Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
Computer Networks 44 (2004) 153–176
COMPUTER NETWORKS
www.elsevier.com/locate/comnet

Modeling short-lived TCP connections with open multiclass queuing networks ☆

M. Garetto, R. Lo Cigno [1], M. Meo, M. Ajmone Marsan *

Dipartimento di Elettronica, Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 Torino, Italy

Received 22 September 2002; received in revised form 26 March 2003; accepted 24 July 2003

Responsible Editor: S. Palazzo

Abstract

In this paper we develop an open multiclass queuing network model to describe the behavior of short-lived TCP connections sharing a common IP network. The queuing network model is paired with a simple model of the IP network, and the two models are solved through an iterative procedure. The combined models need as inputs only the primitive network parameters, and they produce estimates of the packet loss probability, the round trip time, the TCP connection throughput, and of the average TCP connection completion time (that is, of the average time necessary to transfer a file with given size over a TCP connection). We derive models for both TCP-Tahoe and TCP-NewReno. The Tahoe model is presented in detail, while the NewReno model is presented describing differences with respect to Tahoe. Results are shown for both models. The analytical performance predictions are validated against detailed simulation experiments in realistic networking scenarios, proving that the proposed modeling approach is accurate.
© 2003 Elsevier B.V. All rights reserved.

Keywords: Queuing models; TCP performance; Short-lived connections; Transfer latency; Completion time

1. Introduction

Models of the TCP behavior are receiving great attention from both the academic and the industrial communities. The reason for such enormous interest is essentially one: the behavior of TCP drives the performance of the Internet. Over 90% of the Internet connections use TCP, and over 95% of the bytes that travel over the Internet use TCP [1]; hence, the availability of an accurate model of TCP is a necessity for the design and planning of Internet segments and corporate intranets.

Models appeared so far in the literature can be grouped in two classes:

☆ This work was supported by the Italian Ministry for University and Scientific Research through the PLANET-IP Project. A short, preliminary version of this work was presented at the *Seventh International Workshop on Protocols For High-Speed Networks (PfHSN 2002)*, Berlin, Germany, April 22–24, 2002.

* Corresponding author. Tel.: +39-011-5644032; fax: +39-011-5644099.

E-mail addresses: garetto@polito.it (M. Garetto), locigno@polito.it (R. Lo Cigno), michela@polito.it (M. Meo), ajmone@polito.it (M. Ajmone Marsan).

[1] Renato Lo Cigno is currently with Dipartimento di Informatica e Telecomunicazioni, Università di Trento, Via Sommarive 14, 38050 Povo, Trento, Italy.

1389-1286/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S1389-1286(03)00349-9

1. Models that assume that the round trip time (RTT) and the loss characteristics of the IP network are known, and try to derive from them the throughput (and the delay) of TCP connections; works as [2–5] belong to this class;
2. Models that assume that only the primitive network parameters (topology, number of users, data rates, propagation delays, buffer sizes, etc.) are known, and try to derive from them the throughput and the delay of TCP connections, as well as the RTT and the loss characteristics of the IP network; works in Refs. [6–12] and many others belong to this second class.

Often, models in the second class incorporate a model similar to those of the first class, together with a simplified model of the network that carries the TCP segments, and the two models are jointly solved through a fixed point iterative procedure. This situation is illustrated by Fig. 1, where the 'TCP sub-models,' describe the behavior of TCP connections, and the 'network sub-model,' focuses on the underlying IP network. Several TCP sub-models are needed because several different groups of TCP connections can be identified in a network according to the used version of TCP (Tahoe, Reno, NewReno, SACK, ...), the loss probability, the round trip delay, and other parameters. The TCP sub-models compute the load offered to the system, given the average RTT and loss probability. The network sub-model estimates the RTT and the loss probability, given the load offered to the system.

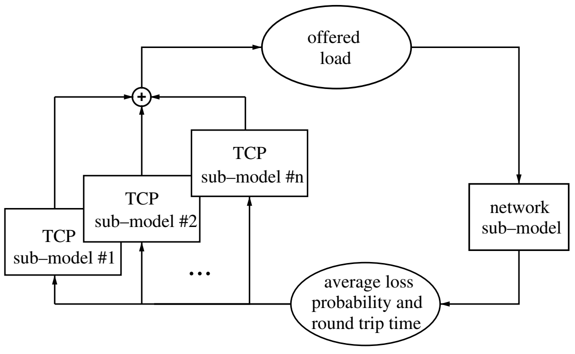

Fig. 1. High-level description of the model solution.

TCP sub-models can either assume a greedy behavior of TCP connections, i.e., consider situations representing the transfer of extremely long files with FTP, or more realistically consider the case of finite (and normally short-lived) TCP connections, as typical of Web browsing applications, standard FTP sessions, e-mail transfers, etc. The second alternative is more difficult to study, due to the overlap of the connection lifetime dynamics with the TCP protocol dynamics. While models of greedy TCP connections normally can only permit the estimation of the connection throughput, models of finite TCP connections often allow also the estimation of the (average) connection completion time.

In this paper we use open multiclass queuing networks (OMQNs) to develop a complete model that describes the behavior of an arbitrarily large number of short-lived TCP connections sharing a common infrastructure for the transfer of TCP segments. The TCP connections can use two different versions (Tahoe and NewReno), with any arbitrary mix of connections. The extension to other TCP versions is straightforward. The overall model needs as inputs only primitive network parameters, and produces estimates of the packet loss probability, the average RTT, the TCP connection throughput, and of the average TCP connection completion time. The TCP model presented here allows an unprecedented modeling accuracy for interacting short-lived TCP connections, thus providing extremely useful insight into the detailed TCP dynamics, yet maintaining a computationally simple solution.

We use simulation experiments to validate our modeling approach, showing that the performance estimates that our models can generate are extremely accurate; moreover, in case of random early detection (RED) we include a comparison with the model in [4].

The paper is organized as follows. After discussing related work in Section 1.1, we present the proposed approach in Section 2. We then describe in Section 3 the details of the TCP model by focusing on the Tahoe version of the protocol. The underlying IP network model is described in Section 4. In Section 5, we show how completion times can be computed. The model modifications which are necessary to deal with TCP-NewReno are presented in Section 6. The complexity of the model is discussed in Section 7. In Section 8, the model is validated by comparison against simulation results. Finally, Section 9 concludes the paper.

1.1. Related work

As already noted, the literature on TCP modeling is huge, thus we only recall here some works and papers most related to this work and the research project it is related to.

The analysis of the behavior of TCP presented in [2,3] are based on measurements. In [2] the modeling technique is empirical, while in [3] the model is based on the analysis of TCP transmission cycles. The packet loss ratio and connection RTT are needed as inputs in order to allow the model to compute the TCP throughput and average window size. The first paper assumes uncorrelated losses, while the second one also takes into account loss correlation. An extension to this latter paper [4] allows computing the latency of short file transfers. The work in [5] presents both an empirical model and a more detailed one that is based on the analysis of all the possible evolutions of the TCP window, depending on the loss pattern. These four works fall in the first category defined above.

The remaining works discussed here, belong instead in the second category we defined above, though the modeling technique adopted by the different authors (and the accuracy of results) may differ significantly.

The authors of [6,7] use differential stochastic equations to model the combined macroscopic behavior of TCP and the network itself, which is assumed to use active queue management (AQM) techniques. The resulting model can cope with multiple interacting connections, defining a closed-loop system that can be studied with powerful control-theoretic approaches.

Markovian modeling is used in [8], that presents Markov reward models of several TCP versions on lossy links.

The modeling technique adopted in this paper was introduced in [9] and citations therein. It is based on the description of the protocol through a queuing network that allows the estimation of the load offered to the IP network. The model in [9] considered only greedy connections; however, the TCP traffic on the Internet is the result of finite-size file transfers, and the main performance figure for users browsing the Web is the file transfer time.

Another work tightly related to our modeling technique was presented in [10]. The authors study the convergence of fixed point approximations for TCP models, and demonstrate the existence of a unique convergence point under mild constraints.

In [11] we have discussed the queuing network modeling technique in general and its application to TCP modeling, without details on any specific model. The work presented in [12], finally, is devoted to the discussion of results obtained with the TCP models described here; however in [12] the models themselves are not presented in detail. In light of the above discussion, the novel contributions of this paper are the following:

1. The OMQNs describing TCP-Tahoe and TCP-NewReno are presented in detail, discussing the insight they give on short-lived TCP connections and the possible use of the OMQN technique for performance analysis of end-to-end protocols;
2. The OMQN solution complexity is discussed showing that it is independent from the channel capacity and the number of concurrent TCP connections, depending only on the maximum window size and the maximum flow dimension;
3. Results are shown for different network scenarios, including multiple bottleneck networks and AQM routers;
4. The accuracy of results is discussed and a comparison is presented between the results obtained with our model and with the model proposed in [4].

2. Queuing network models of TCP and customer classes

Our OMQN model consists of a queuing network in which all queues are $M/G/\infty$. Customers represent TCP connections and queues stand for the possible states of the protocol. All queues have an infinite number of servers, since there are no limitations to the number of TCP connections in any given state within the system. Classes associated with customers identify the number of remaining packets to be sent before the completion of the flow. A customer belonging to class c visiting queue q corresponds to a TCP connection in the protocol state represented by q which still has c packets to transmit before the flow completion. A connection opens when a new customer enters the OMQN, the class N_P associated with the customer at the entrance into the OMQN is equal to the number of packets which compose the file to be transferred. While visiting the OMQN queues which correspond to successful packet delivery, the customer's class decreases. As soon as the customer's class becomes equal to 0, the customer leaves the OMQN, i.e., the connection is closed.

TCP connections are always opened in the same state. Only the queue corresponding to this state receives external arrivals, let this queue be q^*. Let $\lambda_{q,c}$ and $\lambda^{\mathrm{e}}_{q,c}$ respectively denote the total and the external arrival rates at queue q in class c, and let $N_P^{\max}$ denote the maximum allowed file size in packets. The external arrival rate of the OMQN is defined by the vector $[\lambda^{\mathrm{e}}_{q^*,c}] = \lambda^{\mathrm{e}}[\gamma_c]$, $1 \leqslant c \leqslant N_P^{\max}$, where λ^{e} is a traffic scaling factor, and $[\gamma_c]$ is a probability distribution vector describing the TCP connection file size.

The service times of the $M/G/\infty$ queues, that represent the time a connection spends in a given protocol state, are independent from customer classes, since the dynamics of the TCP protocol are independent from the transmission backlog.

Every time a customer visits a queue which represents the successful transmission of a segment, the class c of the customer is decreased by one; when the last packet is successfully transmitted (and the corresponding ACK is received), the customer leaves the OMQN, and the TCP connection closes.

From the specification of external arrival rates and service times, it is possible to derive the distribution of customers in the different queues and classes. From it we can derive the average number of customers of each class in every queue, $E[N_{q,c}]$, and the average TCP connection completion time. All performance metrics are derived directly from the distribution above, as explained in Section 5. Thus, from our modeling perspective, the solution of the OMQN is obtained by solving the system of flow balance equations

$$\lambda_{q,c} = \lambda^e_{q,c} + \sum_{i \in S} \sum_{j=c}^{N_P^{\max}} \lambda_{i,j} P(i,j;q,c) \quad \forall (q,c), \tag{1}$$

where S is the set of queues in the OMQN, and $P(i,j;q,c)$ is the transition probability from queue i in class j to queue q in class c. Note that $\lambda^{\mathrm{e}}_{q,c} = 0$ for any $q \neq q^*$. As discussed in Section 7, the banded structure of the system makes its solution computationally light.

