On the Use of Queueing Network Models to Predict the Performance of TCP Connections*

M. Garetto, R. Lo Cigno, M. Meo, and M. Ajmone Marsan

Dipartimento di Elettronica – Politecnico di Torino
Corso Duca degli Abruzzi, 24 – I-10129 Torino, Italy
{garetto,locigno,michela,ajmone}@polito.it

Abstract. In this paper we describe an analytical approach to estimate the performance of greedy and short-lived TCP connections, assuming that only the primitive network parameters are known, and deriving from them round trip time, loss probability and throughput of TCP connections, as well as average completion times in the case of short-lived TCP flows. It exploits the queuing network paradigm to develop one or more 'TCP sub-models' and a 'network sub-model,' that are iteratively solved until convergence. Our modeling approach allows taking into consideration different TCP versions and multi-bottleneck networks, producing solutions at small computational cost. Numerical results for some simple single and multi-bottleneck network topologies are used to prove the accuracy of the analytical performance predictions, and to discuss the common practice of applying to short-lived TCP flows the performance predictions computed in the case of greedy TCP connections.

1 Introduction

Dimensioning, design and planning of IP networks are important problems, for which satisfactory solutions are not yet available. Many network operators and Internet Service Providers (ISPs) are dimensioning their networks by trial and error, and the more sophisticated dimensioning approaches are often still largely based on the packet network design algorithms devised in the '70s [1]. These rely on Poisson assumptions for the user-generated packet flows, thus completely neglecting the recent findings on traffic self-similarity, as well as the closed-loop congestion control algorithms of TCP, that carries about 90% of the Internet packets. A careful solution to the IP network dimensioning problem must be based on performance models capable of accurately estimating the efficiency of the applications run by end users. This translates in the need for models capable of accurately forecasting the throughput of TCP connections and the corresponding delay in the transfer of files. A large number of approaches have been recently proposed to estimate the performance of TCP connections interacting over a common underlying IP network (a brief overview of some of the recent proposals is presented in Section 2). They can be grouped in two classes:

* This work was supported by the Italian Ministry for University and Scientific Research through the PLANET-IP Project.

S. Palazzo (Ed.): IWDC 2001, LNCS 2170, pp. 536–555, 2001.

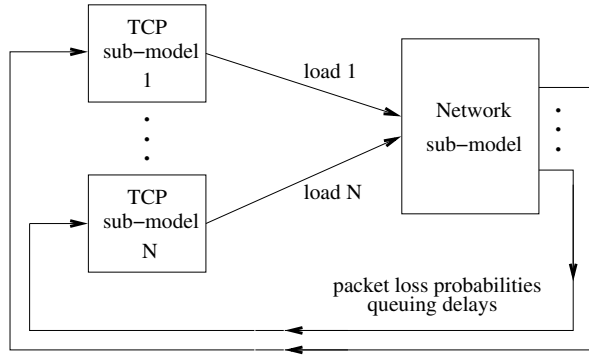

Fig. 1. Interaction between the TCP and the network sub-models.

1. models that assume that the round trip time and the loss characteristics of the IP network are known, and try to derive from them the throughput (and possibly the delay) of TCP connections;
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 possibly the delay of TCP connections, as well as the round trip time and the loss characteristics of the IP network.

Often, models in the second class incorporate a sub-model similar to the models of the first class to account for the dynamics of TCP connections, together with a sub-model that describes the characteristics of the IP network that carries the TCP segments. The two sub-models are jointly solved through an iterative fixed point algorithm (FPA). This is the approach that we adopt also in this paper, using the queuing network modeling paradigm to develop both the 'TCP sub-model' and the 'network sub-model'. Actually, with our approach, the TCP sub-model itself can be made of several components, each describing a group of 'homogeneous' TCP connection, where with such term we indicate TCP connections sharing common characteristics (same TCP version, comparable round trip time, similar loss probability). Each group is modeled with a network of $M/G/\infty$ queues, that describes in detail the dynamics of the considered TCP version, as explained later in Section 3. Instead, the network sub-model is made of a network of $M^{[D]}/M/1/B_i$ queues, where each queue represents an output interface of an IP router, with its buffer of capacity B_i packets. The routing of customers on this queuing network reflects the actual routing of packets within the IP network, as explained later in Section 4.

A graphical representation of the modeling approach is shown in Fig. 1. The TCP sub-model comprises several components referring to homogeneous groups of TCP connections. Each component receives as inputs (from the network sub-model) the appropriate estimate of the packet loss probability along the TCP connection routes, as well as the estimates of the queuing delays at routers. Each component produces estimates of the load generated by the TCP connections in the group. The network sub-model receives as inputs the estimates of the load generated by the different groups of TCP connections, and computes the loads on the network channels, and the packet loss probabilities and

average queuing delays. These are fed back to the TCP sub-models in an iterative procedure that is stopped only after convergence is reached. Thus, our modeling approach allows taking into consideration different TCP versions and multi-bottleneck networks. In addition, our modeling approach allows considering both long- and short-lived TCP connections, as explained in Section 3, and obtains solutions at very little cost, even in the case of thousands TCP connections that interact over the underlying IP network. The input parameters of the whole modeling approach are just the primitive characteristics of the protocol and the physical network (channel lengths and data rates, buffer sizes, packet lengths, file sizes, TCP versions, TCP connection establishment rates, maximum window sizes, etc.) All other parameters are derived from the protocol and network characteristics. The convergence of the FPA used in the solution of the overall composed model was studied and proved in several cases.

2 Related Work on TCP Models

The analysis of the behavior of TCP presented in [2] and [3] is 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 round trip time are needed as inputs to both models in order to allow the derivation of the TCP throughput and average window size. While the first paper assumes that losses are not correlated, the second paper also takes into account the correlation in packet losses. An extension to this latter paper that allows computing the latency of short file transfers is presented in [4]. Modeling techniques based on differential equations are used in [5] and [6]. A fluid flow model is used in [5] to compare the performances of TCP-NewReno and TCP-Vegas. The authors of [6] use differential stochastic equations to model the combined macroscopic behavior of TCP and the network itself. 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 [7], that presents Markov reward models of several TCP versions on lossy links. Also the work in [8,9] is based on a Markovian approach. The novelty in these works is the consideration of connections that switch on and off, following a two-state Markov model; however, the main performance figure is still the steady-state TCP connection throughput, since the model cannot provide estimates of completion times. Finally, [10,11,12,13] introduce the TCP modeling technique that is adopted in this paper. The models in both [10] and [11] consider greedy connections; the models in [12,13] consider instead finite-size file transfers. While the models in [10,11,12] always assumed that just one TCP version is used over the IP network, the model in [13] considers the simultaneous presence of TCP-Tahoe and TCP-NewReno connections.

