail, with the reception of the Kth dupACK at time t, $W_{\rm th}(t^+) = W(t)/2$ and $W(t^+) = W_{\rm th}(t^+)+K$. At this point, the Reno transmitter transmits only the first lost packet (with sequence number equal to $A(t)$, this is the Fast Retransmit) and, as it waits for the ACK it may get dupACK due to the outstanding packets. For each dupACK W is inflated by one, and a packet is transmitted if allowed by A and the new value of W. This is the Fast Recovery algorithm. Finally, in t_f, when the new ACK is received the loss recovery phase finished and $W(t_f)$ is set to $W_{\rm th}(t_f) = W_{\rm th}(t^+)$. The Reno's loss recovery does not work for multiple losses in a window of packets [13], for example, when three packets are dropped from a window of data, the sender is forced to wait for a timeout whenever the number of packets between the first and the second dropped packet is less than $2+3W/4$, where W is the congestion window just before the Fast Retransmit. Note that, the loss recovery strategy in Reno performs better than Tahoe when single packet losses occur in the loss window (the window in which we have the first loss).

In the *New-Reno* algorithm [14], when K duplicate ACKs are received, the Fast Retransmit is applied as in Reno, but the loss recovery procedure is different. The highest sequence number transmitted before the loss is recorded in a variable named *recover*. The Fast Recovery phase differs from Reno when an ACK acknowledging new data, arrives. In fact, this ACK could be the acknowledgment elicited by the first retransmission, or elicited by a later retransmission. If this ACK acknowledges all the packet up to, and including *recover*, then the Fast Recovery procedure is completed as for the Reno case. On the other hand, this ACK is labeled as *partial new ACK*. In this case, the first unacknowledged packet is retransmitted, W is deflated by the amount of new data acknowledged (also A is updated accordingly) and then is incremented by one. So, a new packet is transmitted if allowed by the new value of A and W. The Fast Recovery is completed when the sender receives an ACK up to and including *recover*. The New Reno algorithm can recover from multiple losses in some cases.

Unlike the assumption in [5], where a timer is associated to every packet transmitted, in this paper we consider only one timeout timer characterizing the connection, as in a real TCP protocol [8], [15]. This assumption allows us to correctly track the timeout whenever a partial new ACK is received, that is, when a partial new ACK is received the timeout timer is restarted (see [15]). In [5], due to the assumption of instantaneous feedback, the numerical results were not so different from the ones relative to the real TCP protocol. However, when a finite round trip is considered, performance is significantly affected, so it is important to correctly track the timeout event. Fig. 2 shows how the timer is rescheduled during the New Reno fast recovery phase when both a single timer and a timer per packet is used. Note that, when the recovery phase fails, and the second case is considered, the timeout timer expires at least $2m$ slots earlier.

[2]In this paper, we consider a no-delayed ACK receiver, even if in real TCP implementations the delayed ACK scheme [8] is used.

[3]This value can be negotiated with the receiving entity during the connection, however, in this paper we consider it as a constant value.

[4]We do not explicitly model the timeout back-off mechanism. The accuracy of this simplifying assumption has been verified by simulation. Also, note that it is possible to let the timeout depend on the RTT, m, with no modification to the analysis we present here.

[5]K is a system parameter usually set to 3.

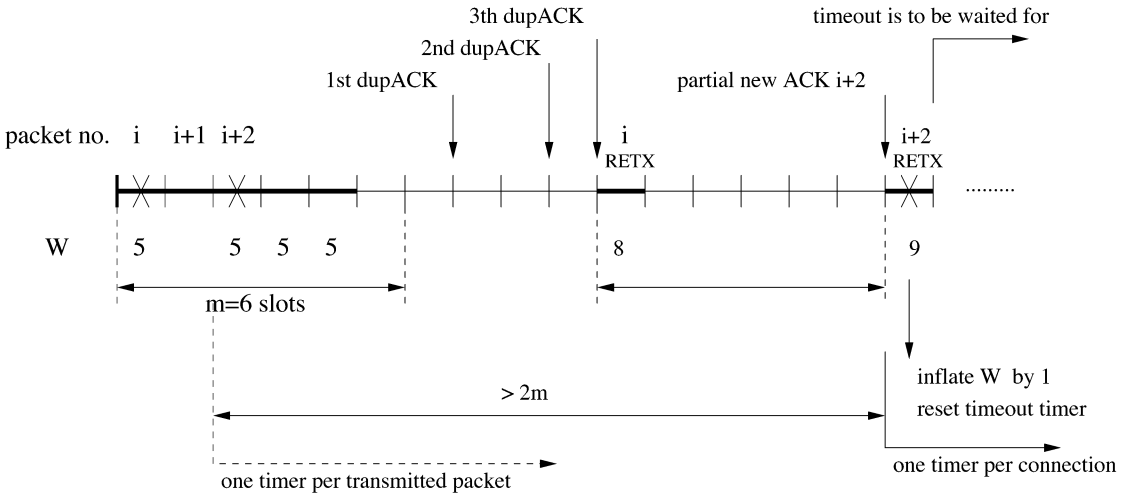

Fig. 2. Scheduling of the timeout event in the New Reno fast recovery phase.

B. Channel Model

We model the correlated packet error process using a discrete-time first-order Markov model as proposed in [16]. The pattern of errors is described by the transition matrix

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

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. Note that $1/(1-p_{BB})$ represents the average length of a burst of errors, which is described by a geometric *r.v.*

Given the matrix $\mathbf{M}$, the channel properties are completely characterized. In particular, it is possible to find the average slot error probability, [6] $\varepsilon = p_{GB}/(p_{GB}+p_{BG})$, that turns out to be dependent on the *fading margin*, indicated with F, (that express the physical characterization of the channel), and the *normalized Doppler bandwidth product*, f_dT, where T is the packet duration [16]. By choosing different ε 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, on the other hand, for higher values of f_dT, successive samples of the channels are almost independent. In Table I the fading margin F, the transition probabilities p_{GG} and p_{BB} and the average burst length $(1-p_{BB})^{-1}$ are reported for several values of f_dT and ε.

A detailed analysis of the packet loss with memory is presented and the case of i.i.d. errors is also considered for comparison.[7]

TABLE I
MARKOV MODEL PARAMETERS FOR DIFFERENT VALUES OF ε AND f_dT

f_dT	ε	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

III. Analytical Approach

The analysis is based on a Markov/renewal reward approach. The joint evolution of the window parameters and the channel state can be tracked by a random process $X(t) = (C(t-1), W_{\text{th}}(t), W(t))$, where $W(t)$ and $W_{\text{th}}(t)$ are the window size and the slow-start threshold in slot t, respectively, and $C(t-1)$ is the channel state in slot $t-1$ (bad, B, or good, G, corresponding to an erroneous or correct transmission, respectively). 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 $X(t)$ (such as the number of outstanding packets). Following the approach proposed in [5], we sample the process at appropriate instants t_k. In particular, by choosing as sampling instants the slots immediately following those in which either a timeout timer expires or a loss recovery phase is successfully completed, we obtain a process $X(k) = X(t_k)$ which is Markov. In fact, immediately after a timeout event, the window size shrinks to 1 and, from the point of view of the window adaptation algorithm, no outstanding packets are present. Therefore, at these instants, knowledge of $(C(t-1), W_{\text{th}}(t), W(t))$ is all there is to know to characterize the window/channel evolution in the future. Likewise, in the instant immediately after the slot in which the loss recovery phase was successfully completed, by definition, all outstanding packets have been acknowledged. Again, $(C(t-1), W_{\text{th}}(t), W(t))$ is all we need to know to characterize the future evolution of the window/channel. We observe that, from the protocol rules, the value of $W(t_k)$ can only be equal to 1 (timeout case) or to $W_{\text{th}}(t)$ (successful loss recovery). Finally, note that unlike in [5], the channel state at time $t_k - 1$ can be either erroneous or correct not only in the timeout case, but also for a successful loss

[6]This is the steady-state probability that the channel is in bad state in a time slot, and is not the same as the error probability experienced by transmitted packets because TCP's window adaptation tries to avoid bad channel conditions. The analytical approach developed in the following takes this into account.

[7]This case is obtained from the correlated channel by imposing $p_{BB} = p_{GB} = \varepsilon$.

recovery. In fact, even if the loss recovery phase must be ended by a a successful transmission, we receive the relative ACK after an RTT. Therefore, the state space of the process $X(k)$ is given by

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

where the first set correspond to timeout and second set corresponds to successful recovery phase. Note that, the total number of states[8] is in this case $4\lceil W_{\max}/2 \rceil - 2$.