The modeling approach based on OMQNs is capable of describing any protocol whose dynamics can be described by a finite state machine (FSM) by associating each queue with a state of the FSM. The use of classes to describe the backlog of the connection allows modeling short-lived connections.

3. The model of TCP-Tahoe

Let W be the maximum TCP window size expressed in segments and C the maximum number of retransmissions before TCP closes the connection. Fig. 2 reports the OMQN model of short-lived TCP-Tahoe connections, in the case $W = 10$, which is the smallest maximum window size that allows a complete description of all states of the protocol. New TCP connections open in the state described by queue FE_1 (thus, $q^* = FE_1$). Customers leave the system whenever they reach class 0, regardless of the queue they are in; these transitions are not shown in Fig. 2 to avoid cluttering the graph.

The main part of Fig. 2 models the TCP-Tahoe data transfer states. The inset in the lower right corner of the figure, present the additional queues that can be used to model the three-way-handshake and the associated opening timeouts, which have a different time base with respect to data timeouts. The half close procedure of TCP can be modeled in a similar way. Since the opening and closing of TCP connections have a minor impact

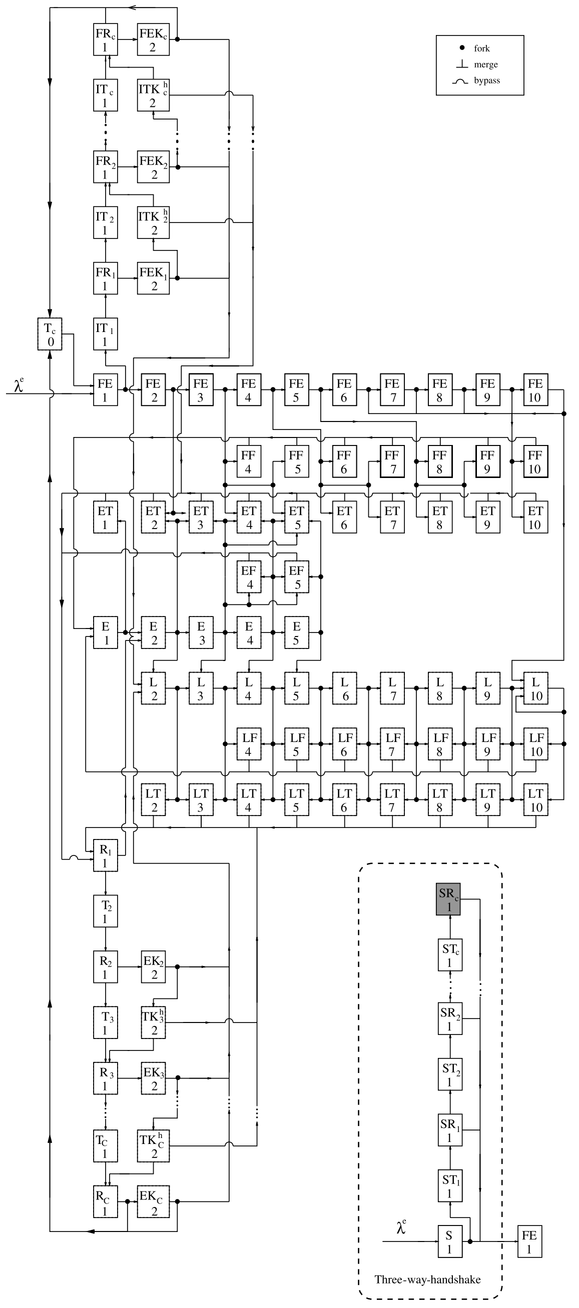

Fig. 2. The OMQN model of TCP-Tahoe; the inset presents the queues that model the three-way-handshake for connection opening.

on the protocol dynamics, we will not consider them any more in the remaining of the paper. We incidentally notice that most of the literature on TCP disregards the opening and closing phases, and also the implementation of the simulator *ns version 2* [13] that we will use to validate our model does not include it.

The queues in Fig. 2 are arranged in a matrix pattern: all queues in the same row correspond to similar protocol states, and all queues in the same column correspond to equal window size. The OMQN in Fig. 2 is highly structured; we identify 13 different types of queues, described in some detail below, assuming that the reader is familiar with TCP-Tahoe (see [14,15]). As explained below for each queue type, the names of the queues are chosen so as to recall the specific behavior of TCP; for example, *E* (*FE*) stands for (*First*) *exponential growth* of the window, i.e., the (first) slow start phase; *L* stands for *linear growth*, i.e., the congestion avoidance phase, and so on.

Queues FE_i (First Exponential growth with window i, $1 \leqslant i \leqslant W$) and E_i (Exponential growth with window i, $1 \leqslant i \leqslant W/2$) model the slow start; the index i indicates the congestion window size. Queues FE_i are visited only during the first slow start phase after the connection is opened, while queues E_i model all other slow start phases; during the first slow start phase the growth is limited only by losses, since *ssthresh* is not yet assigned. As can be seen from Table 1, the average service time at these queues depends on the RTT through a factor σ which is an apportioning coefficient that takes into account the fact that during slow start the TCP congestion window grows geometrically with base 2. As a consequence of the geometric growth, the TCP congestion window quickly skips all sizes between 2^n and 2^{n+1} while packets are transmitted, and then settles for a fairly large portion of the RTT to a dimension that is a power of 2. The correct apportioning depends on many different factors, but we empirically found that $\sigma = 2/3$ fits all the situations we explored. It must be noted that the value of σ affects only the window size distribution, as discussed in [9].

Queues ET_i (Exponential phase Timeout with window i, $1 \leqslant i \leqslant W$) model the TCP transmitter state after a loss occurred during slow start: the transmission window has not yet been reduced, but the transmitter is blocked, because its window

Table 1
Queues service times

Queue	Service time
E_1, FE_1	$\overline{\mathrm{RTT}}$
E_2, FE_2	$\overline{\mathrm{RTT}}$
E_i, FE_i, $i = 2^n$, $n \geqslant 2$,	$\sigma\overline{\mathrm{RTT}}$
E_i, FE_i, $2^{n-1} + 1 \leqslant i \leqslant 2^n - 1$	$\frac{(1-\sigma)\overline{\mathrm{RTT}}}{2^{n-1}-1}$
ET_i, $1 \leqslant i \leqslant 3$	$T_0 - \overline{\mathrm{RTT}}$
ET_i, $i \geqslant 4$	T_0
FF_i, EF_i, LF_i	$\frac{\overline{\mathrm{RTT}}}{2} + \frac{4\overline{\mathrm{RTT}}}{i}$
L_i	$\overline{\mathrm{RTT}}$
LT_2	$T_0 - \overline{\mathrm{RTT}}$
LT_i, $3 \leqslant i \leqslant W$	$T_0 - \overline{\mathrm{RTT}}/2$
T_i, IT_i, $1 \leqslant i \leqslant 7$	$2^i T_0$
T_i, IT_i, $8 \leqslant i \leqslant C$	$64T_0$
R_i, FR_i, EK_i, FEK_i	$\overline{\mathrm{RTT}}$
TK_i^1, ITK_i^1	$2^{a}T_0 - \overline{\mathrm{RTT}}$
TK_i^2, ITK_i^2	$T_0 - \overline{\mathrm{RTT}}$
T_C	180 s

[a] T_0 assumes different values whether the first packet of a connection is lost or otherwise.

is full; the combination of window size and loss pattern forbids a fast retransmit (i.e., less than three duplicate ACKs are received). The TCP source is waiting for a timeout to expire. Queues ET_i ($W/2 \leqslant i \leqslant W$) can be reached only from queues FE_i, i.e., during the first slow start. The service time is a function of the base timeout T_0 which in the model is approximated with [2] max(3 tics, 4 $\overline{\mathrm{RTT}}$).

Queues FF_i (First exponential phase Fast retransmit with window i, $4 \leqslant i \leqslant W$) and *$EF_i$ (Exponential phase fast retransmit with $4 \leqslant i \leqslant W/2$)* model a situation similar to that of queues ET_i, where, instead of waiting for the timeout expiration, the TCP source is waiting for the duplicated ACKs that trigger the fast retransmit. Different queues (FF_i) have been used to distinguish the first loss event during a connection, i.e., fast retransmit events triggered from queues FE_i, from all other cases, modeled by queues EF_i.

Queues L_i (Linear growth phase with window i, $2 \leqslant i \leqslant W$) model the linear growth during congestion avoidance (notice that queue L_1 does not exist).

Queues LF_i (Linear growth phase Fast retransmit with window i, $4 \leqslant i \leqslant W$) model losses that trigger a fast retransmit during congestion avoidance.

Queues LT_i (Linear growth phase Timeout with window i, $2 \leqslant i \leqslant W$) model the detection of losses by timeout during congestion avoidance.

Queues IT_i (Initial Timeout with backoff i, $1 \leqslant i \leqslant C$) and T_j (Timeout with backoff j, $2 \leqslant j \leqslant C$) model the time lapse before the expiration of the *i*th or *j*th timeout for the same segment, i.e., model the backoff timeouts. Queues IT_i model the backoff procedure in case the first segment is lost, when the RTT estimate is not yet available to the protocol and T_0 is set to a default value, typically 12 tics. This event, that may seem highly unlikely, has indeed a deep impact on the TCP connection duration, specially when the connection is limited to a few or even a few tens of segments.

Queue T_C models TCP connection that were closed for an excessive number of timeouts. Closed connections should leave the system; however, the model solution we use is correct only if all customers entering the OMQN in any class, leave in class 0 (a 'work conservation' principle), hence from queue T_C connections are supposed to re-open.

Queues FR_i (First packet Retransmission) and R_i (Retransmission) after ith timeout ($0 \leqslant i \leqslant C$) model the retransmission of a packet when the timeout expires.

Queues FEK_i (First Exponential phase with Karn's algorithm) and EK_i (Exponential phase with Karn's algorithm) after ith timeout ($1 \leqslant i \leqslant C$) model the first stage of the slow start phase (i.e., the transmission of the first two non-retransmitted packets) after a backoff timeout. During this phase the Karn algorithm [16,14, Chapter 21.3] has a deep impact on the protocol performance under heavy load.

Queues ITK_i^h (Initial Timeout when Karn's algorithm is active) and TK_i^h (Timeout when Karn's algorithm is active) after the ith timeout ($1 \leqslant i \leqslant C; h = 1, 2$) model the wait for timeout expiration when losses occurred in queues FEK_i (EK_i); the superscript *h* in Fig. 2 discriminates two queues that are drawn together, but have different service times. The one with $h = 1$ is entered when

[2] The 'tic' is the time granularity of the TCP protocol, i.e., the minimum amount of time that separates non-self-clocked protocol operations.

the first packet of the pair transmitted in queues FEK_i (EK_i) is lost, the other when the lost packet is the second. The transition from ITK_i^1 (TK_i^1) is to queue FR_i (R_i), while the transition from ITK_i^2 (TK_i^2) is to queue R_1.

The average service times (reported in Table 1) represent the time a typical connection spends in a given protocol state. The service times derive directly from the protocol properties and are mainly function of the average RTT, $\overline{\text{RTT}}$. Service times are independent from the customer class.