3 TCP Sub-models

The behavior of concurrent TCP flows is described with a network of $M/G/\infty$ queues, where each queue describes a *state* of the TCP protocol, and each customer stands for an active TCP connection; thus, the number of customers at a queue represents the number of TCP connections that are in the corresponding state, and the times spent at

queues represent the time spent by TCP connections in states. Actually, even if we use the queuing network notation, that is most familiar to telecommunications engineers, the TCP sub-model can also be viewed as a stochastic finite state machine (FSM), which can be derived from the FSM description of the protocol behavior. When greedy TCP connections are studied, a fixed finite number N of active TCP connections must be considered; correspondingly, in the TCP sub-model, N customers move within the queuing network, which is *closed*. Instead, in the case of short-lived TCP connections, the model must consider a number of active TCP connections that dynamically varies in time: a TCP connection is first opened, then it is used to transfer a file composed of a finite number of packets, and finally, after the file has been successfully transferred, the connection is closed. This scenario is modeled by an *open* queuing network: a customer arriving at the queuing network represents an opening connection; a customer leaving the queuing network represents the file transfer completion. Moreover, in order to properly model the evolution of a short-lived TCP connection, the finite size of the file to be transferred has to be taken into account. Thus, besides accounting for the protocol state, the description of the behavior of a short-lived TCP connection has to be enriched with a notion of the amount of data to be transferred. This is modeled by introducing customer classes: the class of a customer represents the number of packets that still have to be transferred. A TCP connection which must be used to transfer a file using c packets is represented by a customer arriving at the queuing network in class c. Whenever the customer visits a queue that corresponds to the successful transfer of a packet, the class is decreased by one. When the customer class reaches zero (meaning that the last packet has been successfully transferred), the customer leaves the queuing network.

In order to illustrate the flexibility of the proposed modeling approach in both dealing with different TCP versions and modeling the different types of TCP connections, we describe in some detail the case of greedy TCP-Tahoe connections (Section 3.1) and the case of short-lived TCP-NewReno connections (Section 3.2).

3.1 Greedy TCP-Tahoe Connections

We describe the queuing network model of greedy TCP-Tahoe connections, assuming that fixed-size TCP segments are transmitted over the underlying IP network. Each queue in the queuing network is characterized by the congestion window size (*cwnd*) expressed in number of segments. In addition, the queue describes whether the transmitter is in slow start, congestion avoidance, waiting for fast retransmit or timeout. Fig. 2 shows the queuing network when the maximum window size is $W = 10$ segments. Queues 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. Queues below R_0 model backoff timeouts, retransmissions, and the Karn algorithm. The queuing network comprises 11 different types of queues, shortly described below.

Queues $E_i \quad (1 \leq i \leq W/2)$ model the exponential window growth during slow start; the index i indicates the congestion window size.

Queues $ET_i \quad (1 \leq i \leq W/2)$ model the TCP transmitter state after a loss occurred during slow start: the congestion window has not yet been reduced, but the transmitter is blocked because its window is full; the combination of window size and loss pattern

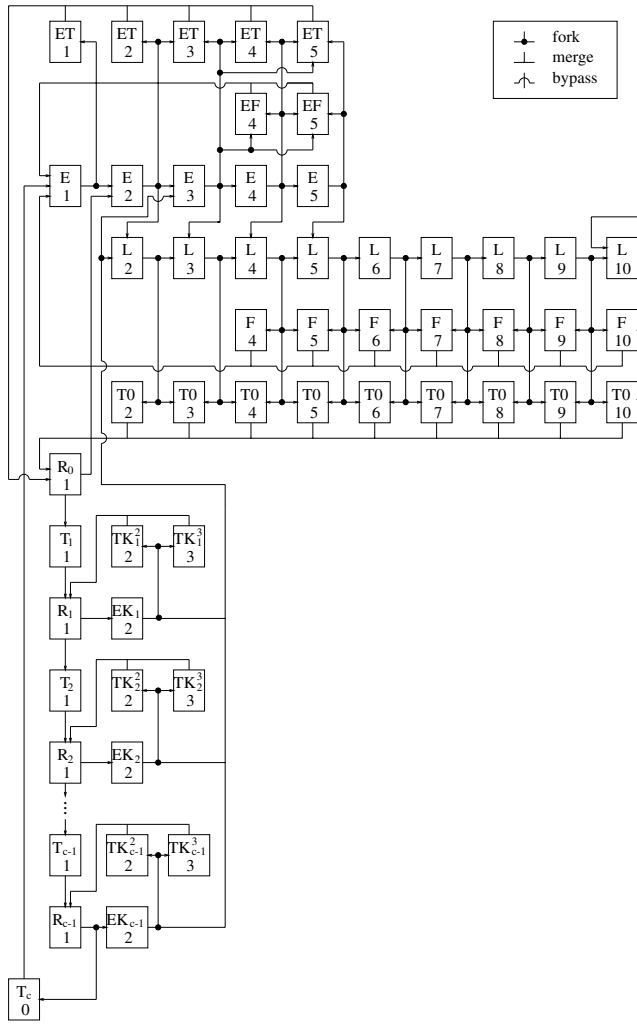

Fig. 2. Closed queueing network model of TCP-tahoe.

forbids a fast retransmit (i.e., less than 3 duplicated ACKs are received), so that the TCP source is waiting for a timeout to expire.

Queues $EF_i \quad (4 \leq i \leq W/2)$ 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.

Queues $L_i \quad (2 \leq i \leq W)$ model the linear growth during congestion avoidance (notice that queue L_1 does not exist).

Queues $F_i \quad (4 \leq i \leq W)$ model losses during congestion avoidance that trigger a fast retransmit.

Queues $T0_i \quad (2 \leq i \leq W)$ model the detection of losses by timeout during congestion avoidance.

Queues $T_i \quad (1 \leq i \leq C)$ model the time lapse before the expiration of the $(i+1)-$th timeout for the same segment, i.e., the backoff timeouts. C is the maximum number of consecutive retransmissions allowed before closing the connection. Queue T_C, actually models TCP connection that were closed for an excessive number of timeouts. In TCP-Tahoe this happens after 16 retransmissions, i.e., $C = 16$. Closed connections are supposed to re-open after a random time; which we chose equal to 180 s, however this value has a marginal impact on results.