In order to evaluate metrics of interest, such as throughput, the above information is not sufficient, so we consider a semi-Markov process, which admits $X(k)$ as its embedded Markov chain. We label transitions of the chain $X(k)$ with transition metrics, which track the 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.

A. Semi-Markov Analysis

Let us define as *cycle* k the time evolution of the system between two consecutive sampling instant t_k and t_{k+1}, where t_k is the kth sampling instant determined as explained in the previous section. 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, so the system state is given by $X(k) = (C(t_{k-1}), W_{\rm th}(t_k), W(t_k)) \in \Omega_X$, where Ω_X is the set of all possible values of $X(k)$ (state space of the sampled process). For simplicity of notation in the following we indicate $C(t_{k-1})$ as C. Also, we assume that the process is stationary, so that all statistics are independent of k. Let $n \ge 1$ be the first slot of the cycle to contain an erroneous transmission. Because of the assumption of a not instantaneous ACK message, the TCP packet transmission in each cycle can be discontinuous, so, to compute the distribution probability of n, we must take in account the only slots in which transmissions actually occur. Conditioned on the channel state in slot $t_k = 0$, the probability to have the first transmission error in slot $t = n$ is 0 when the sender is idle, whereas, when a transmission is allowed, it is given by

$$\alpha_C(n) = \mathrm{P}\left[\text{first error at } t = n | X(k) = (C,\ W_{\rm th},\ W)\right] = \begin{cases} p_{CB}, & n=1 \\ p_{CG} p_{GG}(n_1) p_{GG}(n_2) \ldots p_{GB}(n_l), & n>1 \end{cases} \quad (3)$$

where $\sum_{i=1}^{l} n_i = n-1$ and $p_{GG}(n_i)$, [or $p_{BG}(n_i)$], represents the probability that the transmission in slot k is successful given that the transmission in slot $k - n_i$ was successful (unsuccessful). Note that the probability to have the first error at time n given the channel state at time 0 ($\alpha_C(n)$) is a deterministic quantity once C, $W_{\rm th}$ and W have been fixed. In fact, $W_{\rm th}$ and W are all we need to know to determine the error-free protocol evolution (transmission and window processes). Hence, these processes can be easily tabulated. In order to explain the meaning of (3) consider the situation depicted in Fig. 3, in which we have assumed an RTT of $m = 4$ slots, $W_{\rm th} = 1$ and $W = 1$. In this case $\alpha_C(n) = p_{CG} p_{GG}(n_1) p_{GG}(n_2) p_{GG}(n_3) p_{GG}(n_4) p_{GB}(n_5)$, $n = 13$, where $p_{GG}(n_1)$ is the n_1-step probability ($n_1 = 4$), that is the probability that slot 5 is successful given that slot $5 - n_1$ was successful. The other entries in the expression are obtained similarly. For a given C, $W_{\rm th}$ and W, the transmission evolves following the protocol rules up to and including time n. Therefore, we can easily tabulate the transmission mask that represents the sequence of slots in which a transmission actually occurs. Moreover, $\alpha_C(n)$ is directly derived from this mask as previously described.

[8]States $(C, 1, 1)$, $C = G, B$ are contained in both sets.

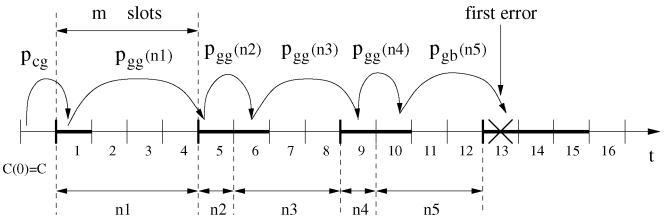

Fig. 3. Distribution probability of the first error at slot n.

Let $Y(k)$ be the system state at time n in cycle k and let Ω_Y be the set of all possible values of $Y(k)$. Note that, at time n there are a number of outstanding packets (packets *in flight*) that depends on $W(n)$, $W_{\rm th}(n)$ and n. Also, by definition we know that the channel state at that time is B. To determine the state at the beginning of cycle $k+1$ the slow-start threshold plays an important role. In fact, the exact number of packets that the sender is allowed to transmit after the first error, (n_{tx}), depends on the number of packets in flight at time n. In particular, during the recovery phase, the TCP window size is further incremented by the number of incoming ACKs elicited by these packets. Therefore, defining $W_{\rm fin}$ as the final value reached by W and $n_{\rm acks}$ as the number of incoming ACKs, we can simply derive n_{tx} as $\lceil W_{\rm fin} \rceil - 1$, since the error at time n always consumes one window unit. Referring to slot n, $n_{\rm acks}$ is obtained as follows:

$$n_{\rm acks} = \begin{cases} \min\{\lceil W(n) \rceil - 2, m-1\}, & W(n) \le W_{\rm th}(n) \\ \min\{\lceil W(n) \rceil - 1, m-1\}, & W(n) > W_{\rm th}(n) \end{cases} \quad (4)$$

where $W(n)$ represents the maximum number of outstanding (unacknowledged transmitted) packets at time n. Since the value of the RTT is m, and the transmission before the error event is assumed to be error-free, $\min(\lceil W(n-1) \rceil, m)$ gives the number of packets transmitted in the round preceding slot n. In (4), we take in the minimum $m-1$ instead of m because we are evaluating the number of ACKs received after the error. In more detail, m is the maximum number of packets sent in the round preceding slot n and so it represents the maximum number of incoming ACKs that can be received from slot n onward; with $m-1$ we account for the ACK received in slot n. The two terms $\lceil W(n) \rceil - 2$ and $\lceil W(n) \rceil - 1$ in (4) are an estimate of $\lceil W(n-1) \rceil - 1$, and are chosen according to the value of $W(n)$ and $W_{\rm th}(n)$. If $W(n) \le W_{\rm th}$, we are in the situation depicted in Fig. 4(a), where the window size in the previous round is equal to $W(n) - 1$ and the ACK relative to the first of the $\lceil W(n) \rceil - 1$ packets is the one that determines the increment of W at time n. In this case the number of ACKs received after the error event is equal to $W(n) - 2$. The situation in which $W(n) > W_{\rm th}(n)$ is analogous [see Fig. 4(b)], but here at slot n, W is incremented by $1/W(n-1)$, so $\lceil W(n) \rceil$

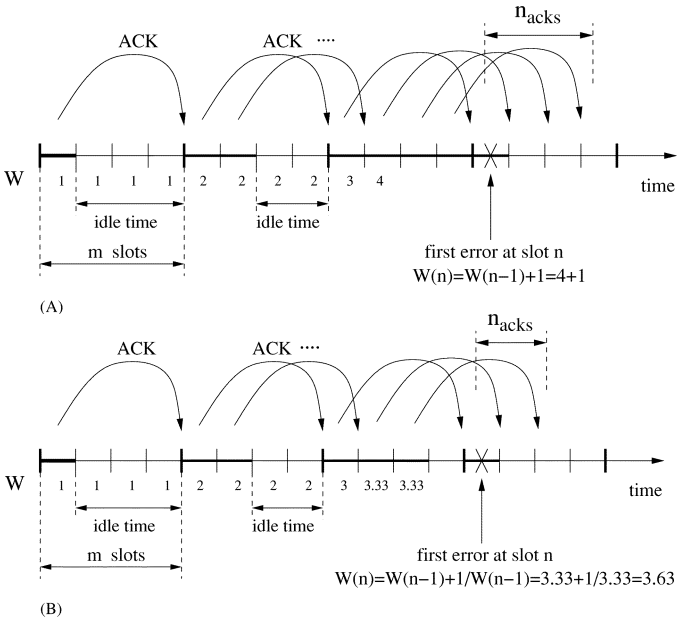

Fig. 4. Number of ACKs received after the first error: (a) slow-start case, $W(1) = 1$ and $W_{\rm th}(1) = 5$. (b) congestion-avoidance case, $W(1) = 1$ and $W_{\rm th}(1) = 3$.

usually remains equal to $\lceil W(n-1) \rceil$. Note that this last assumption is not always true, since some values of $W(n-1)$ could exist such that the above condition is not verified, i.e., $\lceil W(n-1) + 1/W(n-1) \rceil \neq \lceil W(n-1) \rceil$. In this case, to exactly derive $W(n-1)$ from the knowledge of $W(n)$, the state space Ω_X would have to be further expanded, but this would lead to such a big state space to make the problem analytically intractable. For this reason, when $W(n) > W_{\rm th}(n)$, we approximate $n_{\rm acks}$ as $\lceil W(n) \rceil - 1$. However, we have verified by simulation that our assumption has a negligible effect on the performance results for a wide range of the system parameters and is, therefore, entirely acceptable in general. Once computed n_{ack}, $W_{\rm fin}$ is derived in a straightforward way from the knowledge of $W(n)$ and $W_{\rm th}(n)$ by using the TCP protocol window incrementing rules. Specifically, $W_{\rm fin}$ is computed by using the following:

$$W_{\rm fin} = r_{n_{\rm acks}} \tag{5}$$

where

$$\begin{aligned} r_0 &= W(n) \\ r_i &= r_{i-1} + f(r_{i-1}) \quad i \geq 1 \end{aligned} \tag{6}$$

$$f(x) = \begin{cases} 1, & x < W_{\rm th} \\ \frac{1}{x}, & x \geq W_{\rm th} \end{cases}. \tag{7}$$

We can observe that, in our model, $Y(k)$ is described by $(W_{\rm th}(n),\ W(n))$, where $1 \leq W_{\rm th}(n) \leq \lceil W_{\max}/2 \rceil$ and $1 \leq W(n) \leq W_{\max}$.