The transition probabilities between queues derive from the TCP dynamics and characteristics, and they are related mainly to the packet loss probability. It is well known (see for instance [3,9]), that, especially under droptail queue management schemes, the TCP dynamics introduce correlations in packet losses, and these have a deep impact on performance and consequently on any modeling process. On a temporal axis, the correlation extends for roughly a RTT due to the protocol dynamics. In fact, whenever a loss occurs, there is already a whole window of unacknowledged information in the transmission pipe. [3] ACKs relative to these packets sustain the transmission of new packets for about a RTT before the protocol reacts to the loss event. All the transmitted packets during this period contribute to the congestion phase, thus introducing correlation in the loss process. The silence period (one RTT in fast retransmit or a timeout) following the loss detection disrupts the correlation on a longer time scale. We introduce the following quantities:

- P_L: average packet loss ratio, $P_S = 1 - P_L$;
- P_{L_f}: loss ratio for the *first* packet in a window; this is also the probability of losing a burst of packets; $P_{S_f} = 1 - P_{L_f}$;
- P_{L_a}: packet loss ratio *after* the first packet in a window is lost; this is the packet loss ratio within a burst, and takes into account the correlation; $P_{S_a} = 1 - P_{L_a}$.

[3] The "pipe" is the amount of information that can be stored along the forward (segments) and backward (ACKs) transmission paths. It is obviously related to the bandwidth-delay product, though it may also include buffering at nodes.

The relationship between P_L, P_{L_f} and P_{L_a}, as well as their derivation and rationale are discussed in Section 4.2.

Tables 2–5 report the transition probabilities $P(q_i, c; q_j, c-k)$ from source queue q_i (first column) to destination queue q_j (second column), decreasing the customer class of k units (third column) under the conditions indicated in the last column; $\wedge$ is the logical AND operator.

The probability of all transitions following the correct transmission of a group of packets are rather straightforward to compute. They are reported in Table 2. The only modeling problem arises in transitions from exponential to linear growth, where the value of *ssthresh* is crucial. Unfortunately, the value of *ssthresh* does not depend on the current state, but on the state when the last packet was lost. Explicitly taking into account the value of *ssthresh* would lead to the explosion of the number of queues needed in the model. We resorted to a stochastic description of the distribution of *ssthresh*, that is taken into account by the terms

$$Z_{i,j}^{\mathrm{T}} = \frac{P_{\mathrm{T}}(i)}{1 - P_{\mathrm{C}}(j)} \quad \text{and} \quad Z_{i,j}^{\mathrm{C}} = \frac{1 - P_{\mathrm{C}}(i)}{1 - P_{\mathrm{C}}(j)},$$

Table 2
Transition probabilities modeling successful transmission events

q_i	q_j	k	$P(\cdot)$	Condition
FE_1	FE_2	1	P_{S_f}	
FE_i	FE_{i+1}	2	$P_{S_f}^2$	$2 \leqslant i \leqslant W-1$
FE_W	L_W	2	$P_{S_f}^2$	
FEK_i	FE_3	2	$P_{S_f}^2$	$2 \leqslant i \leqslant C$
E_1	E_2	1	$P_{S_{fc}}$	
E_i	E_{i+1}	2	$P_{S_{fc}}^2 Z_{i+1,i}^{\mathrm{C}}$	$2 \leqslant i < W/2$
	L_i	2	$P_{S_{fc}}^2 Z_{i,i}^{\mathrm{T}}$	
L_i	L_{i+1}	i+1	$P_{S_f}^{i+1}$	$2 \leqslant i \leqslant W-1$
L_W	L_W	W	$P_{S_f}^W$	
R_1	E_2	1	$P_{S_{fc}}$	
EK_i	L_2	2	$P_{S_{fc}}^2 Z_{2,2}^{\mathrm{T}}$	$2 \leqslant i \leqslant C$
	E_3	2	$P_{S_{fc}}^2 Z_{3,2}^{\mathrm{C}}$	

Table 3
Transition probabilities modeling the exponential backoff and Karn's algorithm

q_i	q_j	k	$P(\cdot)$	Condition
FR_i	IT_{i+1}	0	P_{L_f}	$1 \leqslant i \leqslant C-1$
	FEK_i	1	P_{S_f}	$1 \leqslant i \leqslant C$
IT_i	FR_i	0	1	$1 \leqslant i \leqslant C$
FR_C	T_C	0	P_{L_f}	
FEK_i	ITK^2_{i+1}	1	$P_{S_f}P_{L_f}$	$1 \leqslant i \leqslant C-1$
	ITK^1_{i+1}	0	P_{L_f}	
FEK_C	T_C	0	P_{L_f}	
ITK^1_i	FR_i	j	$P^j_{S_a}P^{1-j}_{L_a}$	$(0 \leqslant j \leqslant 1) \wedge (2 \leqslant i \leqslant C)$
ITK^2_i	R_1	2	$P^2_{S_a}$	
	R_1	j	$P_{L_a}P^j_{S_a}$	$(0 \leqslant j \leqslant 1) \wedge (2 \leqslant i \leqslant C)$
EK_i	TK^2_{i+1}	1	$P_{S_{fc}}P_L$	$1 \leqslant i \leqslant C-1$
	TK^1_{i+1}	0	P_L	
EK_C	T_C	0	P_L	
TK^1_i	R_i	j	$P^j_{S_a}P^{1-j}_{L_a}$	$(0 \leqslant j \leqslant 1) \wedge (3 \leqslant i \leqslant C)$
TK^2_i	R_1	2	$P^2_{S_a}$	
	R_1	j	$P_{L_a}P^j_{S_a}$	$(0 \leqslant j \leqslant 1) \wedge (3 \leqslant i \leqslant C)$
R_1	T_2	0	P_L	
R_i	EK_i	1	$P_{S_{fc}}$	$2 \leqslant i \leqslant C$
	T_{i+1}	0	P_L	$2 \leqslant i \leqslant C-1$
T_i	R_i	0	1	$2 \leqslant i \leqslant C$
R_C	T_C	0	P_L	
T_C	FE_1	0	1	

Table 4
Transition probabilities corresponding to loss events during the first slow start phase

q_i	q_j	k	$P(\cdot)$	Condition
FE_1	IT_1	0	P_{L_f}	
FE_2	ET_2	0	P_{L_f}	
	ET_3	1	$P_{S_f}P_{L_f}$	
FE_i	ET_j	0	$P_{L_f}P'_{\mathrm{to}}(j,c)$	$(3 \leqslant i \leqslant W/2) \wedge$
	FF_j	0	$P_{L_f}P'_{\mathrm{ft}}(j,c)$	$[j = 2(i-1)]$
	ET_j	1	$P_{S_f}P_{L_f}P'_{\mathrm{to}}(j,c)$	$(3 \leqslant i \leqslant W/2) \wedge$
	FF_j	1	$P_{S_f}P_{L_f}P'_{\mathrm{ft}}(j,c)$	$(j = 2i-1)$
	ET_W	0	$P_{L_f}P'_{\mathrm{to}}(W,c)$	$\frac{W}{2}+1 \leqslant i \leqslant W$
	FF_W	0	$P_{L_f}P'_{\mathrm{ft}}(W,c)$	
	ET_W	1	$P_{S_f}P_{L_f}P'_{\mathrm{to}}(W,c)$	$W/2 \leqslant i \leqslant W$
	FF_W	1	$P_{S_f}P_{L_f}P'_{\mathrm{ft}}(W,c)4$	
FF_i	E_1	j	1	$4 \leqslant i \leqslant W \wedge j = 3 + P_{S_a}(i-4)$

$P'_{\mathrm{to}}(x,y) = P_{\mathrm{to}}(x-1,y-1)$; $P'_{\mathrm{ft}}(x,y) = P_{\mathrm{ft}}(x-1,y-1)$.

where $P_{\mathrm{T}}(i)$ is the probability that the exponential growth threshold *ssthresh* has value i and $P_{\mathrm{C}}(i)$ is the cumulative distribution of $P_{\mathrm{T}}(i)$. Though already derived in [9], the explanation of the computation of the distribution of *ssthresh* is reported in Appendix A for ease of reference.

Also transitions modeling the exponential backoff and Karn's algorithm, either following the loss of the first packet in the connection (queues IT_i, FR_i, ITK^h_i, and FEK_i) or the loss of any other packet (queues T_i, R_i, TK^h_i, and EK_i), are easily computed from the protocol dynamics. The corresponding transition probabilities are reported in Table 3, except for those deriving from the conclusion of a backoff event, that are reported in Table 2, since they correspond to the correct transmission of two consecutive new packets.

Transitions that are triggered by losses require instead a careful inspection of the possible loss patterns, since they influence both the possibility of fast retransmit or timeout, and the number of packets that are correctly transmitted thus contributing to the customer class decrease. In addition, the customer class influences the fast retransmit probability; indeed, when one of the last three packets of a session is lost, the number of ACKs is not large enough to trigger a fast retransmit, and the loss can be detected only with a timeout. It follows that the timeout and fast retransmit probabilities are functions of the window size, the loss pattern and the class of the customer. Let i be the number of packets that TCP can send after the loss of a segment, i.e., the window size at the instant of the loss detection minus one, and h be the number of packets that are indeed transmitted, which is a function of the customer class; then the timeout probability is the probability that less than three ACKs reach the transmitter

$$P_{\mathrm{to}}(i,h) = \begin{cases} 1, & h \leqslant 2, \\ \sum_{j=0}^{2} \binom{h}{j} P_{L_a}^{h-j} P_{S_a}^{j}, & 3 \leqslant h \leqslant i, \\ \sum_{j=0}^{2} \binom{i}{j} P_{L_a}^{i-j} P_{S_a}^{j}, & h \geqslant i, \end{cases} \qquad (2)$$

Table 5
Transition probabilities corresponding to loss events during steady state operation

q_i	q_j	k	$P(q_i, c; q_j, c-k)$	Condition
E_1	ET_1	0	P_L	
E_2	ET_2	0	P_L	
	ET_2	1	$P_{S_{fc}} P_L Z^{\mathrm{T}}_{2,2}$	
	ET_3	1	$P_{S_{fc}} P_L Z^{\mathrm{C}}_{3,2}$	
E_i	ET_j	0	$P_L Z^{\mathrm{T}}_{j,i} P_{\mathrm{to}}(j-1, c-1)$	$(3 \leqslant i \leqslant W/4) \wedge$
	EF_j	0	$P_L Z^{\mathrm{T}}_{j,i} P_{\mathrm{ft}}(j-1, c-1)$	$(i \leqslant j \leqslant W/2), j \neq 2(i-1)$
	ET_j	0	$P_L Z^{\mathrm{C}}_{j,i} P_{\mathrm{to}}(j-1, c-1)$	$(3 \leqslant i \leqslant W/4) \wedge$
	EF_j	0	$P_L Z^{\mathrm{C}}_{j,i} P_{\mathrm{ft}}(j-1, c-1)$	$[j = 2(i-1)]$
	ET_j	1	$P_{S_{fc}} P_L Z^{\mathrm{T}}_{j,i} P_{\mathrm{to}}(j-1, c-1)$	$(3 \leqslant i \leqslant W/4) \wedge$
	EF_j	1	$P_{S_{fc}} P_L Z^{\mathrm{T}}_{j,i} P_{\mathrm{ft}}(j-1, c-1)$	$(i \leqslant j \leqslant W/2), j \neq 2i-1$
	ET_j	1	$P_{S_{fc}} P_L Z^{\mathrm{C}}_{j,i} P_{\mathrm{to}}(j-1, c-1)$	$(3 \leqslant i \leqslant W/4) \wedge$
	EF_j	1	$P_{S_{fc}} P_L Z^{\mathrm{C}}_{j,i} P_{\mathrm{ft}}(j-1, c-1)$	$(j = 2i-1)$
L_2	LT_2	0	P_{L_f}	
	LT_3	1	$P_{S_f} P_{L_f}$	
	LT_3	2	$P^2_{S_f} P_{L_f}$	
L_3	LT_3	0	P_{L_f}	
L_i	LT_i	0	$P_{L_f} P'_{\mathrm{to}}(i, c)$	$4 \leqslant i \leqslant W-1$
	LF_i	0	$P_{L_f} P'_{\mathrm{ft}}(i, c)$	
	LT_{i+1}	j	$P^j_{S_f} P_{L_f} P_{\mathrm{to}}(i-1, c-1-j)$	$4 \leqslant i \leqslant W-1 \wedge$
	LF_{i+1}	j	$P^j_{S_f} P_{L_f} P_{\mathrm{ft}}(i-1, c-1-j)$	$1 \leqslant j \leqslant i$
L_W	LT_W	j	$P^j_{S_f} P_{L_f} P_{\mathrm{to}}(W-1, c-1-j)$	$1 \leqslant j \leqslant W-1$
	LF_W	j	$P^j_{S_f} P_{L_f} P_{\mathrm{to}}(W-1, c-1-j)$	
ET_2	R_1	j	$P^{1-j}_{L_a} + P^j_{S_a}$	$0 \leqslant j \leqslant 1$
LT_2	R_1	j	$P^{1-j}_{L_a} + P^j_{S_a}$	$0 \leqslant j \leqslant 1$
ET_i	R_1	j	$P_{L_a} P^j_{S_a}$	$3 \leqslant i \leqslant W \wedge 0 \leqslant j \leqslant 1$
	R_1	2	$P^2_{S_a}$	$3 \leqslant i \leqslant W$
LT_i	R_1	j	$P_{L_a} P^j_{S_a}$	$3 \leqslant i \leqslant W \wedge 0 \leqslant j \leqslant 1$
	R_1	2	$P^2_{S_a}$	$3 \leqslant i \leqslant W$
EF_i	R_1	j	1	$4 \leqslant i \leqslant W/2 \wedge j = 3 + P_{S_a}(i-4)$
LF_i	E_1	j	1	$4 \leqslant i \leqslant W \wedge j = 3 + P_{S_a}(i-4)$