Queues $R_i \quad (0 \leq i \leq C-1)$ model the retransmission of a packet when timeout expires.

Queues $EK_i \quad (1 \leq i \leq C-1)$ model the first stage of the slow start phase (i.e., the transmission of the first 2 non-retransmitted packets) after a backoff timeout. During this phase the Karn algorithm ([15] and [16] Ch. 21.3) has a deep impact on the protocol performance under heavy load.

Queues $TK_i^2 \quad (1 \leq i \leq C-1)$ model the wait for timeout expiration when losses occurred in queues EK_i leaving the congestion window at 2.

Queues $TK_i^3 \quad (1 \leq i \leq C-1)$ model the wait for timeout expiration when losses occurred in queues EK_i leaving the congestion window at 3.

For the specification of the queuing network model, it is necessary to define for each queue: the average service time, which models the average time spent by TCP connections in the state described by the queue; and the transition probabilities $P(Q_i, Q_j)$, which are the probabilities that TCP connections enter the state described by queue Q_j after leaving the state described by queue Q_i. Depending on the queue type, the service time is a function of either the average round-trip time or the timeout. For example, the average time spent in queue L_i (window size equal to i and congestion avoidance) is equal to the round-trip time; the average time spent in queue T_i ($(i+1)-$st timeout expiration) is equal to $2^i T_0$, where T_0 is the initial timeout and the term 2^i accounts for back-offs. Similarly, all other queue service times are set.

The probabilities $P(Q_i, Q_j)$ that customers completing their service at queue Q_i move to queue Q_j, can be computed from the dynamics of TCP. These dynamics depend on the working conditions of the network, i.e., they depend on the round-trip time as well as the packet loss probability. In order to cope with the correlation among losses of packets within the same congestion window, we introduce two different packet loss probabilities: the probability of loss of the first segment of the active sliding window, P_{L_f} (where f stands for 'first'), and the loss probability for any other segment in the same window P_{L_a} (where a stands for 'after' the first segment loss). Notice that, due to loss correlation, P_{L_a} is much larger than P_{L_f}. Besides depending on P_{L_f} and P_{L_a}, the probabilities of moving from queue to queue are also function of the distribution of the window growth threshold *ssthresh*. We denote by $P_T(i)$ the probability that *ssthresh* is equal to i. For example, the probability that a customer moves from queue E_1 to queue E_2 equals the probability that the segment transmitted while in queue E_1 is successfully delivered, i.e, $1 - P_{L_f}$. Consider now queue E_i, which represents the transmission of two packets following an ACK reception in slow start with window size equal to i. After visiting queue E_i, a customer moves to queue E_{i+1} if both transmissions are successful

and the threshold $ssthresh$ is not reached; the probability of this event is equal to

$$P(E_i, E_{i+1}) = \left(1 - P_{L_f}\right)^2 \frac{\sum_{k=i+1}^{W} P_T(k)}{\sum_{k=i}^{W} P_T(k)}.$$

The first term accounts for the probability that both segments are successfully transmitted; while the second term is the probability that the threshold is larger than i. Instead, the customer moves from E_i to L_i when the threshold is equal to i; the corresponding transition probability is

$$P(E_i, L_i) = \left(1 - P_{L_f}\right)^2 \frac{P_T(i)}{\sum_{k=i}^{W} P_T(k)}.$$

Similarly, all other transition probabilities can be derived. The distribution of $P_T(k)$ is determined based on the window size distribution (see [11]).

In order to compute the average load Λ offered to the network by TCP connections, we need to derive the load Λ_Q offered by connections in the state described by queue Q. If we denote with $\mathcal{P}_Q$ the number of packets offered to the underlying IP network by a connection in queue Q, the actual load offered by queue Q is $\Lambda_Q = \lambda_Q \mathcal{P}_Q$, where λ_Q is the customer arrival rate at queue Q. The terms λ_Q are computed from the queuing network solution, assuming a total of N customers (connections); the terms $\mathcal{P}_Q$ are specified by observing the TCP behavior. For example, each connection in queue E_i (slow-start growth and window size equal to i) generates two segments; each connection in queue L_i (congestion avoidance and window size equal to i) generates i segments. See [11] for a more detailed description of the service times and transition probabilities.

3.2 Short-Lived TCP-NewReno Connections

The main difference between the Tahoe and NewReno TCP versions lies in the presence of the fast recovery procedure [14], which permits to avoid the slow start phase, if losses are detected via duplicated ACKs, entering congestion avoidance after halving the congestion window. The number of queues and queue types needed for the description of TCP-NewReno is approximately the same as for TCP-Tahoe, being the states of the protocol and the values that the window can assume the same. The differences are mainly in the transition probabilities, which reflect the different loss recovery strategies of the two protocol versions. For instance, during the NewReno fast recovery procedure, the actual transmission window keeps growing, to allow the transmission of new packets while the lost ones are recovered. In addition, when considering short-lived instead of greedy connections, special care is needed in modeling the first slow-start growth, during which the threshold $ssth$ has not been set yet. The impact of this period can be remarkable, specially for very short flows. In the development of the queuing network model of short-lived NewReno TCP connections, the following classes of queues are added, with respect to those shown in Fig. 2.

Queues $FE_i\ (1 \leq i \leq W)$ model the exponential window growth during the first slow start phase after the connection is opened; the index i indicates the transmission

window size. During the first slow start phase the window can grow up to W, and the growth is limited only by losses, since *ssthresh* is not yet assigned a value different from W.

Queues $ET_i\,(W/2 \leq i \leq W)$ are similar to queues $ET_i\,(1 \leq i \leq W)$ but can be reached only from queues FE_i, i.e., during the first slow start on the opening of the connection.

Queues $FF_i\,(4 \leq i \leq W)$ model a situation similar to that of queues $EF_i \quad (4 \leq i \leq W/2)$ but are entered from queues FE_i in the first slow start phase.

Queues $FT_i\,(1 \leq i \leq C)$ model the time lapse before the expiration of the i-th or $(j+1)-$st timeout 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, which is one of the most interesting cases.