As described in [5], we can separate the system evolution in two parts. The first part, in which the system makes a transition from state $X(k) \in \Omega_X$ to a state $Y(k) \in \Omega_Y$, is characterized by the transition matrix $\mathbf{\Phi}^{(1)}(z)$; whereas the second part, in which the system makes a transition from state $Y(k) \in \Omega_Y$ to a state $X(k) \in \Omega_X$, is characterized by the transition matrix $\mathbf{\Phi}^{(2)}(z)$. The statistic of the cycle is fully described by the matrix

$$\mathbf{\Phi}(z) = \mathbf{\Phi}^{(1)}(z)\mathbf{\Phi}^{(2)}(z) \tag{8}$$

whose entries are the transition functions associated to transitions from Ω_X to itself. In practice the function $\mathbf{\Phi}^{(1)}(z)$ is responsible of tracking the error-free protocol evolution until the instant in which the first error event occurs, whereas $\mathbf{\Phi}^{(2)}(z)$ is used to track the system evolution from the error to the instant where it is resolved (either by means of timeout or retransmissions, i.e., Fast Recovery and Fast Retransmit algorithms) and a new good channel period is entered. The variable z is a vector of transform variables, $z = (z_d, z_t, z_s)$, where z_d tracks the delay, z_t the number of transmissions and z_s the number of successes. More precisely, we can define

$$\Phi_{ij}(z_d, z_t, z_s) = \sum_{N_d, N_t, N_s} \xi(N_d, N_t, N_s) z_d^{N_d} z_t^{N_t} z_s^{N_s} \tag{9}$$

where $\xi(N_d, N_t, N_s)$ is the probability that the system makes a transition to state j in exactly N_d slots, and that in slots $\{1, 2, \ldots, N_d\}$ N_t transmission attempts are performed and N_s successes are counted, given that the system was in slot i at time 0.

In particular we can note that the transition matrix of the embedded Markov chain is given by $\mathbf{P} = \mathbf{\Phi}(1,1,1)$ whereas, the matrix of the average delays can be found as

$$\begin{aligned} \boldsymbol{D} &= \left. \frac{\partial \mathbf{\Phi}(z_d, z_t, z_s)}{\partial z_d} \right|_{z_d, z_t, z_s = 1} \\ &= \boldsymbol{D}_1 \mathbf{\Phi}^{(2)}(1,1,1) + \mathbf{\Phi}^{(1)}(1,1,1)\boldsymbol{D}_2 \end{aligned} \tag{10}$$

where

$$\begin{aligned} \boldsymbol{D}_1 &= \left. \frac{\partial \mathbf{\Phi}^{(1)}(z_d, z_t, z_s)}{\partial z_d} \right|_{z_d, z_t, z_s = 1} \\ \boldsymbol{D}_2 &= \left. \frac{\partial \mathbf{\Phi}^{(2)}(z_d, z_t, z_s)}{\partial z_d} \right|_{z_d, z_t, z_s = 1}. \end{aligned} \tag{11}$$

The average number of transmissions, $\boldsymbol{T}$, and successes, $\boldsymbol{S}$, can be found similarly. According to the renewal reward theory described in [17]–[19], we can evaluate the average TCP throughput as

$$\text{throughput} = \frac{\sum_{i \in \Omega_X} \pi_i \sum_{j \in \Omega_X} P_{ij} S_{ij}}{\sum_{i \in \Omega_X} \pi_i \sum_{j \in \Omega_X} P_{ij} D_{ij}} \tag{12}$$

where π_i, $i \in \Omega_X$, are the steady-state probabilities of the embedded Markov chain (transition matrix $\boldsymbol{P}$). The average number of transmissions per slot can be similarly computed, by using T_{ij} instead of S_{ij}. This last metric is related to the energy consumption of the protocol [20].

B. Computation of $\mathbf{\Phi}^{(1)}(z)$

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 $\mathbf{\Phi}^{(1)}$ applies to all versions of TCP.

Let $X = X(k) = (C, W_{\rm th}, W)$ the starting state of cycle k. The first part of the cycle has a duration of $N_d = n$ slots with probability $\alpha_C(n)$, $n \geq 1$. The window size at time n is a deterministic function $w(n, m, W_{\rm th}, W)$ of m, $W_{\rm th}$ and W. So we can denote the window size at time n by

$$W(n) = w(n, m, W_{\rm th}, W). \tag{13}$$

Therefore, this deterministic function can be easily tabulated. It can be observed that the evolution of the window strongly de-

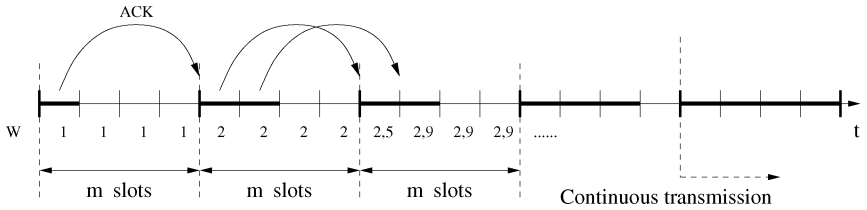

Fig. 5. Evolution of the window size.

pends on the RTT value. Fig. 5 shows an example of window evolution, where $W_{\rm th}$ in the first slot is equal to 1. Note that W remains equal to one segment for the first m slots, since the first ACK message arrives exactly an RTT later. The successive increments depend on the value of $W_{\rm th}$ and are always triggered by incoming ACKs. In more detail, the window size at time n increments if and only if a new ACK arrives, that is, if m slots earlier a successful transmission occurred. Similar considerations can be made for any initial values of W and $W_{\rm th}$, so that in general the error-free window evolution is computed applying recursively (14), shown at the bottom of the page, and taking $\min(W_n, W_{\max})$, where $f(\cdot)$ is defined in (7). One can note that in the discontinuous phase, the window size increments more slowly than in the continuous transmission mode. So, for large m, W takes more time to reach its maximum value. This strongly limits the throughput performance since the link is not fully exploited. The transmission mask, $TX(i)$, $1 \le i \le n$ is derived in a straightforward manner from the knowledge of $W(n)$

$$TX(i) = \begin{cases} 1, & \Delta(W_i, m) \ge 0 \\ 0, & \Delta(W_i, m) < 0 \end{cases} \tag{15}$$

where

$$\Delta(W_i, m) = \begin{cases} W_i - m, & i = km;\ k \in \mathbb{N} \\ W_i - (i - \lfloor \frac{i}{m} \rfloor m), & \text{elsewhere} \end{cases}. \tag{16}$$

With the function $\Delta(W_i, m)$ we are checking whenever the window in position i suffices to ensure a transmission in that slot. In more detail, by subdividing (starting from the first slot of the cycle) the time in rounds of length m, $s_{cr} = i - \lfloor i/m \rfloor m$ gives the number of slots covered by i in the current round. So, remembering that the protocol, at any time, allows a maximum of W_i unacknowledged packets, a packet in position i can be transmitted only if $W_i - s_{cr} \ge 0$. Similarly, when i is an integer multiple of m, we have a transmission only if the window size in that slot is greater or equal to m. Moreover, at time n the delay N_d is equal to n slots, but the number of packet transmissions, N_t, is no longer equal to n like in [5], but it is equal to $\eta = \eta(n, m, W_{\rm th}, W)$, that is the number of packets actually transmitted. This value is computed using the transmission mask as follows:

$$\eta(n, m, W_{\rm th}, W) = \sum_{i=1}^{n} TX(i). \tag{17}$$

The number of successes is $N_s = \eta - 1$. Therefore, once the starting state X and the final state $Y = (W_{\rm th}(n), W(n))$, have been selected we can write

$$\Phi_{XY}^{(1)}(z_d, z_t, z_s) = \sum_{n \in \mathcal{C}(X,Y)} \alpha_C(n) z_d^n z_t^{\eta} z_s^{\eta - 1} \tag{18}$$

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

C. Computation of $\mathbf{\Phi}^{(2)}(z)$

The second part of the cycle is characterized by a transition function, $\mathbf{\Phi}^{(2)}(z)$, that depends on the way the different TCP versions handle packet loss recovery and, therefore, it must be computed separately in the various cases.