with $3 \leqslant i \leqslant W$. Its complement ($P_{\mathrm{ft}}(i,h) = 1 - P_{\mathrm{to}}(i,h)$) is the fast retransmit probability. Transitions corresponding to losses during the first slow start phase are reported in Table 4, while those relative to steady-state operations are listed in Table 5.

During slow start, we consider separately the case when the first segment sent with window size i is lost, and the case when the second segment is lost. In the first case, the term P_{L_f} appears in the transition probabilities, while in the second case the expression of the transition probabilities contains $P_{S_f}P_{L_f}$.

Once P_{L_f}, P_{L_a} and $P_{\mathrm{to}}(i,h)$ are known, the derivation of the transition probabilities relating to losses during the first slow start phase (transitions from queues FE_i in Table 4) and during congestion avoidance (transitions from queues L_i in Table 5) is simple. Transitions due to losses during slow start are influenced by the loss pattern and the *ssthresh* distribution, that define how much the transmission window is opened before the transmitter detects the loss, and hence the destination queue.

3.1. Packet generation and network load

A TCP connection in a given queue q and class c generates a number of packets $\Pi_{q,c}$, which is the minimum between the number of packets allowed by the protocol state, Π_q, and the remaining packets to be transmitted during the TCP session, $\Pi_{q,c} = \min(\Pi_q, c)$. It follows that, from each queue q of the OMQN model, the load offered to the network by a TCP connection is

$$\Lambda_q = \sum_{c=1}^{N_P^{\max}} \lambda_{q,c}\Pi_{q,c}, \tag{3}$$

where $\lambda_{q,c}$ is the arrival rate at queue q in class c computed from the OMQN solution. The total load offered to the network is then

$$\Lambda = \sum_{q \in S} \Lambda_q. \tag{4}$$

To solve (3) and (4) we need to define Π_q for each $q \in S$.

Some queues represent states of the protocol in which no packet is generated: queues T_i and IT_i, which model the backoff timeouts, and queue ET_1, which stands for the timeout expiration delay when the only packet in the window has been lost.

One packet per service is generated in queues R_i and FR_i, which model the retransmission of a packet when the timeout expires, and in queues FE_1 and E_1, where the TCP window size equals 1.

The generation of two packets in queues FE_i and E_i, with $i > 1$, is due to the exponential window size increase in slow-start mode. Two packets per service are generated also in queues EK_i and FEK_i, standing for the transmission of the first two non-retransmitted packets after a backoff timeout. Similarly, two packets are generated in queues TK_i^3 and ITK_i^3.

More complex is the derivation of the number of packets whose generation is allowed by the protocol for each service at queues EF_i, ET_i and FF_i, for $i \geqslant 3$, since it depends on which packet was lost in the previously visited queue. It results: $\Pi_{EF_i} = \Pi_{ET_i} = \Pi_{FF_i} = \Pi_i$,

$$\Pi_i = \left[(i-2)\frac{P_{ll}}{1-P_{ss}^2} + (i-1)\frac{P_{ss}P_{ll}}{1-P_{ss}^2} \right], \tag{5}$$

where P_{ll} (P_{ss}) assumes the values P_{L_f} or P_L (P_{S_f} or $P_{S_{fc}}$), depending on the previously visited queue (see Section 4.2). Instead, the load offered by the connections in queues L_i is

$$\Pi_{L_i} = [P_{L_f} i + (1-P_{L_f})(i+1)], \quad i \leqslant W-1, \tag{6}$$

$$\Pi_{L_W} = W. \tag{7}$$

The computation of the load offered to the underlying IP network by connections at individual queues of type F_i and LT_i is cumbersome, since for each queue it depends on the loss pattern. On the contrary, if we consider the aggregate load collectively offered by the set of queues

$$\Phi_{q_{ll}} = \bigcup_i [F_i \cup LT_i], \tag{8}$$

then its computation is much easier

$$\Lambda_{\Phi_{q_{ll}}} = \sum_{i=2}^{W} \left[\sum_{j=1}^{i} j P_{S_f}^j P_{L_f} \right] \lambda_{L_i}. \tag{9}$$

Finally, for queues TK_i^2, ITK_i^2 and ET_2, we can write

$$\Pi_{TK_i^2} = \Pi_{ITK_i^2} = \Pi_{ET_2} = \frac{P_{ss}P_{ll}}{P_{ll} + P_{ss}P_{ll}}. \qquad (10)$$

3.2. Duplicate packets and class reduction

Customers reduce their class upon the correct transmission of packets. Unfortunately, when the loss probability is high, TCP-Tahoe sometimes retransmits packets that were already successfully received; these duplicate packets should not contribute to the customer class reduction.

Directly taking into account this phenomenon in a model is very difficult; however, we may approximate its effect as follows, splitting the cases when the loss is detected by timeout or fast retransmit. Let i be the TCP window size when a transmitter in class c detects the loss with a fast retransmit. Then, following our model, $k(i,c) = \max[3, P_{S_a}(\min(i,c) - 1)]$ packets are successfully transmitted after the lost one. Since we cannot compute which of these packets will be unnecessarily retransmitted, we assume that only $k(i,c)^{\epsilon}$ packets, with $\epsilon \leqslant 1$, contribute to the class reduction. Though it might be possible to find more accurate ways to represent the phenomenon by analyzing loss patterns from real traces or simulations, or simply find a direct method to set the value of ϵ, we decided to tune it empirically, finding that $\epsilon = 0.9$ fits most scenarios. If the loss is detected with a timeout, then $k(i,c) \in [0,1,2], i \geqslant 4$, and we chose to simply set $k(i,c) = 1$.

4. Modeling the underlying IP network

As sketched in Fig. 1, the proposed modeling technique is based on the separation of the TCP model (represented with one or more OMQNs) from the model of the underlying IP network.

Since the focus in this paper is on TCP, the IP network model is kept as simple as possible. Nevertheless, it must allow the correct estimation of the key parameters that drive TCP performance.

A single server queue is the obvious modeling choice for each router interface in the IP network. The network is modeled with a set of $M^{[D]}/M/1/B$ queues, where packets arrive in batches whose size varies between 1 and W with distribution $[D]$. The variable size batches model the burstiness of the TCP transmission within $\overline{\text{RTT}}$. The Markovian arrival of the batches finds its reason mainly in the Poisson assumption for the TCP connections arrival process, as well as in the fairly large number of connections present in the model. The Markovian assumption about service times is more difficult to justify, since the transmission time of fixed length packets is constant; however, the influence on results of using an $M^{[D]}/D/1/B$ queue was observed not to be worth the increased complexity.

4.1. Evaluation of $\overline{RTT}$

The average queuing time t_B is obtained directly from the solution of the $M^{[D]}/M/1/B$ queue; $\overline{\text{RTT}}$ is then estimated as the sum of t_B, the average two-way propagation delay of connections, and the packet transmission time in routers. The key point of the modeling process is the determination of the batch size distribution $[D]$. We compute the batch sizes starting from the number of segments N_R sent by a TCP transmitter during $\overline{\text{RTT}}$. This is clearly a gross approximation and a pessimistic assumption, since TCP segments are not transmitted together, and they do not arrive together at the router buffers. To balance this pessimistic approximation, the batch size is reduced by a factor $\mu < 1$. Actually, since the TCP burstiness is much higher during slow start than during congestion avoidance, we use two different factors: μ^{e} during the exponential window growth, and μ^{l} during the linear window growth. Unfortunately, it was not possible to find a direct method for the computation of these two factors, but a simple heuristic optimization led to the choice $\mu^{\text{e}} = 2/3$, $\mu^{\text{l}} = 1/3$, that yields satisfactory results in every tested scenario.

The computation of N_R is straightforward, starting from the solution of the OMQN of Fig. 2. For every queue whose service time is $\overline{\text{RTT}}$, $N_R = \Pi_q$. In the other cases, queues must be grouped, based on the protocol dynamics. For instance, during exponential growth the grouping is based on the window doubling,

$$N_R = \sum_{i=2^j+1}^{2^{(j+1)}} \Pi_{E_i}. \tag{11}$$

All other cases are computed similarly, but have a minor impact on performance, since this latter is dominated by the large values of N_R, due to large window size.

Besides the computation of batch sizes, it is also necessary to evaluate the generation rates of batches. For reasons that will be clear in Section 4.2, we separate the generation of bursts during the first slow start phase and during congestion avoidance, the relative rate being $\lambda_{bf}(i)$, from the generation of bursts during steady-state slow start phases, whose rate is $\lambda_{bc}(i)$. $\lambda_{bf}(i)$ and $\lambda_{bc}(i)$ are different for each session length, since, specially for short sessions, the number of packets to be transferred influences the probability that a connection visits a given queue (e.g., a connection that must transfer 10 packets cannot transmit a packet with window size larger than 6).

4.2. Evaluation of loss probabilities

As queue management schemes, we consider both RED and droptail.

In the case of RED, given a value of the average buffer occupancy, the loss probability is actively applied to segments according to the considered RED profile. In our model, we compute the average buffer occupancy from the network model, i.e., from the $M^{[D]}/M/1/B$ queues, and we derive the corresponding loss probability in the RED profile. Losses are then assumed to be uncorrelated. In the OMQN model of TCP, this implies that the considered loss probabilities, P_L, P_{L_f} and P_{L_a}, are all the same.

Due to correlation among losses, the case of droptail is, instead, more complex. While the average packet loss ratio P_L can be easily computed from the steady-state solution of the $M^{[D]}/M/1/B$ queue, the probabilities P_{L_f} and P_{L_a} require loss correlations to be taken into account.