Queues $FR_i\,(0 \leq i \leq C),\ FEK_i\,(0 \leq i \leq C),\ FTK_i^2\,(0 \leq i \leq C),\ FTK_i^2\,(0 \leq i \leq C)$ are similar to queues R_j, EK_j, TK_j^2 and TK_j^3 and refer to packet retransmission and timeout expiration for losses during the first slow start phase.

Lack of space forbids a detailed description of this model, giving all the queues service times and transition probabilities. The interested reader can find all these informations in [12].

When considering short-lived connections, a customer leaving a queue may also change class (remember that the class indicates the number of packets still to be transmitted before closing the connection). We denote by $P(Q_i, c; Q_j, c-k)$ the probability that a customer leaving queue Q_i in class c moves to queue Q_j in class $c-k$, meaning that k packets were delivered in the protocol state represented by queue Q_i. The value of k depends on the protocol state, on the packet loss probability, and on the window size. For example, after visiting a queue which represents the transmission of two consecutive packets in exponential window growth (queues FE_i or E_i) k can be equal to 0, 1, 2, depending on the number of lost packets. Leaving queues L_i (linear growth mode, window size i), k is equal to i if all packets in the window are successfully transmitted; k is smaller than i if some of the packets are lost. Again, due to space limitation, we cannot provide more details about transition probabilities, and refer readers to [12] for further informations.

A TCP connection in a given queue Q and class c generates a number of packets equal to $\mathcal{P}_{Q,c}$, which is the minimum between c and the number of packets whose transmission is allowed by the protocol state. The load offered by connections in queue Q is given by, $\Lambda_Q = \sum_{c=1}^{N_P^{\max}} \lambda_{Q,c}\mathcal{P}_{Q,c}$ where $\lambda_{Q,c}$ is the arrival rate of customer in class c at queue Q, and $N_P^{\max}$ is the maximum considered file size. The average load offered to the network is then $\Lambda = \sum_{q \in S} \Lambda_Q$, where S is the set of all queues. The terms $\mathcal{P}_{Q,c}$ are computed similarly to the class decrement k discussed above. The difference consists in that $\mathcal{P}_{Q,c}$ includes all the packets that are submitted to the network, while k accounts for successfully transmitted packets only.

4 Network Sub-models

The network sub-model is an open network of queues, where each queue represents an output interface of an IP router, with its buffer. The routing of customers on this queuing network reflects the actual routing of packets within the IP network. In the description of the network sub-model, we assume that all router buffers exhibit a drop-tail behavior. However, active queue management (AQM) schemes, such as RED, can also be considered, and can actually lead to faster convergence of the FPA used in the solution of the complete model.

Different approaches, with variable complexity, were tested to model queues at router output interfaces. When dealing with greedy TCP connections, the best tradeoff between complexity and accuracy was obtained by using a simple $M/M/1/B_i$ queue to model each router output interface. This queuing model was observed to provide sufficiently accurate estimates of packet loss probability and average round-trip time with very limited cost; other approaches with significantly greater complexity provided only marginal improvements in the estimate accuracy. The Poisson packet arrival process at queue $M/M/1/B_i$ has rate Λ_i, which is computed from the TCP sub-model(s) and the TCP connection routes. The average customer service time equals the time to transmit a TCP segment, i.e., $8 \cdot MSS/C_i$ s, where MSS is the Maximum Segment Size in bytes, and C_i is the data rate on the i-th channel in b/s. The average segment loss probability P_{Li} can be derived from the $M/M/1/B_i$ queue solution. From P_{Li}, the loss probability of the first segment, P_{Li_f}, and the loss probability of any other segment of the same window, P_{Li_a}, can be derived. In [3] the authors assume that the intra-connection correlation among losses is 1, i.e., they assume that, after a loss, all the remaining packets in a congestion window are also lost, though the first lost segment is not necessarily the first segment in the window. In our case, considering that the TCP behavior is such that the first lost packet in a burst of losses is the one in the lowest position in a window (ACKs make the window slide until this is true), it seems more realistic to assume that the loss rate within a burst is obtained by scaling the initial loss rate: $P_{Li_a} = \alpha_i P_{Li_f} \qquad 1 \leq \alpha_i \leq \dfrac{1}{P_{Li_f}}$, with the constraint that the average loss rate is respected. Empirical observations show that the loss correlation becomes stronger as the window size grows; to catch this behavior we set $\alpha = \overline{w}$, subject to the above constraint, where $\overline{w}$ is the average congestion window size computed by the TCP sub-model for flows crossing the i-th channel. The TCP connection average round-trip times are computed by adding propagation delays and average queueing delays suffered by packets at router interfaces. When dealing with short-lived TCP flows, the $M/M/1/B_i$ queuing model fails in providing satisfactory results. This is due to the fact that in this case the traffic is much more bursty than for greedy TCP connections. In fact, the fraction of traffic due to connections in slow-start mode is much larger, and slow-start produces more bursty traffic than congestion avoidance. We chose to model the increased traffic burstiness by means of batch arrivals, hence using $M^{[D]}/M/1/B_i$ queues, where the batch size varies between 1 and W_i with distribution $[D]$. The variable size batches model the burstiness of the TCP transmissions within the TCP connection round-trip time. The Markovian assumption for the batch arrival process is mainly due to the Poisson assumption for the TCP connection generation process (when dealing with short-lived connections), as well as the fairly

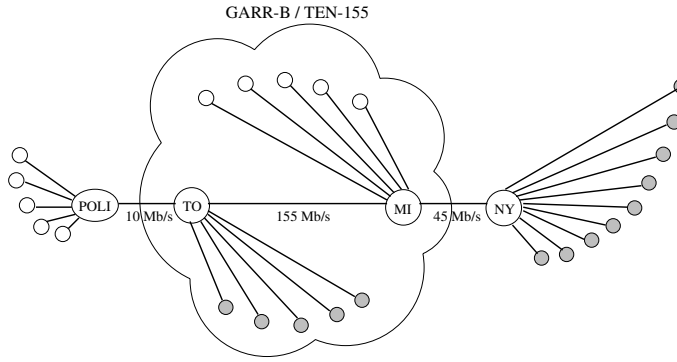

Fig. 3. Abstract representation of the Internet as seen from hosts connected to the LAN of Politecnico di Torino; white circles represent TCP clients, grey circles TCP servers

large number of TCP connections present in the network. The average packet loss ratio, P_{Li}, and the average time spent by packets in the router buffer, are obtained directly from the solution of the $M^{[D]}/M/1/B_i$ queue; the average round-trip times are then estimated by adding queueing delays and average two-way propagation delays of TCP connections. The key point of the modeling process in this case is the determination of the batch size distribution $[D]$. We compute the batch sizes starting from the number of segments N_{Ri} sent by a TCP transmitter during a round-trip time. Clearly, this is a gross approximation and a pessimistic assumption, since no two TCP segments are transmitted at the same time, and arrive at the same time at the router buffers. To balance this pessimistic approximation, the batch size is reduced by a factor $\mu < 1$. Actually, since the TCP transmission 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 $\mu^e = 2/3$, $\mu^l = 1/3$, that yielded satisfactory results in all tested scenarios.