Define the time in which the first error occurs as time 0, so that the first slot in the second part of the cycle corresponds to time 1. Consider, also, $\varphi_{ij}(k, n)$ as the probability that there are k successes in slots 1 through n and that the channel is in state j at time n, given that the channel was in state i at time 0. Let $Y = (W_{\rm th}, W)$ be the starting state for the second half of the cycle. Then, referring to (5), the number of packets that the transmitter is allowed to send after the error is $n_{tx} = \lceil W_{\rm fin} \rceil - 1$.

D. Computation of $\mathbf{\Phi}^{(2)}(z)$ for OldTahoe

In the case of TCP OldTahoe, there are neither Fast Retransmit nor Fast Recovery. 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, will start in state $X(k+1) = (C, \lceil W_{\rm fin}/2 \rceil, 1)$ with transition function

$$p_{BC}(T_0 - 1) z_d^{T_0 - 1} z_t^{n_{tx}} z_s^{N_s} \tag{19}$$

where N_s is a random variable. Since the exact analysis is tedious to compute, we use here a simple approach based on a bounding technique. In such cases, instead of considering the average of the reward function for each transition, $E[N_s]$, we take $0 \le N_s \le n_{tx}$, where $N_s = 0$ correspond to the case in

$$W_n = \begin{cases} W, & n = 1, \ldots, m \\ W_{n-1} + f(W_{n-1}), & W_{n-m} \ge (n - \lfloor \frac{n}{m} \rfloor m),\ n = km + u;\ k \in \mathbb{N},\ u = 1, 2, \ldots, m-1 \\ W_{n-1} + f(W_{n-1}), & W_{n-m} \ge m,\ n = km;\ k \in \mathbb{N} \\ W_{n-1}, & \text{elsewhere} \end{cases} \tag{14}$$

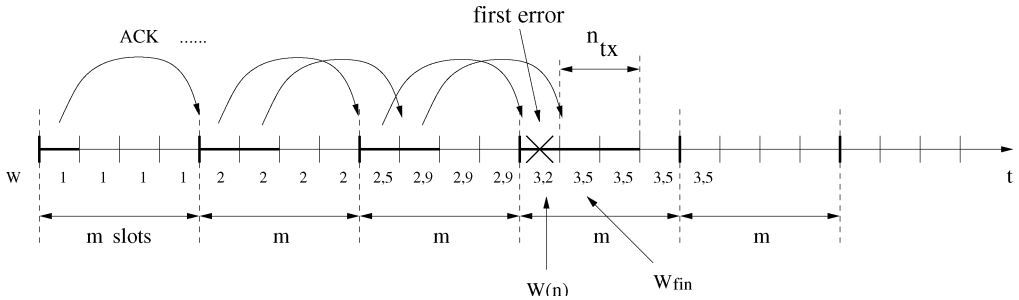

Fig. 6. Assumption of the first error in the first slot on a round trip.

which none of the transmissions after the first error is successful, while the upper bound counts all these transmissions as correct. This approach gives two analytical bounds for the throughput performance.

Moreover, here, a simple approximation is used, that is, we consider that the first error in the cycle always occurs in the first slot of an RTT, as shown in Fig. 6. This is equivalent to assuming a continuous transmission after the first error and only in this case the analysis above is correct. In correlated channels, since the error burst exactly starts from the loss at time 0, the assumption to have a continuous transmission after the first error leads to a pessimistic case, i.e., where the burst is more likely affecting all the packets transmitted after time 0. Another approximation is the one used to account for the timeout value. In particular, we assume that a single timeout timer is used for the TCP connection and that it is re-initialized at the beginning of time 0, i.e., exactly when the first error occur. This justifies the duration of the cycle T_0 [see (19)]. In real TCP implementations, the instants in which the timeout timer is restarted depend on the RTT value, m, on the transmission mask and on where errors occur. For these reasons an exact tracking of this process, though in principle possible, turns out to be very tedious and the assumption above is used instead. We proved by simulations that this assumption has a negligible effect when T_0 is large compared to m, that is also the case of interest. The same assumptions will be implicitly considered in the sequel.

E. Computation of $\mathbf{\Phi}^{(2)}(z)$ for Tahoe

In the case of Tahoe, there is no Fast Recovery. Once Fast Retransmit is triggered, regular transmission is resumed, starting from the first unacknowledged packet and by setting the TCP window to 1. On the other hand, if Fast Retransmit cannot be triggered, then a timeout event has to be waited for.

1) $n_{tx} < K$:
If $n_{tx} < K$, fast retransmit can not be triggered since the number of packets that can be further transmitted is less than K. So, K duplicate ACKs will never be received. In this case, the 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 the timeout instant will be greater than W, since new ACKs can arrive from the previous RTT. Therefore, the window size advances till $W_{\rm fin}$. Hence, after the timeout, the algorithm will set $W_{\rm th} = \lceil W_{\rm fin}/2 \rceil$, $W = 1$. Then, the transition function to state $X(k+1) = (C, \lceil W_{\rm fin}/2 \rceil, 1)$ is

$$p_{BC}(T_0 - 1) z_d^{T_0 - 1} z_t^{n_{tx}} z_s^{N_s}. \tag{20}$$

N_s is bounded as in the OldTahoe case, in particular, $N_s = 0$ gives a lower bound for the throughput performance, while the upper bound is obtained by setting $N_s = n_{tx}$.

2) $n_{tx} \geq K$:

Case 2.1 - *Fast Retransmit is not Triggered:* If fewer than K slots in n_{tx} are successful, fast retransmit will not be triggered. At this point, we rely on the approximation above that each error occurs in the first slot of the round. Only where this condition is verified, in fact, are the n_{tx} packets following the erroneous one at time 0 transmitted continuously, and the following analysis is exact. The same consideration applies to the analysis presented in the rest of the paper. The destination values of $W_{\rm th}$ and W will be as in the previous case. Let $\mathcal{A}(C', j)$ be the event that there are j successful slots in $\{1, 2, \ldots, n_{tx}\}$ ad that the channel in slot n_{tx} is C'. Then, $P[\mathcal{A}(G, j)] = \varphi_{BG}(j, n_{tx})$ and $P[\mathcal{A}(B, j)] = \varphi_{BB}(j, n_{tx})$. The transition function from state Y to state $X(k+1) = (C, \lceil W_{\rm fin}/2 \rceil, 1)$ is then given by

$$p_{BC}(T_o - n_{tx} - 1) z_d^{T_o - 1} z_t^{n_{tx}} \sum_{j=0}^{K-1} P[\mathcal{A}(B, j)] z_s^{N_s(B,j)}$$
$$+ p_{GC}(T_o - n_{tx} - 1) z_d^{T_o - 1} z_t^{n_{tx}} \sum_{j=0}^{K-1} P[\mathcal{A}(G, j)] z_s^{N_s(G,j)} \tag{21}$$

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 n_{tx}. Note that the sums are limited to $K - 1$ instead of n_{tx} since the number of successes must be less than K for the considered case of fast retransmit not triggered.

Case 2.2 - *Fast Retransmit is Triggered:* If the Kth duplicate ACK is received, fast retransmit is triggered m slots after the Kth success in $\{1, 2, \ldots, n_{tx}\}$. Define as $K \leq i \leq n_{tx}$ the slot in which the

Kth successful transmission occurs. Then the next cycle starts in slot $i+m$, the destination state is $X(k+1)=(C,\lceil W_{\rm fin}/2\rceil,1)$, and the transition function is given by

$$\varphi_{BG}(K,i)\sum_{C'\in\{G,B\}}\sum_{j=0}^{n(i)}\Big\{P_{C'}[j|n(i)]\times p_{C'C}(m-n(i)-1)\,z_d^{i+m-1}z_t^{n_{tx\rm eff}(i)}z_s^{N_s}\Big\}\quad(22)$$

where $n(i)=n_{tx\rm eff}(i)-i$ and $n_{tx\rm eff}(i)$ is the number of packets transmitted between the error at time 0 and the beginning of the recovery phase (when the Kth duplicate ACK is received)

$$n_{tx\rm eff}(i)=\begin{cases}n_{tx}, & n_{tx}<i+m-1\\ i+m-1, & n_{tx}\ge i+m-1\end{cases}\quad(23)$$

C' is the channel state in the slot where the $n_{tx\rm eff}(i)$th packet is transmitted, whereas C is the channel state in the slot just before the one in which the next cycle starts. The function $P_{C'}[j|n(i)]$ represents the probability to have j correct packets in $\{i+1,\dots,n_{tx\rm eff}(i)\}$

$$P_{C'}[j=0|n(i)=0]=\begin{cases}1, & C'=G\\ 0, & C'=B\end{cases}\quad(24)$$

$$P_{C'}[j|n(i)>0]=\varphi_{GC'}(j,n(i)).\quad(25)$$

The quantity N_s can be bounded by $0\le N_s\le K+j$, where 0 and $K+j$ represent the lower-bound and the upper-bound case, respectively. The lower bound accounts here for the very worst case where all the $K+j$ packets correctly transmitted in slot 1 through $n_{tx\rm eff}(i)$ are retransmitted during following cycles.