Loss correlations in TCP were already studied in [3,9], but in the context of long-lived TCP connections. Both works hint to the possibility that the loss probability within the window of the first lost packet is almost constant. In [3], the authors assume that all packets are lost after the first one; the first lost packet is assumed to follow a Bernoulli distribution within the window, leading to an average loss probability equal to 0.5 within a window with a burst of losses. In [9] we made a different assumption, linking the loss ratio after the first loss to the average window size, i.e., setting $P_{L_a} = \overline{w}P_{L_f}$, and assuming that the first lost packet is always shifted at the beginning of the transmission window. However, if we consider that $\overline{w}$ decreases with the loss ratio, P_{L_a} may indeed be almost constant. To verify this hypothesis, we measured P_{L_a} with the *ns-2* simulator, that resulted almost constant for several simulation scenarios and connection durations, with small variations in extreme situations. The value that seems to best fit a general scenario is $P_{L_a} = 0.2$. Given P_{L_a}, P_{L_f} is computed by imposing that the average loss probability is equal to P_L.

Since an accurate estimation of the average buffer occupancy is easier to obtain than the droptail loss probability and its correlation, the case of droptail buffers is more critical than the case of RED buffers. For this reason, in order to prove the power of our approach, in deriving analytical results we will mainly focus on droptail queue management.

5. Computation of the completion times

In order to compute the average connections completion time, let us assume that all customers enter the OMQN with the same class N_P. The average time Θ spent in the OMQN is derived from Little's theorem $\Theta = N/\lambda^e$, where N is the total average number of customers in the OMQN, and λ^e is the external arrival rate of customers.

When the initial class of the arriving customers is not constant, Little's result computes the average over all connections, regardless of the initial class. Class N_P customers comprise connections entering the OMQN with class N_P as well as all those connections whose initial class is larger than N_P, but still have N_P packets to transmit. Yet, the traffic mix influences the overall network performance, hence we devised a solution based on a three-step solution of the OMQN model.

Step 1. The model is solved with the chosen external arrival distribution $\lambda^{e}[\gamma_c]$. During this step, parameters such as $\overline{\mathrm{RTT}}$, P_L, P_{L_f} are computed.

Step 2. Keeping constant the parameters computed in step 1, the OMQN model is solved again for each external arrival class $N_P = 1, \ldots, N_P^{\max}$. During this step the *ssthresh* distribution is computed again, since the one obtained in step 1 is not representative of each input class (e.g., a connection with 10 packets to transmit cannot have *ssthresh* $= 20$), but only of their average. For the same reason $\lambda_{bf}(c)$, $\lambda_{bc}(c)$, P_{L_f} and P_{L_a} are also recomputed. Finally, Little's result is used to compute each input class latency.

Step 3. If desired, the average completion time (over all classes) can be computed by averaging the latency of each class computed at Step 2.

It must be observed that the availability of estimates for the average transfer time of files of a given length also provides estimates for the goodput of short-lived TCP connections, by just computing the product of the file size in segments times the inverse of the average file transfer time.

6. The OMQN model of TCP-NewReno

Up to this point we have described the modeling process in the case of TCP-Tahoe. We now shortly discuss the modeling of TCP-NewReno, focusing on differences with respect to Tahoe. First of all, notice that the network model remains unchanged, as well as the correlation models and the computation of $\overline{\mathrm{RTT}}$, P_L, and completion times.

Fig. 3 reports the OMQN model of TCP-NewReno. We briefly discuss here the meaning of queues that differ from the Tahoe model, while service times and transition probabilities are reported in Appendix B. The queues that are new or different from Tahoe are shaded in Fig. 3, and we focus on these. The arrangement of queues in rows is slightly different from Fig. 2, in order to allow a more linear drawing of transitions.

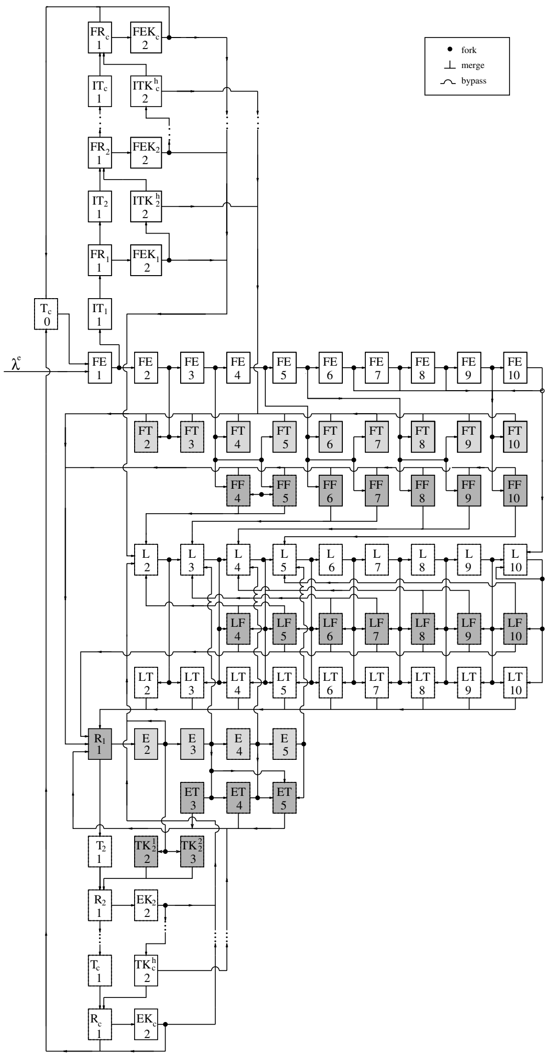

Fig. 3. The OMQN model of TCP-NewReno.

The main difference between Tahoe and NewReno lies in the presence of the fast recovery procedure [15], which avoids the slow start phase if losses are detected via duplicated ACKs, entering congestion avoidance after halving the congestion window. It follows that queues describing the initial slow start, congestion avoidance and timeouts are all equal in the two models. Differences concentrate in the queues describing the fast recovery in NewReno instead of the simpler fast retransmit in Tahoe, plus some additional modifications in slow starts following timeouts during normal operation. Notice that queues EF_i $(4 \leqslant i \leqslant W/2)$ are not present in the NewReno model. These queues modeled fast retransmits after a loss during slow start in TCP-Tahoe. In NewReno, after the first slow start, the AIMD (additive increase, multiplicative decrease) congestion control should completely avoid additional slow starts, unless there are not enough ACKs in the pipe to start a fast recovery, so that the loss is detected with a timeout. Hence, steady-state operation slow starts are fairly rare, and are only consequence of a very small window or a fast recovery that was not correctly concluded. In this second case, as explained, TCP-NewReno avoids additional fast recoveries to escape pathological conditions that may bring the protocol to a stall. [4]

Queues FF_i (First Fast recovery) $(4 \leqslant i \leqslant W)$ model losses that trigger a fast recovery during the first slow start. The transitions from these queues are directed to congestion avoidance queues $(FF_i \rightarrow L_j, j = \lfloor i/2 \rfloor)$ if the fast recovery is completed correctly, otherwise there is a timeout and the transition is always to queue R_1. The main difference with respect to the analogous queues in the Tahoe model lies in the different service times (see Appendix B) and the load offered to the network, that, besides the retransmitted packet, takes into account the new packets that can be injected in the network during fast recovery.

Queues LF_i (Linear growth Fast recovery) $(4 \leqslant i \leqslant W)$ model losses that trigger a fast recovery during congestion avoidance; the same considerations made for queues FF_i apply here too.

Queues FT_i (First Timeout) $(2 \leqslant i \leqslant W)$. These queues substitute the queues ET_i in the Tahoe model and have basically the same meaning. The main difference is that queues ET_i in Tahoe model timeouts that occur in *any* slow start phase, while in NewReno they only model the first timeout (hence the change of notation) in the initial slow start. For this reason queue FT_1 does not exist, since the loss of the fist packet of the connection is modeled with the queue IT_1.

Queue R_1 (Retransmission) models the retransmission of a packet after a loss during steady-state operation. It takes the role of both queues R_1 and E_1 in the Tahoe model.

Queues E_i (Exponential growth) $(2 \leqslant i \leqslant W/2)$. The only difference with the corresponding queues in Tahoe is that a packet loss while in these queues is necessarily recovered with a timeout in NewReno, thus the transitions are either to L queues or to ET queues.

Queues ET_i (Exponential growth Timeout) $(2 \leqslant i \leqslant W/2)$. These queues model the wait for timeout to expire when a packet is lost during a steady-state slow start. Notice that the queue ET_1 is missing, since the loss of the first new segment sent after a timeout triggers the Karn algorithm, which is taken into account by queues TK.

Queues TK_2^1 and TK_2^2 (Timeout with Karn's algorithm). First of all these queues take into account the Karn's algorithm considering the longer timeout when multiple packets are retransmitted. Besides, together with E_i and ET_i these queues model the *bugfix* algorithm mentioned above.

7. Solution complexity

The complete model solution is obtained by iterating the solutions of the network sub-model and the TCP sub-models with a fixed point algorithm (FPA), until convergence of the FPA is reached, according to a specified threshold. The network sub-model solution is very simple; thus, the complexity of each step of the iterative procedure is dominated by the solution of the TCP OMQN model. We do not have a formal proof of the convergence of the FPA with this model; however, we never encountered a case in which convergence was not reached. Besides, the convergence of FPA with models of greedy TCP sources under some mild restricting conditions was proved in [10].

All the performance figures of interest depend on the average distribution of customers in the queues, which is obtained by solving the flow balance problem defined by (1). The number of

[4] The algorithm adopted to do so is informally known as "*bugfix*," name used in BSD based releases and in IETF RFC2589, and marks the main difference between Reno and NewReno.

flow balance equations is equal to $M_q \cdot N_P^{\max}$, where M_q is the number of queues in the OMQN and $N_P^{\max}$ denotes the maximum allowed file size in packets. By exploiting the banded structure of the system of linear equations, due to the fact that on transitions the customers can only keep their class unchanged or decrease it by a value which never exceeds the maximum window size W, the complexity of the solution results to be $O(M_q \cdot N_P^{\max} \cdot W)$. Since M_q is of the order of a few hundreds, the CPU time required for each step of the FPA is extremely small. The number of iterations before convergence depends on the accuracy required for the FPA; a relative accuracy of 10^{-6} is generally reached in a number of iterations which varies between a few tens and a few hundreds.

For instance, if the TCP maximum segment size (MSS) is 1024 bytes, the advertised receiver window size is 64 kbytes and $C = 16$ (the case we use for the model validation in the next section), then $W = 64$ and $M_q = 594$ queues. Using $P_L = 0.01$ and $P_T(i) = 1/(W/2 - 1)$ as initial values of the FPA, and setting the accuracy threshold of the FPA to 10^{-6}, the number of iterations ranges from 20–30 for the simplest cases, to 200–300 for the most critical cases (where the external load approaches 1). Correspondingly, the CPU time ranges between a few seconds up to no more than one minute on standard PC hardware.

8. Validation and results

In order to validate our analytical model of TCP, we compare the performance predictions obtained from the OMQN model solution against point estimates and 95% confidence intervals obtained from very detailed simulation experiments. The tool used for simulation experiments is *ns version 2* [13]; confidence intervals were obtained with the "batch means" technique, using 30 batches. Simulations last for 1000–2000 s when the bottleneck is a 45 Mbit/s link, and are four times longer when the bottleneck is a 10 Mbit/s link. Results with faster links can be quite easily obtained with the model, but comparisons against simulation become difficult, because simulation runs are exceedingly long.

8.1. Network topology and traffic load

We chose a networking environment which closely resembles the actual path followed by Internet connections from our University LAN to Web sites in Europe and the USA, where two clearly distinct traffic patterns can be identified.