5 Numerical Results

In this section we present some selected results, trying to highlight the flexibility of the modeling technique and the accuracy of the performance indices that can be evaluated using our methodology. Most results are validated against point estimates and confidence intervals obtained with *ns-2* simulations [17].

5.1 The Network Topology

The networking setup we chose to assess the results obtained with the queueing network models of TCP, closely resembles the actual path followed by Internet connections from our University LAN to web sites in Europe and the USA. A schematic view of the network topology is shown in Fig. 3; 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, grey 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 IP network for universities and research institutions, named GARR-B (a portion of the European TEN-155 academic network), 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 is assumed to be uniformly distributed between 100 and 6,800 km. From Milano, a 45 Mb/s undersea channel whose length is about 5,000 km reaches New York, and connects the GARR-B network to the north-American Internet backbone (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 from Milano is assumed to be uniformly distributed between 200 and 2,800 km. Finally, the distance of servers in the US from the router in NY is assumed to be uniformly distributed between 200 and 3,800 km. For simplicity, we assume one-way TCP connections with uncongested backward path, so that ACKs are never lost or delayed. We consider three types of TCP connections:

1. TCP connections from US servers to Politecnico di Torino clients; with our previous assumptions, the total length of these connections ranges from 5,351 to 8,960 km.
2. TCP connections from GARR-B/TEN155 servers to clients at Politecnico di Torino; connections lengths range between 151 and 6,860 km.
3. TCP connections from US web sites to users connected to the GARR-B/TEN-155 network through the MI-NY link; connections lengths range between 5400 and 11,600 km.

Given the link capacities, congestion occurs in either the 10 Mb/s POLI-TO channel (which is crossed by connections of types 1 and 2) or the 45 Mb/s MI-NY channel (which is crossed by connections of types 1 and 3), or both. The packet size is assumed constant, equal to 1,024 bytes; the maximum window size is 64 packets. We consider variable buffer sizes, and the default TCP tic value, equal to 500 ms. Connections that cross both links are those of group (1) and are assumed to be 25% of all connections traversing POLI-TO link (N1).

When the POLI-TO channel is congested, and the MI-NY channel is lightly loaded, the scenario is called LOCAL ACCESS, since the bottleneck is only in the local access link. When instead the MI-NY channel is congested, and the POLI-TO channel is lightly loaded, the scenario is called USA BROWSING, since the bottleneck is only in the transoceanic link. In both these cases, the topology practically reduces to a single bottleneck network. When both channels are congested, the topology is a two-bottleneck network.

5.2 Results for Greedy TCP Connections

In this section we discuss some of the results obtained for greedy connections with the closed queueing network model. Due to space limitations, we only consider connections

traversing both bottlenecks, i.e., TCP connections of type (1) in the networking scenario discussed above, and we just show numerical results for loss probability and average congestion window size. A discussion of the results obtained in the case of single bottleneck networks (LOCAL ACCESS or USA BROWSING scenarios) can be found in [11], where we show that our model also derives very accurate estimates for the distribution of the TCP congestion window and slow start threshold. This proves that the model actually captures most of the internal dynamics of the TCP protocol, not only its average behavior. In [11] we also show how the model can be used to compute the performance (e.g., the throughput) of one specific connection chosen from all those that interact over the IP network.

Fig. 4 reports results for the packet loss probability of type (1) TCP connections. The top plot presents analytical and simulation results versus N_1 and N_2 (the numbers of connections on the POLI-TO and MI-NY channels, respectively); the bottom plot

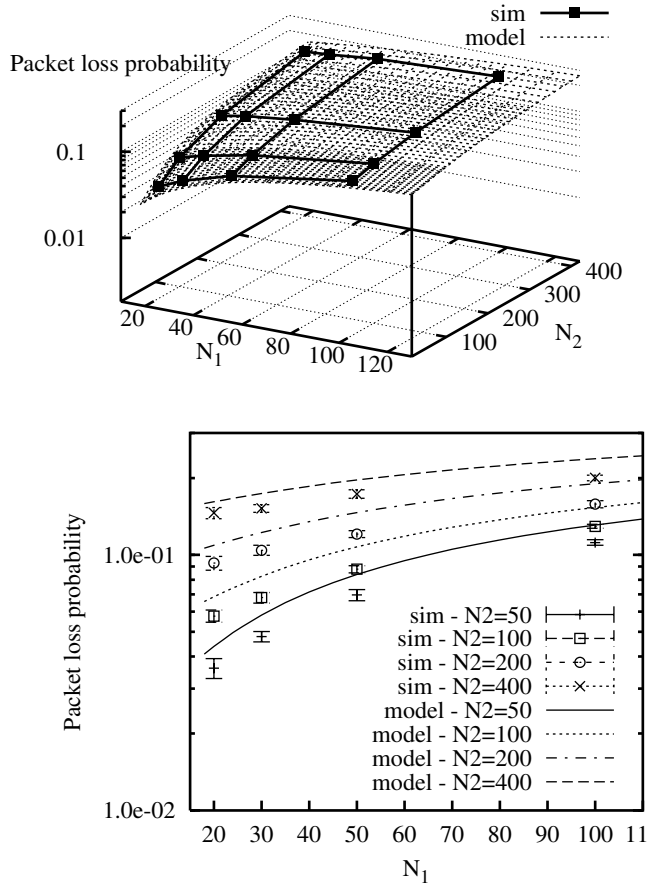

Fig. 4. Packet loss probability when connections cross both bottlenecks versus the number of connections N_1 and N_2 (top plot) and versus N_1 with constant N_2 (bottom plot)