F. Computation of $\tilde{\mathbf{\Phi}}^{(2)}(z)$ for Reno

Since TCP Reno differs from Tahoe only after fast retransmit is triggered, all the events considered for TCP Tahoe and corresponding to no fast retransmit still apply in this case. Hence, in the following we only need to consider the case in which fast retransmit is triggered (Case 2.2), i.e., the case in which the Kth successful transmission occurs at time i in $\{1,2,\dots,n_{tx}\}$. Let $\mathcal{B}(K)$ be the event that the packet failure at time 0 is followed by K consecutive successes, and let $\mathcal{B}(i,l_1)$, $K<i\le n_{tx}$, $0<l_1\le K$ 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 shown in (26) and (27) at the bottom of the page.

Case 2.2.a: Consider first the occurrence of the event $\mathcal{B}(K)$. Since at the end of slot $K+m-1$ the Kth duplicate ACK is received, the retransmission is performed in slot $K+m$.

- If this retransmission is successful, the loss recovery phase is successfully completed and a new cycle starts at time $K+2m$. In this case, the destination state is $X(k+1)=(C,\lceil W_{\rm fin}/2\rceil,\lceil W_{\rm fin}/2\rceil)$[9] and the transition function is given by

$$P[\mathcal{B}(K)]p_{GG}(m)p_{GC}(m-1)\times z_d^{K+2m-1}z_t^{n_{tx\rm eff}(K)+1}z_s^{N_s+1}\quad(28)$$

where we account for the probability to have the error at time 0 followed by K correct packets ($P[\mathcal{B}(K)]$) and a correct retransmission (term $p_{GG}(m)$, m slots after the transmission of the Kth packet); the term $p_{GC}(m-1)$ accounts for the channel state in the slot preceding the start of the new cycle. N_s can be bounded as $K\le N_s\le n_{tx\rm eff}(K)$, giving the throughput lower and upper bound, respectively.

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

$$P[\mathcal{B}(K)]p_{GB}(m)p_{BC}(T_o-K-m-1)\times z_d^{T_o-1}z_t^{n_{tx\rm eff}(K)+1}z_s^{N_s}.\quad(29)$$

where N_s is bounded as in the previous case and N_t accounts for the retransmission of the lost packet in addition to the $n_{tx\rm eff}(K)$ transmissions.

Case 2.2.b: Consider the occurrence of the event $\mathcal{B}(i,l_1)$. In the case of multiple losses in a congestion window, TCP Reno is able to successfully complete the Fast Recovery phase only when the congestion window at the first error is large enough. So, in this article we consider that multiple losses event always leads to deadlock and consequent timeout.[10]

- If the retransmission at time $i+m$ is a failure, we can observe that the protocol

[9]According with Reno's rules, after the reception of the Kth duplicate ACK, $W_{\rm th}$ is set to half the window size $W_{\rm fin}$ and never changed until the successful completion of the lost recovery phase, where the window size is set to $W_{\rm th}=\lceil W_{\rm fin}/2\rceil$.

[10]This assumption is better verified as the round trip increases.

$$P[\mathcal{B}(K)]=p_{BG}p_{GG}^{K-1}\quad(26)$$

$$P[\mathcal{B}(i,l_1)]=\begin{cases}p_{BB}\varphi_{BG}(K,i-1), & l_1=1;\ i=K+1,\dots,n_{tx}\\ p_{BG}p_{GG}^{l_1-2}p_{GB}\varphi_{BG}(K-l_1+1,i-l_1), & l_1=2,\dots,K;\ i=K+1,\dots,n_{tx}\end{cases}\quad(27)$$

behavior is similar to the previous case, i.e., the next cycle will start in state $X(k+1)=(C,\lceil W'/2\rceil,1)$, with transition function

$$P[\mathcal{B}(i,l_1)]p_{GB}(m)p_{BC}(T_o-i-m-1)\times z_d^{T_o-1}z_t^{n_{tx\text{eff}}(i)+1}z_s^{N_s} \tag{30}$$

where the term $P[\mathcal{B}(i,l_1)]p_{GB}(m)$ represents the probability that the Kth correct packet (after the loss at time 0) is transmitted in slot i and that the retransmission of the first lost packet is unsuccessful. In this case, a time out event must be waited for; this will happen exactly T_o slots after the beginnig of slot 0. Here, N_s can be bounded as $l_1-1\le N_s\le n_{tx\text{eff}}(i)$.

- On the other hand, if the retransmission at time $i+m$ is successful the system will timeout at the end of slot $T_o+i+2m-1$. In fact, according to the protocol rules, we consider only one timeout timer associated to the connection and this timer is reset (see [15]) at the reception of the ACK relative to the loss at time 0 (at the end of slot $i+2m-1$). The window size will be updated to $W''=\min\{\lceil W_{\text{fin}}/2\rceil+K+1,W_{\max}\}$ (the ACK for the successful retransmission causes the window to be further increased by one with respect to the previous case). The next cycle will restart in state $X(k+1)=(C,\lceil W''/2\rceil,1)$ with transition function

$$P[\mathcal{B}(i,l_1)]p_{GG}(m)p_{GC}(T_o+m-1)\times z_d^{T_o+i+2m-1}z_t^{n_{tx\text{eff}}(i)+1}z_s^{N_s} \tag{31}$$

where N_s can be bounded as $l_1\le N_s\le n_{tx\text{eff}}(i)+1$. In fact, in this case, the number of successes is at least one more than in the previous case.

G. Computation of $\mathbf{\Phi}^{(2)}(z)$ for New Reno

Finally, let us consider TCP New Reno. The only case in which it is different from 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 2.2.b). All other cases are the same as in Reno. Refer to the definition of $\mathcal{B}(i,l_1)$, $K<i\le n_{tx}$, $0<l_1\le K$ as the event in which the Kth success occurs at time i and the first loss after the loss in 0 occurs at time l_1. If the Kth successful transmission is performed at time $i>K$, then besides the first lost packet (at time 0) there are $i-K$ losses in slot l_1 through $i-1$.

1) Successful Loss Recovery: Let n_{error} be the total number of packets in error in slots $\{0,1,\ldots,n_{tx\text{eff}}(i)\}$. Conditioned on i, this variable is given by the sum of a deterministic and a random term. The former is the number of errors up to slot i (that is $i-K+1$), while the latter, that we call n_{loss}, $0\le n_{\text{loss}}\le(n_{tx\text{eff}}(i)-i)$, represents the number of errors in $\mathcal{Q}=\{i+1,\ldots,n_{tx\text{eff}}(i)\}$. Let also $P_{C'}[n_{\text{loss}}]$ be the probability to have exactly n_{loss} losses in $\mathcal{Q}$ and to have channel state C' in slot $n_{tx\text{eff}}(i)$. This probability is computed as

$$P_{C'}[n_{\text{loss}}=0|n(i)=0]=\begin{cases}1, & C'=G\\ 0, & C'=B\end{cases} \tag{32}$$

$$P_{C'}[n_{\text{loss}}|n(i)>0]=\varphi_{GC'}(n(i)-n_{\text{loss}},n(i)) \tag{33}$$

where $n(i)=n_{tx\text{eff}}(i)-i$. If we have n_{error} consecutive successful retransmissions, loss recovery is successfully completed, since at time $i+mn_{\text{error}}$ the last lost packet is successfully retransmitted and the corresponding ACK (received at the end of slot $i+mn_{\text{error}}+m-1$) will acknowledge all outstanding packets. A new cycle will then start at time $i+(n_{\text{error}}+1)m$ in state $X(k+1)=(C,\lceil W_{\text{fin}}/2\rceil,\lceil W_{\text{fin}}/2\rceil)$, since W_{th} remains equal to $\lceil W_{\text{fin}}/2\rceil$ and W is set to W_{th} upon completion of the loss recovery phase. The transition function corresponding to this event is