The topology of the network we consider is shown in Fig. 4; at the far left we can see a set of terminals connected to the internal LAN of Politecnico di Torino. These terminals are the clients of the TCP connections we are interested in (white circles in the figure represent TCP clients; gray circles represent TCP servers). The distance of these clients from the Politecnico router is assumed to be uniformly distributed between 1 and 10 km. The LAN of Politecnico is connected to the Italian research network, named GARR-B/TEN-155, through a 10 Mb/s link whose length is roughly 50 km (this link will be called POLI-TO). Internally, the GARR-B/TEN-155 network comprises a number of routers and 155 Mb/s links. One of those connects the router in Torino with the router in Milano; its length is set to 100 km. Through the GARR-B/TEN-155 network, clients at Politecnico can access a number of servers, whose distance from Politecnico di Torino is assumed to be uniformly distributed between 100 and 9900 km. From Milano, a 45 Mb/s undersea channel whose length is about 5000 km reaches New York, and connects GARR-B to North-American Internet backbones (this link will be called MI-NY). Many other clients use the router in Milano to reach servers in the US. The distance of those clients

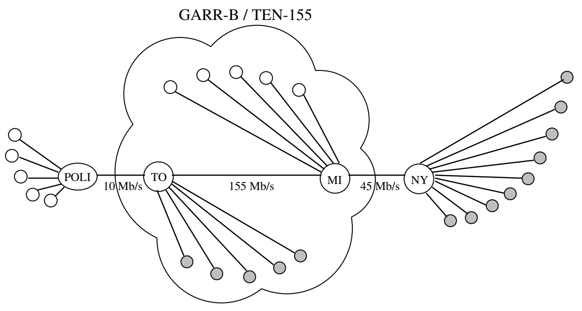

Fig. 4. Abstract view of the Internet from the Politecnico di Torino LAN; servers are shaded, clients are white.

from Milano is assumed to be uniformly distributed between 200 and 2800 km. The distance of servers in the US from the router in NY is assumed to be uniformly distributed between 200 and 3800 km.

At present, the GARR-B/TEN-155 network is over-dimensioned, thus it is never congested. Both the POLI-TO and the MI-NY link, instead, are bottleneck channels that often incur heavy and persistent congestion. From the abstract network represented in Fig. 4 we carve out the following three different traffic patterns that give rise to the three scenarios we name US Browsing (US Browsing), Local Access (Local Access), and 2 Bottlenecks (2 Bottlenecks).

US Browsing: This scenario becomes relevant when the traffic pattern makes the MI-NY link the bottleneck of the system. When this is the case, we have connections with lengths between 5400 and 11,600 km that compete for the 45 Mbit/s link.

Local Access: This scenario corresponds to moments when the access link of our University is the bottleneck of the system. Connections compete for the 10 Mbit/s on the POLI-TO link, and their lengths are distributed between 151 and 9960 km.

2 *Bottlenecks*: This latter scenario, finally, corresponds to time periods when both MI-NY and POLI-TO links are overloaded. The IP network model in this case must account for both routers, so that correlations at the flow level can be captured. The network model is however still Markovian, so that correlations at the packet level are not described.

The packet size is 1024 bytes; the maximum window size is 64 packets. We consider the cases of buffer sizes equal to either 128 or 64 packets, and TCP tic equal to 500 ms. The amount of data to be transferred by each connection (i.e., the file size) is expressed in number of segments. We consider three different cases. In the first two, all connections use the same TCP version, either Tahoe or NewReno, and the length of connections is mixed: 50% of the connections are 10 segments long, 40% are 20, and 10% are 100. The third scenario encompasse as different mixes of Tahoe and NewReno, but TCP connections are assumed to transfer constant size files. The size ranges from 10 to 200 segments.

We initially consider droptail buffers, but in a later section we also discuss results for RED buffers.

8.2. Numerical results for TCP-Tahoe

Fig. 5 reports the average packet loss probability computed with the model, as well as point estimates and confidence intervals obtained via simulation in both the US Browsing and Local Access scenarios. Results are plotted versus the normalized external load, which is the load the network would have if no packets were lost, so that no retransmissions are necessary. The upper curve refers to the case of 64 packet buffers, while the lower one refers to the case of 128 packet buffers. Markers correspond to simulations and report confidence intervals.

As predicted by the model, the different conditions of the US Browsing and Local Access scenarios have a minor impact on the average packet loss probability, that is predicted very accurately by the model over a large range of network loads. In particular, given the load at the bottleneck, the possible differences between the loss probability of the two scenarios are due to the burstiness and correlation of the segment arrival process at the bottleneck. Simulation results prove that these differences are negligible.

Fig. 6 reports the average completion time for 10 segments files (left plot) and 100 segments files

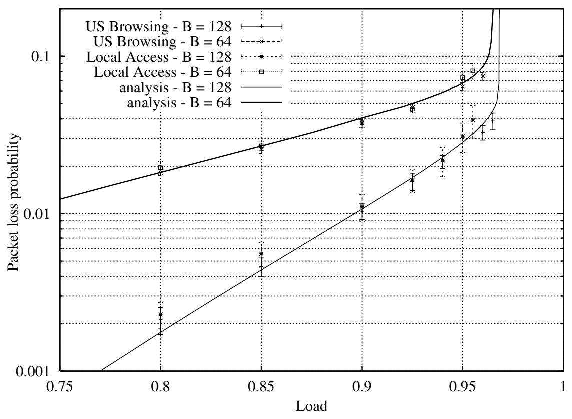

Fig. 5. TCP-Tahoe: packet loss probability as a function of the external load.

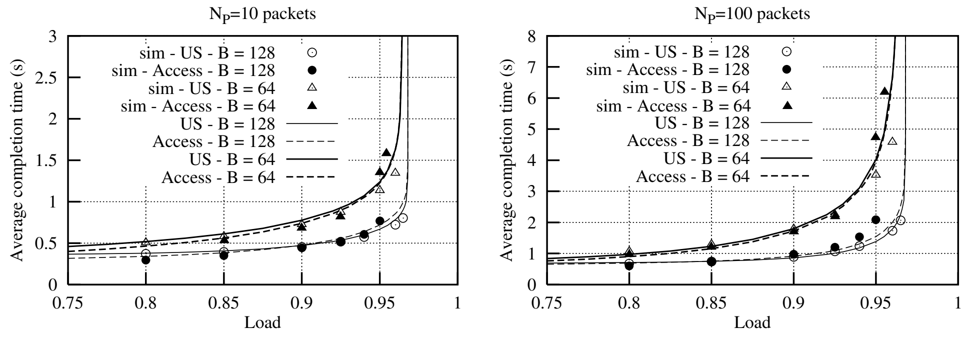

Fig. 6. TCP-Tahoe: average flow completion time as a function of the external load; 10 segment flows in the left plot, and 100 segment flows in the right plot.

(right plot). The file size has a major impact on results (notice the different y-scales of the plots), as expected. The TCP performance is still dominated by the buffer size (that drives the loss ratio), while the scenario has a minor impact. Results for the 20 segments flows are not reported to avoid cluttering the figure; they (obviously) lie between the 10 and 100 segment curves.

The presence of instability asymptotes around load 0.97 is evident in both figures. The vertical asymptotes correspond to points where the actual load of the network (considering non-necessary retransmissions, i.e., duplicated packets) approaches 1.

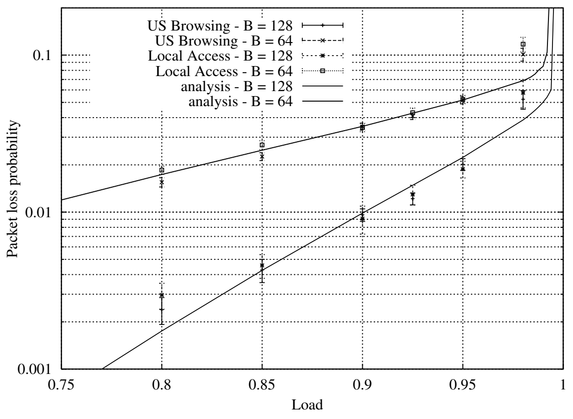

Fig. 7. TCP-NewReno: packet loss probability as a function of the external load.

8.3. Numerical results for TCP-NewReno

Fig. 7 reports the average packet loss probability in the same conditions as those of Fig. 5. The loss probability is similar to TCP-Tahoe, but the instability asymptotes are now closer to 1 (namely, a little beyond load 0.98).

For what concerns the session transfer time, we present the results in a different form, so as to give additional insight. Fig. 8 reports the results for buffer 128 only. The case with buffer 64 yields similar results, not reported to avoid repetitive patterns. The left plot refers to the US Browsing scenario, while the right one refers to the Local Access scenario. Both plots report three curves, one for each type of connection mix in the network. For light and medium loads, the session latency is dominated by the transmission delay, since loss events are rare; hence longer sessions have longer transfer times and the increase is roughly linear. The interesting fact is that the asymptote is the same for all connection lengths, meaning that the protocol is reasonably fair toward connections of different lengths.

The same is true also for TCP-Tahoe, but it is less evident from the presentation format of Fig. 6, which, on the other hand, better highlights the role of the buffer.

8.4. Mixed TCP-Tahoe and NewReno connections

We consider here only the US Browsing scenario with different mixes of Tahoe and NewReno

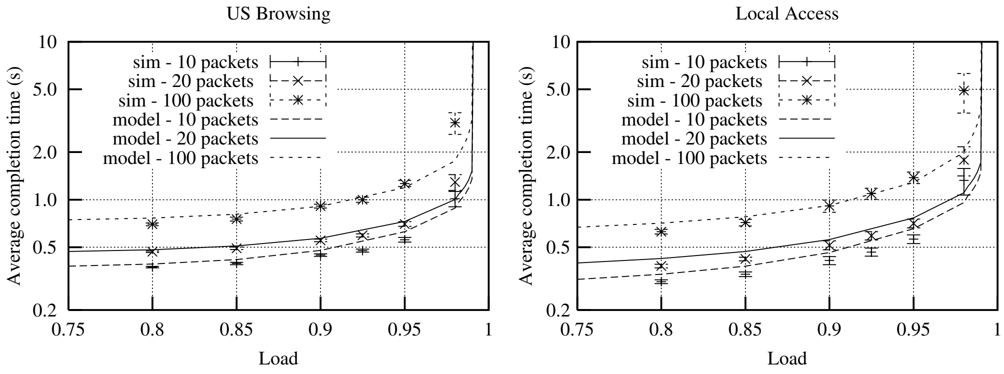

Fig. 8. TCP-NewReno: average flow completion time as a function of the external load for the US Browsing scenario (left plot) and for the Local Access scenario in the case of 128 packets buffer size.

connections competing for the bottleneck resources. To avoid excessive complexity, we consider the ideal case in which all files to be transmitted have identical size (we say these are *homogeneous* TCP connections), corresponding to a given number of segments (20, 50, 100, 200). We present numerical results for the cases of 50% Tahoe and 50% NewReno traffic (briefly 5T5R), 10% Tahoe and 90% NewReno (1T9R), 90% Tahoe 10% and NewReno (9T1R); the buffer size is always 128 packets.

The results are summarized in Fig. 9, that shows the packet loss probability and the completion times as a function of the external load. Notice that simulations confirm the assumption that Tahoe and NewReno connections mixed over the same channels perceive the same segment loss probability. The effect of the file size on the loss probability is remarkable: for equal load, the difference between 20-segments connections and 200-segments connections is between one and two orders of magnitude. This large difference is mainly due to the much higher burstiness of the traffic generated by longer TCP connections (recall that the maximum window size is set to 64 segments).