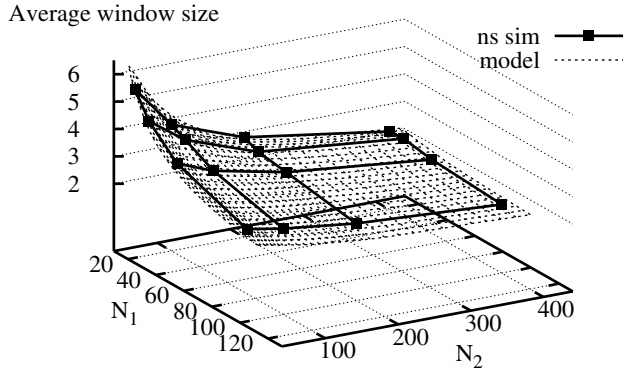

Fig. 5. Average window size when connections cross both bottlenecks versus the numbers of connections N_1 and N_2

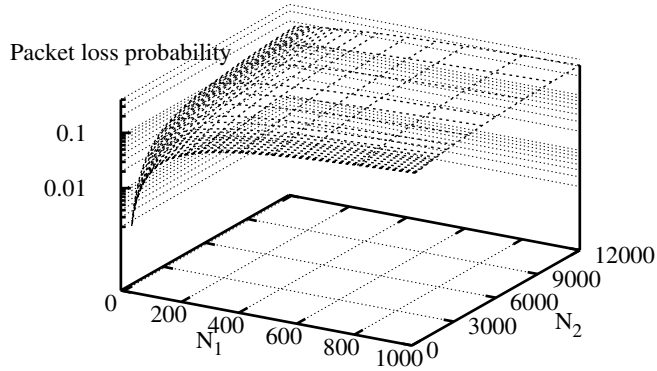

Fig. 6. Packet loss probability when connections cross both bottlenecks versus the numbers of connections N_1 and N_2; scenario with increased channel capacity

presents analytical and simulation results versus N_1 for different values of N_2. The average window size of the same connections is shown Fig. 5. For both performance indices, the accuracy of analytical performance predictions is excellent.

One of the most attractive features of a useful analytical model is its ability to provide results for scenarios where simulation fails, due to excessive CPU or memory requirements. We found that running simulations in *ns-2* with more than about 1,000 TCP connections becomes very expensive with standard computers; this is a significant

drawback, because channel speeds in IP networks are growing fast, and the number of concurrent TCP connections follows closely. We explored with the queuing network model a not-too-futuristic scenario where the POLI-TO link capacity is 100 Mb/s and the MI-NY link capacity is 1 Gb/s (other links are scaled accordingly); simulation results are obviously not available for this case. Fig. 6 reports the 3D plot of the packet loss probability in this scenario. The number of concurrent connections is up to 1,000 on the POLI-TO link, and up to 12,000 on the MI-NY link, that collects most of the traffic between the U.S. and Northern Italy. We can observe that the qualitative performance of the network does not change significantly by increasing the capacity and the number of connections. In particular, we found that for a given value of bandwidth per TCP connection, the loss ratio and the average TCP window size are almost constant. This observation gives strength to empirical models, such as those in [3], that derive the TCP throughput as a function of the loss probability. In addition, it partially relieves the worries concerning the possibility of exploiting TCP over very fast WANs, often expressed by researchers, but never proved due to the impossibility of simulating such environments or to load test beds with present day applications.

5.3 Results for Short-Lived TCP Connections

We now discuss results obtained with the open multiclass queuing network model for short-lived NewReno connections in the case of single bottleneck topologies. 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 a traffic scenario where the file size has the following distribution: 50% of all TCP connections comprise 10 segments, 40% comprise 20 segments and 10% comprise 100 segments. Of course, the chosen file size distribution is just an example, and is way too simple to acceptably describe a realistic situation. Note however that, given any file size distribution, the queuing network model can be solved with limited complexity.

Fig. 7 reports analytical and simulation results for the average packet loss probability in both the USA BROWSING and LOCAL ACCESS scenarios. Results are plotted versus the normalized external load, which is the load that would results in the case of no packet loss, so that retransmissions are not necessary. The upper curve refers to the case of 64 packet buffers, while the lower one refers to the case of 128 packet buffers. Since with our modeling approach the packets loss probability depends on neither the TCP connections round trip time, nor the link capacity (note that the external load is normalized), the analytical loss probability estimates are the same for both channels, and simulation results indeed confirm this prediction.

When the file transfer latency is considered, results are obviously no more independent from link capacities and round trip times. Fig. 8 reports results for buffer 128 only; the case with buffer 64 yields similar results, not reported for lack of space. The left plot refers to the USA BROWSING scenario, while the right one refers to the LOCAL ACCESS scenario. Both plots report three curves, one for each file size. For light and medium loads, the TCP session latency is dominated by the transmission delay, since loss events are rare; hence longer sessions have longer transfer times and the latency increase is roughly linear.

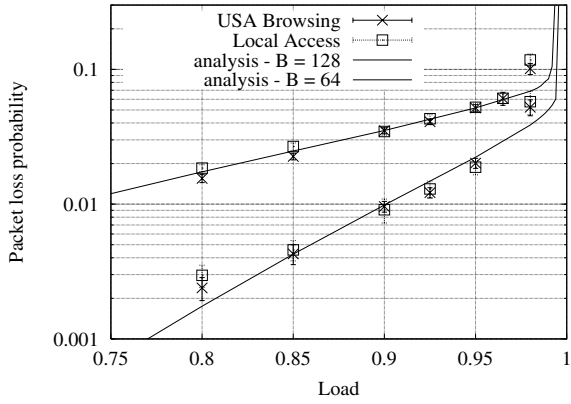

Fig. 7. Packet loss probability as a function of the external load for NewReno connections

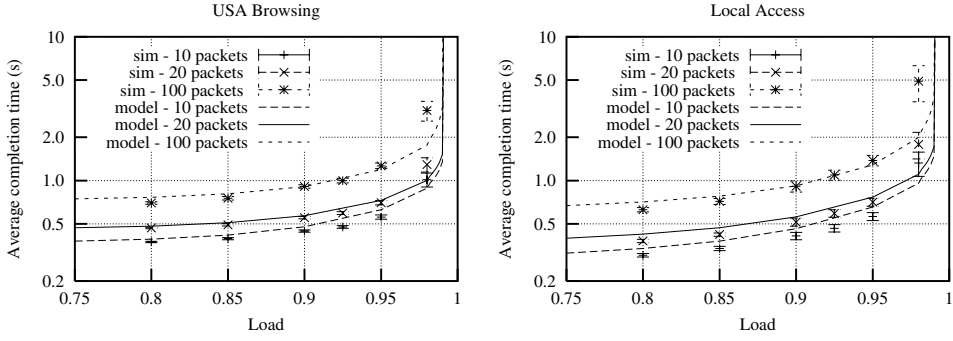

Fig. 8. Average session completion time as a function of the external load for the USA BROWSING scenario (left plot) and for the LOCAL ACCESS scenario in the case of 128 packets buffer