$$\begin{aligned}P[\mathcal{B}(i,l_1)]\cdot\sum_{C'\in\{G,B\}}\sum_{n_{\text{loss}}=0}^{n(i)}\Big\{&P_{C'}[n_{\text{loss}}|n(i)]\\ &\times p_{C'G}(m-n(i))p_{GG}(m)^{n_{\text{error}}-1}p_{GC}(m-1)\\ &\times d^{i+(n_{\text{error}}+1)m-1}z_t^{n_{tx\text{eff}}(i)+n_{\text{error}}}\Big\}z_s^{n_{tx\text{eff}}(i)+1}\end{aligned} \tag{34}$$

where $n(i)=n_{tx\text{eff}}(i)-i$ and $n_{\text{error}}=i-K+1+n_{\text{loss}}$. Note that, since the recovery phase ends successfully, all packets in slots $\{0,1,\ldots,n_{tx\text{eff}}(i)\}$ are eventually received correctly, so N_s is exactly given by $n_{tx\text{eff}}(i)+1$. Note that C is the channel state in the slot just before the one in which the new cycle begins, whereas C' represents the channel state in slot $n_{tx\text{eff}}(i)$. In (34), we consider the case of multiple losses in a window. The total number of losses has been separated in two contributions. The first is relative to the number of lost packets between the loss at time 0 and the transmission of the Kth correct packet ($i-K+1$ losses). The second contribution, instead, accounts for the number of losses between the transmission of the Kth correct packet and the slot in which the fast retransmit starts (n_{loss} losses). As observed above, the total number of losses is given by $n_{\text{error}}=i-K+1+n_{\text{loss}}$. In (34) we sum over all possible values of n_{error} and over all possible values of the channel state in the slot where the last of the $n_{tx\text{eff}}(i)$ packets is transmitted. From here, we use the term $p_{C'G}(m-n(i))$ to evolve the channel state until the slot in which fast retransmit starts.[11] In z_d we account for the total delay, i.e., we sum to i the time needed to recover from the n_{error} losses after the Kth successful packet (transmitted is slot i), in z_t we count all the transmitted packets, whereas in z_s we count only successes ($n_{tx\text{eff}}(i)$ packets sent before the beginning of the recovery phase plus the retransmission of the loss at time 0).

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 fewer than n_{error} successes. Suppose that, after the successful retransmission of the first loss at time 0, we have exactly j consecutive good retransmissions ($0\le j\le n_{\text{error}}-2$) followed by a failure. This event has probability $p_{GG}^j(m)p_{GB}(m)$. After the jth success

[11]Since we are in the successful loss recovery case, we assume that the first and all following retransmissions are successful.

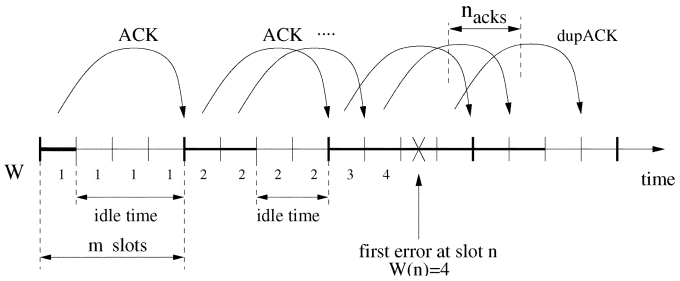

Fig. 7. n_{acks}.

the value of the window is given by $W''' = \min\{\lceil W_{\mathrm{fin}}/2\rceil + K + 1 + j, W_{\max}\}$. The initial state of the next cycle corresponds to the timeout relative to the $(j+2)$nd loss in the window, since the first $j+1$ have been successfully retransmitted, and is $X(k+1) = (C, \lceil W'''/2\rceil, 1)$. The transition function in this case is given by

$$P[\mathcal{B}(i,l_1)] \sum_{C'\in\{G,B\}} \sum_{n_{\mathrm{loss}}=0}^{n(i)} \Bigg\{ P_{C'}[n_{\mathrm{loss}}|n(i)] p_{C'G}(m-n(i)) \times \sum_{j=0}^{n_{\mathrm{error}}-2} \Big[p_{GG}(m)^j p_{GB}(m) p_{BC}(T_o-1) \times z_d^{i+(j+2)m+T_o-1} z_t^{n_{tx\mathrm{eff}}(i)+j+2} z_s^{j+1} \Big] \Bigg\} \tag{35}$$

where $n(i) = n_{tx\mathrm{eff}}(i) - i$ and $n_{\mathrm{error}} = i - K + 1 + n_{\mathrm{loss}}$. In the equation above, similarly to (34), we sum over all possible values of n_{error} and C'. In this case, however, not all packets in error are successfully recovered. We account for this fact by means of the sum over j. Here, a timeout event must be waited for, and it will happen exactly $T_o - 1$ slots after the first retransmission failure (term $p_{BC}(T_o - 1)$).

Finally, the transition function matrix $\mathbf{\Phi}^{(2)}$ is computed by assigning to each entry $\mathbf{\Phi}^{(2)}_{XY}$ the sum of all the transition functions leading from state X to state Y, as explained in [5].

IV. Results

In this Section the throughput performance of the various versions of TCP are analyzed.

First of all we note that, in our study, we have considered some approximations in order to keep the model analytically feasible. In the following we point out these approximations in order to better clarify the analysis and to permit a better understanding of the results presented in this Section. First of all, (4) gives an overestimate of the number of ACKs received after the first error. This is true especially for high error probabilities and large values of m. Under these conditions, in fact, the number of packets in flight could be less than the window value at the time in which the error occurs (see Fig. 7). This happens when the transmission in the round preceding the one where the error occurs is discontinuous. In any case, the use of (4) always leads to an analytical upper bound for the window size reached after the first error (W_{fin}). For this reason, what we model is an approximated version of the real protocol that corresponds to an analytical upper-bound for what concerns throughput performance of the real protocol. Hence, the throughput lower and upper bounds considered in the computation of the matrix $\mathbf{\Phi_2}(z)$ must be viewed as the upper and the lower bounds for the approximate model and not for the real protocol. In any event, as will be clear from the results presented, the approximation considered is very good.

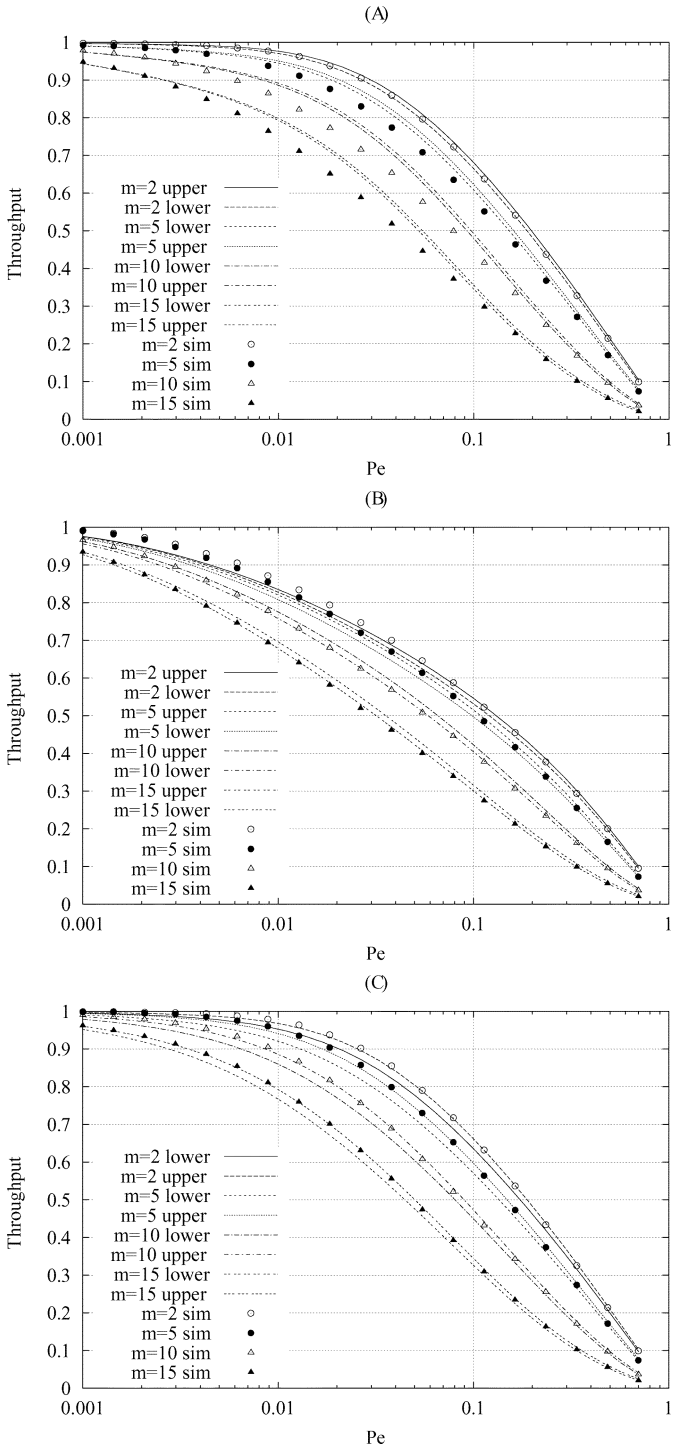

Fig. 8. Throughput comparison between analysis and simulation with $f_dT = 0.01, K = 3, W_{\max} = 16, T_o = 100$. (a) Tahoe, (b) Reno, and (c) New Reno.