Comparing the results obtained with different percentages of Tahoe and NewReno traffic, we can note that the packet loss probability slightly increases with the Tahoe traffic share. This is due to the fact that Tahoe produces a larger number of unnecessary retransmissions with respect to NewReno. Indeed, with mostly NewReno traffic, the network utilization can be pushed closer to 1, as already noted with NewReno connections alone. Thus, by mixing the two TCP versions, Tahoe has a negative impact on NewReno, because it increases the overall traffic and the resulting packet loss probability.

The right column of Fig. 9 compares the performance of Tahoe and NewReno in terms of average file transfer time. Even if the difference is very small, for low loads NewReno performs worse, in both simulation experiments and our model. Instead, when used alone, NewReno produces a lower packet loss probability, and thus obtains better performance than Tahoe in terms of average completion time. The considered setup exhibits a strong correlation among losses, so that many packets can be lost in the same TCP window, thus resulting in a difficult operating condition for TCP in general, and for NewReno in particular. For high packet loss probabilities, both our model and simulations predict that Tahoe performs worse than NewReno, as generally accepted.

8.5. Numerical results for the two bottleneck network

Though the focus of this paper is on TCP, limiting the analysis and validation of the model to a single bottleneck is not entirely satisfactory, since no clues are given on the possible influence of correlations between multiple bottlenecks on the

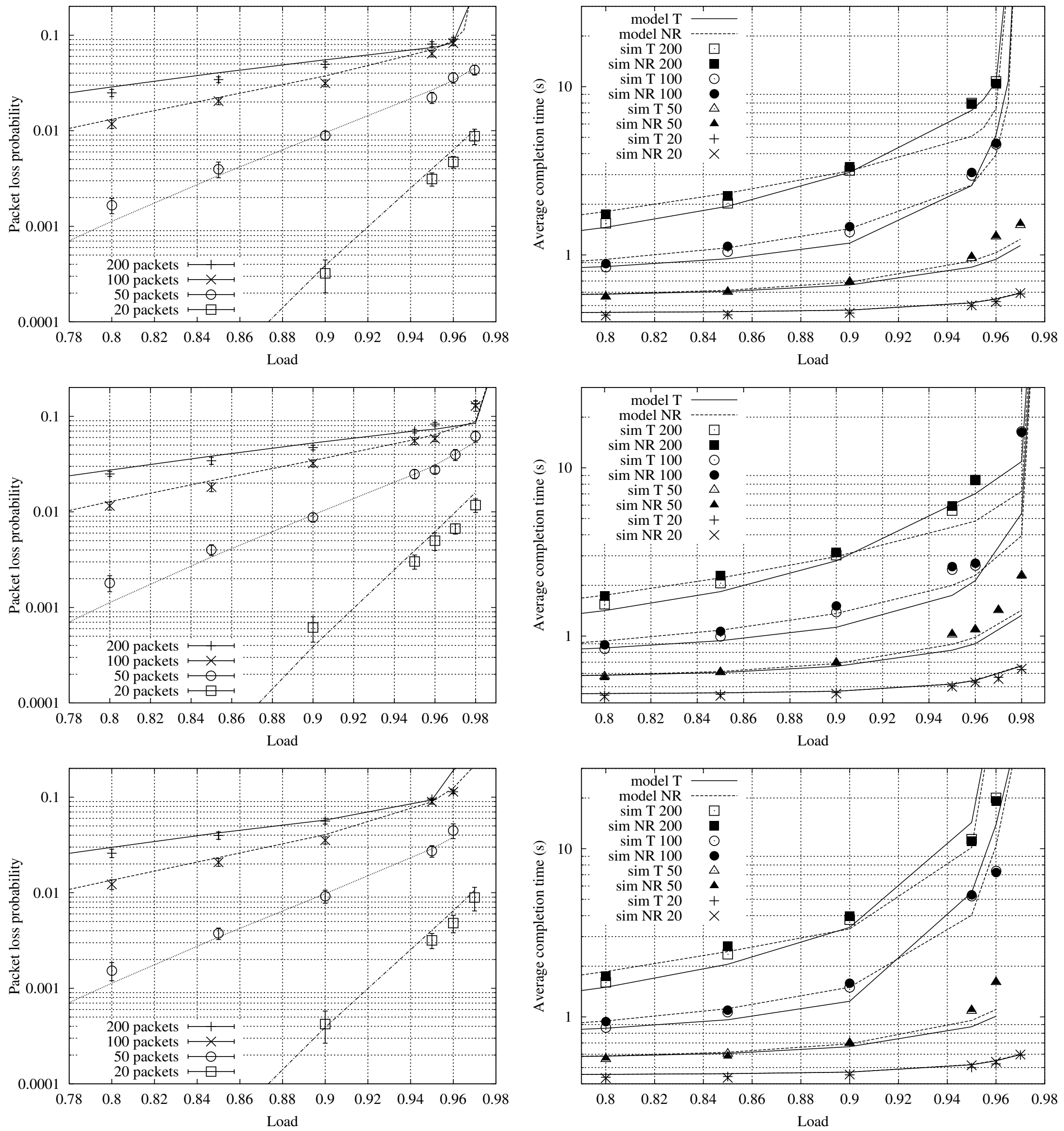

Fig. 9. Mixing Tahoe and NewReno connections in the US Browsing scenario: packet loss probability (left column) and completion times (right column); 5T5R (upper plots), 1T9R (middle plots) and 9T1R (lower plots); markers represent simulation results; lines represent model results.

model itself. Figs. 10 and 11 report results for the 2 Bottlenecks scenario, when the traffic pattern is such that both links MI-NY and POLI-TO are overloaded. Fig. 10 reports the overall loss rate experienced by flows crossing both links as a function of the load of link POLI-TO (Load 1) parametrized on different values of the load of link MI-NY (Load 2). The flow length distribution is the same as for previous plots. Fig. 11 presents a 3D plot of the durations of 20 packets connections. Also in this case the accuracy of the model is satisfactory, showing that the performance of TCP

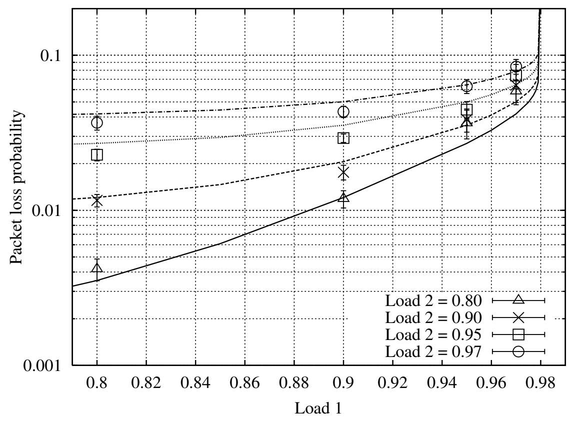

Fig. 10. Two-bottleneck scenario: packet loss probability.

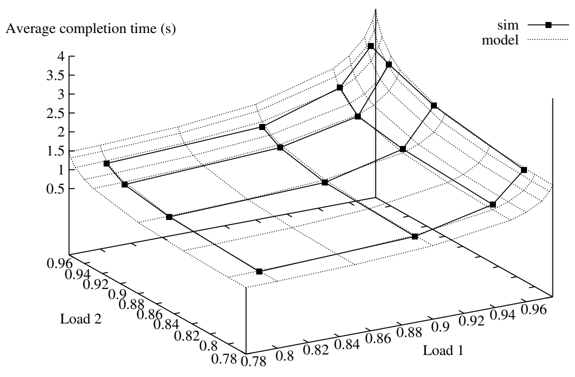

Fig. 11. Two-bottleneck scenario: completion times for connections with 20 segments.

depends basically on the overall loss rate. The same conclusion is intrinsic to models like [3,4], where the loss rate is an external parameter and does not take into account the network topology.

8.6. Numerical results for AQM networks and comparison with other approaches

The last scenario we explore considers IP networks comprising AQM routers, adopting gentle RED. We present results only for a single bottleneck setup, where the buffer size is 128 packets, the RED parameters are: low threshold 10 packets, high threshold 50 packets, maximum forced loss probability 10^{-1}, and low-pass filter parameter 10^{-5}.

This scenario allows the comparison of the results with the model proposed in [4] (CSA model), that was developed only for AQM routers that do not introduce correlation in the loss process. The CSA model requires as input the loss rate in order to compute the average flow duration, given the number of packets composing the flow. If the loss rate is known, this model is computationally lighter than ours, thus a comparison of the results is extremely interesting.

First of all, Fig. 12 reports the packet loss probability that results from the solution of our model compared with simulation estimates. The agreement is excellent, as expected, since, from a modeling point of view, the uncorrelated loss scenario generated by RED is easier to handle than the correlated one produced by droptail routers.

Fig. 13 presents flow completion times vs. the bottleneck nominal load. Markers represent simulation point estimates (with their confidence intervals). Dotted lines refer to our model, and continuous lines refer to the CSA model. In order to be as fair as possible in the model comparison, we fed the CSA model with the loss probability obtained by simulation, while our model jointly derives loss probability and flow durations. For low loads and short flows, the two models yield very similar results. When connections are long and the load is high, on the other hand, the simpler

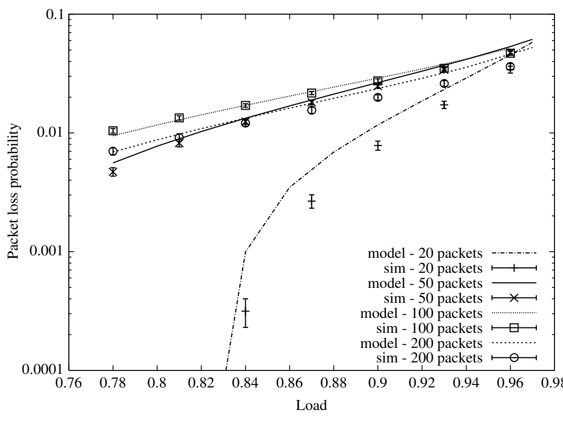

Fig. 12. US Browsing scenario with RED routers: packet loss probability; markers represent simulation results; lines represent model results.

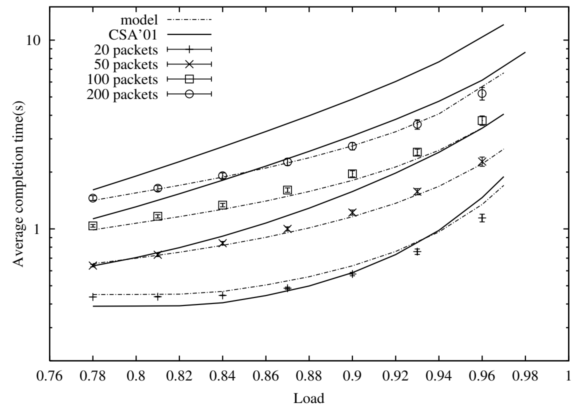

Fig. 13. US Browsing scenario with RED queue management: completion times and comparison with the CSA model; continuous lines refer to the CSA model, dotted lines to our model.

CSA model tends to overestimate the flows duration, while our model still provides results matching almost perfectly simulation estimates.

The global picture is thus fairly interesting, since one can imagine to use the simple CSA model to obtain approximate estimates of flow durations when expected loss probabilities are known, and use our more precise model in a more advanced stage, when accurate estimates of the loss probabilities and flow durations are required.

9. Conclusions

In this paper we have developed OMQN models that describe the behavior of short-lived TCP connections using either the Tahoe or the NewReno version of the protocol. The models accurately describe the initial slow start phase, as well as the subsequent congestion avoidance and slow start phases, allowing the evaluation of the file transfer time also in presence of short transfers, whose performance is dominated by the initial transient. The models can be combined to represent any mix of Tahoe and NewReno connections that exploit a common IP network with single or multiple bottlenecks, as well as droptail or AQM routers.