5.4 Comparison of the Results Obtained for Greedy and Short-Lived TCP Connections

So far, we have separately discussed the performance of greedy and short-lived TCP connections, but we have not compared the two. Traditionally, in the development of TCP models, short-lived connections were seen as "portions" of a greedy connection, disregarding the importance of transients. We discuss here the different behavior of short-lived and greedy TCP connections. We consider a single bottleneck network topology, that can be either the LOCAL ACCESS or the USA BROWSING scenario, and the TCP-Tahoe version. In order to compare greedy and short-lived TCP connections, it is important to identify a "common operating condition" (COC for short), which cannot be the link utilization, because greedy connections force it to be close to one, provided that TCP connections are not window-limited. The two COC we chose to consider are either the packet loss probability or the average number of active connections (note that fixing one of those does not imply that other system parameters, e.g. the link loads, are

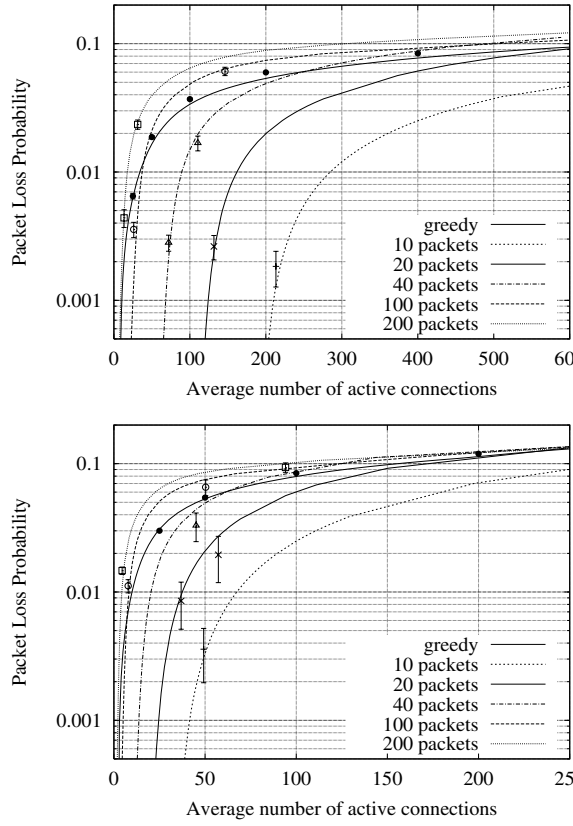

Fig. 9. Packet loss probability as a function of the average number of homogeneous connections in the in the USA BROWSING scenario (top plot) an in the LOCAL ACCESS scenario (bottom plot) with 128 packet buffer

equal). Observe that the COC is in general the result of the interaction between the TCP sources and the network, not a fixed parameter given to the model or to the simulator. The comparison is very interesting from the point of view of IP network dimensioning and planning, since results show that the performance figures obtained assuming the presence of greedy or finite connections can be quite different, thus possibly leading to inaccurate planning results if file sizes are not considered.

We discuss two types of comparisons: the packet loss probability curves as a function of the average number of connections, and the average throughput curves as a function of the packet loss probability. Fig. 9 shows the packet loss probability curves for different values of the file size: 10, 20, 40, 100, 200 packets, or infinite (greedy connections). While the results for finite connections indicate that the loss probability increases with the file size (this is due to an increase in the network load and in the traffic burstiness), the position of the curve for greedy TCP connections is far from intuitive, crossing all other curves as the number of connections increases. The differences in performance are

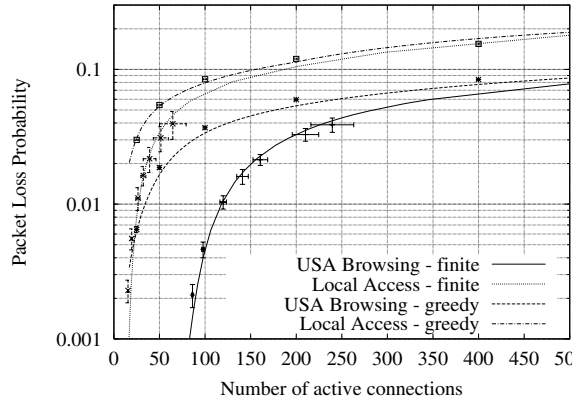

Fig. 10. Packet loss probability as a function of the average number of Tahoe connections (mixing different sizes) in both scenarios in the case of 128 packets buffer

remarkable, but most important is the observation that greedy connections do not allow a "worst case analysis." Transfers of a few hundred kbytes of data (100–200 packets) yield much worse performance than greedy connections, due to a higher burstiness in the traffic generated by TCP during its initial transient that typically lasts several tens of packets. Fig. 10 shows similar curves in the LOCAL ACCESS scenario with mixed length short lived connections, using the file size distribution that was introduced before (50% with length 10, 40% with length 20, and 10% with length 100). Given this file size distribution, the performance results are driven by very short connections, and thus greedy connections do indeed allow a worst case analysis.

Even more interesting is the comparison of the throughput achieved by TCP connections for a given average packet loss probability, as shown in Fig. 11. The difference between greedy and finite connections is impressive, since the convexity of the curves is different in the two cases, suggesting significant differences in the dynamic behavior. For low packet loss probabilities, the throughput of finite connections is limited by the window size, while greedy connections can grow their window large enough to achieve the maximum possible throughput. Less intuitive is the behavior for medium packet loss probabilities, where connections comprising 100 or 200 packets obtain a throughput higher than that of greedy connections. This behavior is rooted in the initial TCP connection transient: during the first slow start phase, the window of finite connections increases geometrically to large sizes, much bigger than those achieved by greedy connections. Larger windows, coupled with the transient dynamics, means burstier traffic and bursty, correlated losses. This results in many connections terminating without any loss, while the others experience only one or two loss events, maybe with several packets lost, but typically triggering a single timeout. For high packet loss probabilities, the throughput becomes similar for all kinds of connections, since the window size always is quite small.