In Fig. 8(a), we report the throughput of TCP Tahoe as a function of the channel error probability P_e by fixing $f_dT = 0.01$ (correlated channel case). In Fig. 8, three kinds of curves are reported: the analytical throughput upper and lower bound and

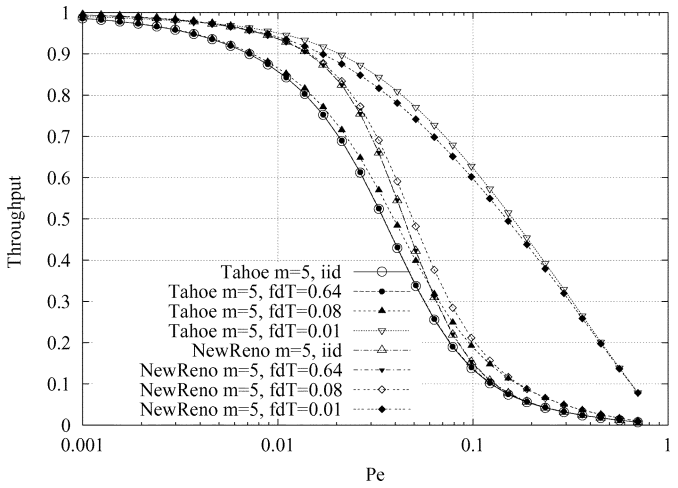

Fig. 9. Throughput (upper bound) comparison by varying f_dT, $K = 3$, $W_{\max} = 16$, $m = 5$, $T_o = 100$.

the values obtained by simulation. As the RTT (m) increases analysis and simulation differ. In particular, the throughput obtained from the analysis is slightly higher than the one obtained by simulation. This is justified by the way in which the quantity n_{ack} has been computed.

In Fig. 8(b), we compare simulation results and analysis for TCP Reno. Here, the approximation introduced in (4) is less influential than in the previous case. This is justified by the fact that, after a successful recovery phase both W and W_{th} are set to $\lceil W_{\text{fin}}/2 \rceil$. By doing so, the performance is less influenced by the value of W_{th} with respect to the Tahoe case (where $W = 1$ and $W_{\text{th}} = \lceil W_{\text{fin}}/2 \rceil$).[12] Moreover, in this figure we note some differences between simulation and analysis when both m and P_e are low. These differences are due to our assumption that TCP Reno always times out in the case of multiple losses in a window. This assumption, in fact, is good only when either m or P_e has a large value, in any other case $W(n)$ could be sufficiently large to ensure the recovery of multiple losses. However, the approximation above appears to be very good as m increases.

Fig. 8(c) shows the same comparison for TCP New Reno. Here, no simplified assumptions are made regarding the recovery process. Simulation results in this case match almost perfectly with the analysis.

In Fig. 9, we compare the throughput[13] of New Reno and Tahoe by varying the channel memory (f_dT). When errors are correlated ($f_dT = 0.01$), New Reno presents a slightly lower throughput than Tahoe. In this case, a burst (b) of erroneous packets follows the error at time n. At this point, TCP Tahoe reacts to the reception of the Kth duplicate ACK by re-starting the connection with the slow-start algorithm. From this point on, the $b + 1$ lost packets are retransmitted as if a timeout event had occurred. In the New Reno recovery phase, instead, it is possible to transmit only one packet per RTT (m slots). So, to recover from the $b + 1$ losses and restore the normal transmission mode a total of $m(b + 1)$ slots are needed. For large values of b (e.g., for $f_dT = 0.01$) this strategy (New Reno) leads to a larger recovery

[12]Here, W_{th} is determinant for the time needed by W to reach m, i.e., to fill the bandwidth-delay product.

[13]Since the curves relative to the throughput lower and upper bounds are very close to each other, in the following graphs we report only the upper bound curve.

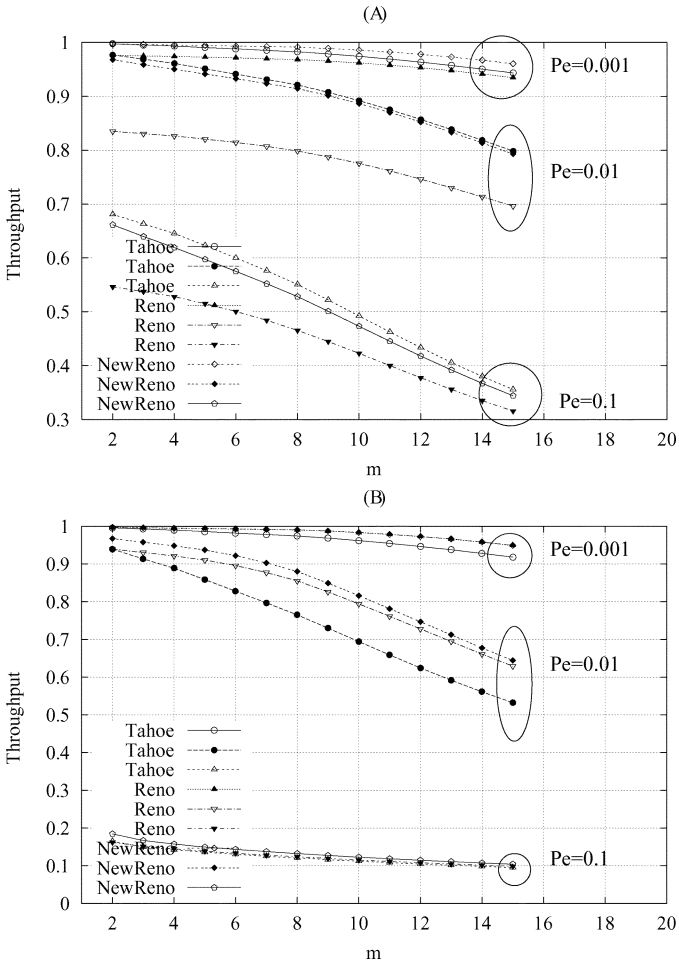

Fig. 10. Throughput (upper bound) versus m with $K = 3$, $W_{\max} = 16$, $T_o = 100$. (a) $f_dT = 0.01$, (b) $f_dT = 0.64$.

time with respect to Tahoe. This is the reason why, in this case, New Reno is affected by a lower throughput. Note that this problem is not present when instantaneous feedback ($m = 1$) is considered, while it becomes more relevant as m increases.

The situation is reversed at low channel correlation ($f_dT \in \{0.08, 0.64\}$). Here, the best recovery strategy is the one implemented by New Reno. This is justified by the fact that New Reno has a more persistent recovery strategy, able to recover multiple losses without leading to a timeout event. This event, in uncorrelated channel is more probable than in correlated environments, due to the fact that the error event (the error burst) probability in this case is larger even if the average packet error probability is the same.

Note also that, from the point of view of the protocol performance, a channel characterized by $f_dT = 0.64$ is equivalent to the independent error case (i.i.d.).

The throughput as a function of m is reported in Fig. 10, for the cases of $f_dT = 0.01$ [Fig. 10(a)] and $f_dT = 0.64$ [Fig. 10(b)]. First of all, we note that uncorrelated errors have a higher impact on throughput than correlated channels. Moreover, in the $f_dT = 0.01$ case, TCP Reno is the one with the lower throughput, and New Reno and Tahoe present almost the same performance. Where $f_dT = 0.64$, instead, Tahoe becomes the worst, New Reno the best, whereas Reno performs in the middle.