Starting from the primitive network parameters, the queuing network models are solved together with a simple model of the IP network, through an iterative procedure. The average loss rate experienced by the flows and the average time for the completion of the file transfer are estimated from the model solution.

The analytical performance predictions generated by the model for realistic networking scenarios have been validated against detailed simulation experiments. The approach was validated in the cases of one or two bottlenecks in the network, employing droptail as well as RED routers, and letting the parameters of the scenario vary over a wide range of values. Results proved that the proposed modeling approach is extremely accurate and flexible, besides providing useful insight into TCP dynamics and behavior.

The model was also compared with the one proposed in [4], which is computationally simpler, but requires the packet loss rate as input. The two models yield similar results when connections are short and the network load is light, while for long flows and high network loads, our model is more accurate.

Appendix A. Distribution of *ssthresh*

The threshold *ssthresh* that discriminates between the slow start and congestion avoidance transmission window growth modes introduces a memory in the TCP behavior. Indeed, the protocol evolution does not depend only on the present window size, but also on the value of the window at the previous loss event. This implies that a complete description of the protocol in steady-state conditions would require an overall number of queues around $W/2 \times M_q$. Rather than doing this, we resort to a stochastic approach based on the consideration that the steady-state distribution of the window size, which can be computed from the steady-state distribution of the number of customers in queues, allows the estimation of the distribution of *ssthresh*.

Let us introduce the set of queues visited only after a loss event

$$\Phi_{q_a} = \bigcup_i [EF_i \cup ET_i \cup LT_i \cup F_i \cup FF_i] \tag{A.1}$$

and let $\lambda_a(i)$ be the aggregate arrival rate at queues in Φ_{q_a} that have the same window size i. Then the probability $P_T(i) = P\{ssthresh = i\}$ is

$$P_T(2) = \frac{\sum_{k=1}^{5} \lambda_a(k)}{\sum_{k=1}^{W/2} \lambda_a(k)}, \tag{A.2}$$

$$P_T(i) = \frac{\lambda_a(2i) + \lambda_a(2i+1)}{\sum_{k=1}^{W/2} \lambda_a(k)}, \quad 3 \leqslant i \leqslant W/2. \tag{A.3}$$

We also define the cumulative distribution

$$P_C(i) = \sum_{k=2}^{i} P_T(k), \quad 2 \leqslant i \leqslant W/2. \tag{A.4}$$

Since arrival rates are obtained by solving the model, and the overall solution is obtained by iteration, we use the values of λ_{a_i} obtained at the previous iteration, checking that the $P_T(i)$ converge to a stable value.

Appendix B. OMQN service times and transition probabilities for the TCP-NewReno model

The model of TCP-NewReno, as described in Section 6, shares most of its characteristics with the model of TCP-Tahoe. We report here only service times and transition probabilities that are different from those of TCP-Tahoe, and that refer to the shaded queues in Fig. 3; however, the fact that a queue is shaded does not necessarily mean that *both* the service time and the transition probabilities are changed. Indeed, as far as service times are concerned, only queues modeling fast recovery are affected. In this case, the average number of packets lost in a window must be taken into account, since exactly one packet per RTT is retransmitted. Table 6 reports the modified service times.

Table 6
NewReno queues service times different from Tahoe

Queue	Service time
FF_i, LF_i	$\frac{\overline{\mathrm{RTT}}}{2} + \frac{4\overline{\mathrm{RTT}}}{i} + [1 + P_{L_a}(i-1)]\overline{\mathrm{RTT}}$
ET_i $1 \leqslant i \leqslant 3$	$T_0 - \overline{\mathrm{RTT}}$
ET_i $i \geqslant 4$	T_0
R_1	$\overline{\mathrm{RTT}}$
TK_2^1,	$T_2 - \overline{\mathrm{RTT}}$
TK_2^2,	$T_0 - \overline{\mathrm{RTT}}$

Table 7
NewReno transition probabilities different from Tahoe

q_i	q_j	k	$P(q_i, c; q_j, c-k)$
ET_i	R_1	0	1
FF_i	$L_{i/2}$	$\theta_t(c)$	$P_S^{\bar{n}}$
	R_1	$\theta_t(c)$	$1 - P_S^{\bar{n}}$
LF_i	$L_{i/2}$	$\theta_t(c)$	$P_S^{\bar{n}}$
	R_1	$\theta_t(c)$	$1 - P_S^{\bar{n}}$

Also changes in transition probabilities affect mainly the queues modeling fast recovery. The modified transition probabilities are reported in Table 7. In particular, all transitions corresponding to successful transmissions are not affected, as well as those relative to losses leading to timeouts and the *ingress* in fast recovery (with the exception of those starting from the queues modeling the steady-state slow start). Summarizing, the only transitions that have a different probability in the NewReno model are those exiting from queues FF_i and LF_i, plus those exiting from ET_i queues.

Transitions at the end of the fast recovery procedure are fairly complex, due to the large number of possible class changes, depending on the window dimension and the number of losses. Let θ_l be the class change corresponding to the successful termination of fast recovery, and θ_t be the class change when a timeout expires during fast recovery. We have

$$\theta_l = i + \left(\frac{i}{2} - 1\right), \tag{B.1}$$

$$\theta_t = P_{S_a}(i-1) + P_S\left(\frac{i}{2} - 1\right), \tag{B.2}$$

where i is the window size; the second term in both equations takes into account the new packets that the fast recovery algorithm allows to transmit. Notice that we implicitly assume that whenever a packet is lost during fast recovery (either a retransmitted packet or a new one), then NewReno goes through a timeout. This is a modeling simplification, since the real behavior implies a timeout when a retransmitted packet is lost, while if a new transmitted packet is lost, the behavior depends on the loss pattern, though a timeout is still highly probable. However, any further detail introduced in the model seems excessive. Given the window size i, the average number of losses that occur during fast retransmit is $\bar{n} = 1 + P_{S_a}(i-1) + P_S(i/2-1)$ (remember that entering fast recovery we have necessarily lost at least one packet), and the probability of retransmitting them all correctly is the transition probability toward the congestion avoidance phase.

References

[1] A. Feldmann, J. Rexford, R. Caceres, Efficient policies for carrying web traffic over flow-switched networks, IEEE/ACM Transactions on Networking 6 (1998) 673–685.

[2] V. Paxson, Empirically-derived analytic models of wide-area TCP connections, IEEE/ACM Transactions on Networking 2 (4) (1994) 316–336.

[3] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, Modeling TCP Reno performance: a simple model and its empirical validation, IEEE/ACM Transactions on Networking 8 (2) (2000) 133–145.

[4] N. Cardwell, S. Savage, T. Anderson, Modeling TCP Latency, Proceedings of IEEE Infocom 2000, Tel Aviv, Israel, March 2000.

[5] B. Sikdar, S. Kalyanaraman, K.S. Vastola, An integrated model for the latency and steady-state throughput of TCP connections, Performance Evaluation 46 (2–3) (2001) 139–154.

[6] V. Mishra, W.B. Gong, D. Towsley, Fluid-based analysis of a network of AQM routers supporting TCP flows with and application to RED, in: Proceedings of ACM SIGCOMM 2000, August 28–September 1, 2000, Stockholm, Sweden.

[7] C.V. Hollot, V. Mishra, W.B. Gong, D. Towsley, A control theoretic analysis of RED, in: Proceedings of IEEE Infocom 2001, Anchorage, AK, USA, April 22–26, 2001.

[8] A. Kumar, Comparative performance analysis of versions of TCP in a local network with a lossy link, IEEE/ACM Transactions on Networking 6 (4) (1998) 485–498.

[9] M. Garetto, R. Lo Cigno, M. Meo, M. Ajmone Marsan, A detailed and accurate closed queueing network model of many interacting TCP flows, in: Proceedings of IEEE Infocom 2001, Anchorage, AK, USA, April 22–26, 2001, pp. 1706–1715.

[10] T. Bu, D. Towsley, Fixed point approximation for TCP behavior in an AQM network, in: Proceedings of ACM SIGMETRICS 2001, Cambridge, MA, USA, June 16–20, 2001.

[11] M. Garetto, R. Lo Cigno, M. Meo, M. Ajmone Marsan, On the use of queueing network models to predict the performance of TCP connections, in: Proceedings of 2001 Tyrrhenian International Workshop on Digital Communications, Taormina (CT), Italy, September 17–20, 2001.

[12] E. Alessio, M. Garetto, R. Lo Cigno, M. Meo, M. Ajmone Marsan, Analytical estimation of the completion time of mixed NewReno and Tahoe TCP traffic over single and multiple bottleneck networks, in: Proceedings of IEEE Globecom 2001, San Antonio, TX, USA, November 25–29, 2001.

[13] ns-2, network simulator (ver.2), URL: http://www-mash.cs.berkeley.edu/ns.

[14] W.R. Stevens, TCP/IP Illustrated, vol. 1, Addison-Wesley, Reading, MA, 1994.

[15] W. Stevens, TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, RFC 2001, IETF, January 1997.

[16] P. Karn, C. Partridge, Improving round-trip time estimates in reliable transport protocols, Computer Communication Review 17 (5) (1987) 2–7.

Michele Garetto received the Dr. Ing. degree in telecommunications engineering from the Politecnico di Torino in May 2000, where he is currently working toward his Ph.D. in the Telecommunication Networks Group. From October 2001 to June 2002 he was with the CS Department at the University of Massachusetts, Amherst, as a visiting scholar. His research interests are in the field of performance evaluation of communication networks.

Renato Lo Cigno is Associate Professor at the Department of Computer Science and Telecommunications (DIT) of the University of Trento, Italy, where he is one of the founding members of the Computer Networks research group. He received a Dr. Ing. degree in Electronic Engineering from Politecnico di Torino in 1988. From 1999 to 2002 he was with the Electronics Department of Politecnico di Torino. From June 1998 to February 1999, he was at the CS Department at UCLA as Visiting Scholar under grant CNR 203.15.8. His current research interests are in performance evaluation of wired and wireless networks, modeling and simulation techniques and flow and congestion control.

Michela Meo received the Dr. Ing. degree in Electronic Engineering in 1993, and the Ph.D. degree in Electronic and Telecommunication Engineering in 1997, both from Politecnico di Torino, in Italy. Since then she has been working in the Telecommunication Networks Research Group of the Politecnico di Torino, where she is currently an Assistant Professor. Her research interests are in the field of performance evaluation of communication networks with a particular focus on wireless systems and end-to-end performance.

Marco Ajmone Marsan is a Full Professor at the Electronics Department of Politecnico di Torino, in Italy, and the Director of the Institute for Electronics, Information Engineering and Telecommunications of the National Research Council. He holds degrees in Electronic Engineering from Politecnico di Torino and University of California, Los Angeles. He has coauthored over 300 journal and conference papers in the areas of Communications and Computer Science, as well as the two books "Performance Models of Multiprocessor Systems" published by the MIT Press, and "Modelling with Generalized Stochastic Petri Nets" published by John Wiley. He received the best paper award at the Third International Conference on Distributed Computing Systems in Miami, FL, in 1982.

In 2002 he was awarded a "Honoris Causa" Degree in Telecommunication Networks from the Budapest University of Technology and Economics. His current interests are in the fields of performance evaluation of communication networks and their protocols.

He is a Fellow of IEEE, and a corresponding member of the Academy of Sciences of Torino. He participates in a number of editorial boards of international journals, including the IEEE/ACM Transactions on Networking.