The consequences of these behaviors are that result for greedy TCP connections should be used with great care: if the packet loss probability and the round trip time

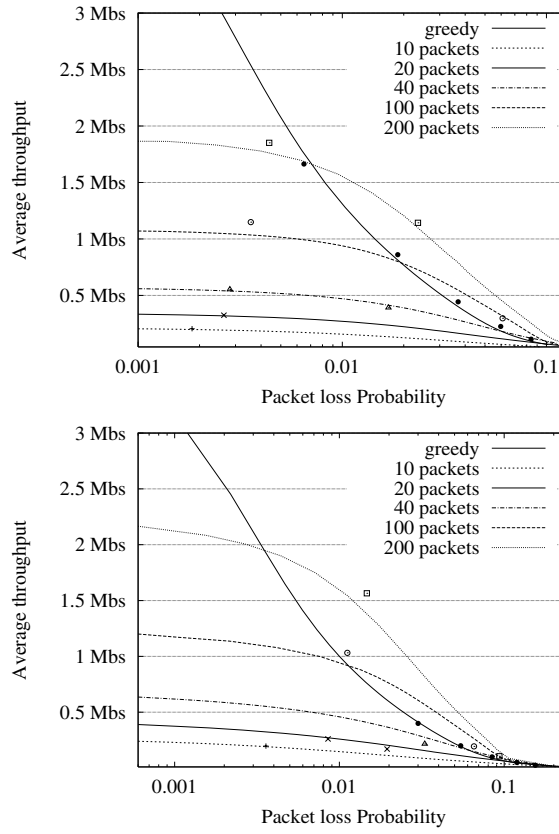

Fig. 11. Average throughput as a function of packet loss probability of homogeneous connections in the USA Browsing scenario (top plot) and in the Local Access scenario

of a TCP connection are known, and the existing closed-form formulas (like the square root formula [3] or its refinements) are used to estimate the connection throughput, the result may be rather different from the throughput of a short-lived TCP connection, and consequently the computed file transfer time may be inaccurate. Considering the fact that real TCP connections are necessarily finite, and often very short, the greedy source approximation should not be used, for instance, to predict the performance of web browsing TCP connections.

6 Conclusions

In this paper we described an analytical approach based on queuing networks to estimate the performance of greedy and short-lived TCP connections in terms of round trip time, loss probability and throughput of greedy and short-lived TCP connections, as well as average completion times of short-lived TCP flows. Models are built in a modular fashion,

developing one or more 'TCP sub-models' and a 'network sub-model,' that are iteratively solved. TCP sub-models are closed single-class queuing networks in the case of greedy TCP connection and open multi-class queuing networks in the case of short-lived TCP flows. The model solution requires as inputs only the primitive network parameters, and allows taking into consideration different TCP versions and multi-bottleneck networks, obtaining solutions at very little cost.

We presented numerical results for some simple single and multi-bottleneck network topologies, proving the accuracy of the analytical performance predictions, as compared to simulation point estimates and confidence intervals obtained with *ns-2*. In addition, we also compared numerical results for greedy and short-lived TCP connections, discussing the common practice of applying to short-lived TCP flows the performance predictions computed in the case of greedy TCP connections. The conclusion of this comparison is that greedy TCP connections can actually provide a worst case indication with respect to short-lived TCP flows, if the latter are rather short, but not in general. Indeed, when short-lived TCP flows comprise some hundreds packets, they can suffer worse performance than greedy TCP connections.

Further work in this area is still in progress, extending the modeling technique to other TCP versions, and trying to approximately account for losses of ACKs on the return path.

References

1. L. Kleinrock, "Queueing Systems, Vol. II: Computer Applications," John Wiley, 1976.
2. V. Paxson, "Empirically-Derived Analytic Models of Wide-Area TCP Connections," *IEEE/ACM Transactions on Networking*, 2(4):316–336, August 1994.
3. J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," *Proceedings of the ACM SIGCOMM'98 - ACM Computer Communication Review*, 28(4):303–314, September 1998.
4. N. Cardwell, S. Savage, T. Anderson, "Modeling TCP Latency," *Infocom 2000*, Tel Aviv, Israel, March 2000.
5. T. Bonald, "Comparison of TCP Reno and TCP Vegas: Efficiency and Fairness", *Performance Evaluation*, No. 36–37, pp. 307–332, Elsevier Science, Aug. 1999
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 Proc. SIGCOMM'2000*, Aug. 28–Sept. 1 2000, Stockholm, Sweden.
7. A. Kumar, "Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link," *IEEE/ACM Transactions on Networking*, 6(4):485–498, August 1998.
8. C. Casetti, M. Meo, "A New Approach to Model the Stationary Behavior of TCP Connections," *Infocom 2000*, Tel Aviv, Israel, March 2000.
9. C. Casetti, M. Meo, "Modeling the Stationary Behavior of TCP-Reno Connections," *International Workshop QoS-IP*, M. Ajmone Marsan, A. Bianco Ed., Lecture Notes in Computer Science 1989, Springer, 2001.
10. R. Lo Cigno, M. Gerla, "Modeling Window Based Congestion Control Protocols with Many Flows", *Performance Evaluation*, No. 36–37, pp. 289–306, Elsevier Science, Aug. 1999
11. M. Garetto, R. Lo Cigno, M. Meo, M. Ajmone Marsan, "A Detailed and Accurate Closed Queueing Network Model of Many Interacting TCP Flows," *in Proc. IEEE Infocom 2001*, April 22–26, Anchorage, Alaska, USA. An extended version of this paper, submitted for publication, is available `URL: http://www1.tlc.polito.it/locigno/papers/info01_extended.ps`

12. M. Garetto, R. Lo Cigno, M. Meo, E. Alessio, M. Ajmone Marsan, "Modeling Short-Lived TCP Connections with Open Multiclass Queuing Networks," subm. for publication, URL: http://www1.tlc.polito.it/locigno/papers/ TCP_OMQN.ps
13. 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," submitted to *IEEE Globecom 2001*
14. W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," RFC 2001, IETF, Jan. 1997
15. P. Karn, C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols," *Computer Communication Review*, Vol. 17, No. 5, pp. 2-7, Aug. 1987.
16. W. R. Stevens. *TCP/IP Illustrated, vol. 1.* Addison Wesley, Reading, MA, USA, 1994.
17. *ns-2*, network simulator (ver.2), Lawrence Berkeley Laboratory, URL: http://www-mash.cs.berkeley.edu/ns.