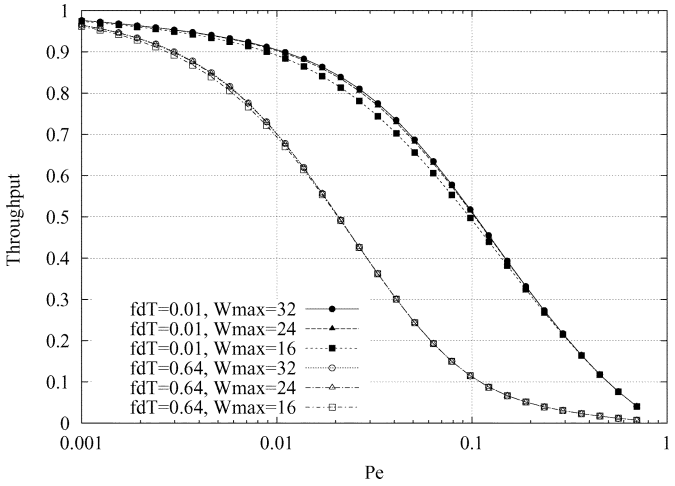

Fig. 11. Tahoe, throughput (upper bound) versus P_e, by varying $W_{\max}$ and f_dT with $K = 3$, $m = 10$ and $T_o = 100$.

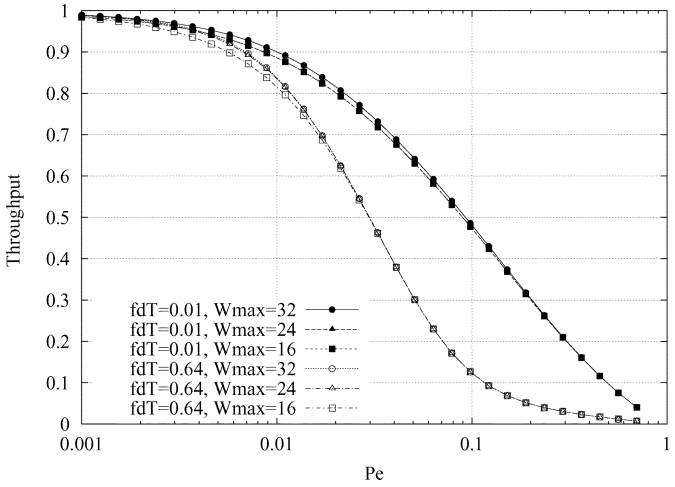

Fig. 12. New Reno, throughput (upper bound) versus P_e, by varying $W_{\max}$ and f_dT with $K = 3$, $m = 10$ and $T_o = 100$.

Figs. 11 and 12 show the TCP throughput as a function of the channel error probability P_e by varying the channel correlation (f_dT) and $W_{\max}$. We note that the curves depend weakly on the $W_{\max}$ value. In particular, when the channel is correlated ($f_dT = 0.01$) the greatest differences are in the New Reno case, i.e., where the Fast Recovery and Fast Retransmit algorithms are used. In such cases, in fact, the success of these recovery algorithms depends on the maximum window size reached during the error-free phase (first cycle in our analysis): the larger the value of the window, the higher the probability that, after the first error, K packets are sent with success. This results in a higher probability of triggering the Fast Recovery algorithm, thereby avoiding timeout and increasing the performance.

We can conclude that the maximum window size must be greater than the bandwidth-delay product ($\geq m$) in order to reach the continuous transmission phase by avoiding the waste of available resources. However, once it has been set following this rule, its size is not determinant to improve performance regardless of the degree of correlation characterizing the packet error process.

V. Conclusion

In this paper, we have developed an analytical framework, based on Markov renewal theory, in order to describe the throughput performance of a TCP protocol working over a wireless link. The main goal of our analysis was the study of TCP performance in a link characterized by a finite round-trip delay. Even though some approximations are introduced for analytical tractability, comparison with simulation results shows that the considered approach is very accurate and is able to predict very precisely the throughput performance of various versions of TCP. The main findings of this study are that: 1) an increasing RTT may significantly affect the throughput performance of TCP, especially when an independent channel is considered; 2) New Reno performs better than Reno and Tahoe when the channel is uncorrelated, whereas the Tahoe recovery strategy is the most efficient when the channel correlation is high; and 3) the maximum window size does not play a determinant role in increasing throughput performance in both correlated and independent channels.

References

[1] 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*, vol. 5, pp. 336–350, June 1997.

[2] O. Ait-Hellal and E. Altman, "Analysis of TCP vegas and TCP reno," in *Proc. IEEE Int. Conf. Communications, ICC*, vol. 1, Monréal, QC, Canada, June 1997, pp. 495–499.

[3] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP reno performance: A simple model and its empirical validation," *IEEE/ACM Trans. Networking*, vol. 8, pp. 133–145, Apr. 2000.

[4] A. Kumar, "Comparative performance analysis of versions of TCP in a local network with a lossy link," *IEEE/ACM Trans. Networking*, vol. 6, pp. 485–498, Aug. 1998.

[5] M. Zorzi, A. Chockalingam, and R. R. Rao, "Throughput analysis of TCP on channels with memory," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 1289–1300, July 2000.

[6] A. A. Abouzeid, S. Roy, and M. Azizoglu, "Stochastic modeling of TCP over lossy links," in *Proc. 19th Annu. Joint Conf. IEEE Computer and Communications, INFOCOM*, vol. 3, Tel-Aviv, Israel, Mar. 2000, pp. 1724–1733.

[7] F. Anjum and L. Tassiulas, "On the behavior of different TCP algorithms over a wireless channel with correlated packet losses," in *Proc. ACM Int. Conf. Measurement and Modeling of Computer Systems, SIGMETRICS*, Atlanta, GA, 1999, pp. 155–165.

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

[9] J. Postel, "Transmission Control Protocol, Darpa Internet Program Protocol Specification,", RFC 793, 1981.

[10] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM Symp. Communications Architectures and Protocols, SIGCOMM*, Stanford, CA, Aug. 1988, pp. 314–329.

[11] W. R. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithm,", RFC 2001, 1997.

[12] M. Allman and W. R. Stevens, "TCP Congestion Control,", RFC 2581, 1999.

[13] K. Fall and S. Floyd. (1995) Simulation-Based Comparisons of Tahoe, Reno and SACK TCP. [Online]. Available: ftp.ee.lbl.gov/papers/sacks.ps.Z

[14] S. Floyd and T. Henderson, "The New Reno Modification to TCP's Fast Recovery Algorithm,", RFC 2582, 1999.

[15] V. Paxson and M. Allman, "Computing TCP's Retransmission Timer,", RFC 2988, 2000.

[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*, Tokyo, Japan, Nov. 1995, pp. 211–215.

[17] R. A. Howard, *Dynamic Probabilistic Systems*. New York: Wiley, 1971.

[18] S. M. Ross, *Stochastic Processes*, 2nd ed. New York: Wiley, 1996.

[19] M. Zorzi and R. R. Rao, "Energy constrained error control for wireless channels," *IEEE Personal Commun. Mag.*, vol. 4, pp. 27–33, Dec. 1997.
[20] ——, "Energy efficiency of TCP in a local wireless environment," in *Mobile Networks and Applications*. Norwell, MA: Kluwer Academic, 2001, vol. 6, pp. 265–278.

Michele Rossi (S'02–M'04) was born in Ferrara, Italy, on October 30, 1974. He received the Laurea degree in electrical engineering (*summa cum laude*) from the University of Ferrara in 2000, where he is currently pursuing the Ph.D. degree.

During the Academic Year 2000/2001, he was a research fellow with the Department of Engineering, University of Ferrara. In 2003, he spent six months at the Center for Wireless Communications (CWC) at the University of California San Diego (UCSD), where he did research on wireless sensor networks. His research interests are on: TCP/IP protocols on wireless networks, TCP/IP header compression, performance analysis of Selective Repeat Link Layer retransmission techniques, efficient multicast data deliver in 3G cellular networks, and MAC/routing in wireless sensor networks.

Raffaella Vicenzi was born in Mantova, Italy, on April 18, 1975. She received the Dr. Eng. degree in electronic engineering (*summa cum laude*) from the University of Ferrara, Ferrara, Italy, in December 2000.

During the Academic Year 2000/2001 she was a research fellow at the Department of Engineering, University of Ferrara.

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, 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, Milan, Italy. After spending three years with the Center for Wireless Communications at UCSD, in 1998 he joined the School of Engineering of the University of Ferrara, Italy, where he is currently a Professor. His present research interests include performance evaluation in mobile communications systems, random access in mobile radio networks, ad hoc and sensor networks, and energy constrained communications protocols.

Dr. Zorzi is the Editor-In-Chief of the *IEEE Wireless Communications Magazine*, and currently serves on the Editorial Boards of the IEEE TRANSACTIONS ON COMMUNICATIONS, the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE TRANSACTIONS ON MOBILE COMPUTING, the *Wiley Journal of Wireless Communications and Mobile Computing* and the *ACM/URSI/Kluwer Journal of Wireless Networks*. He was 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).