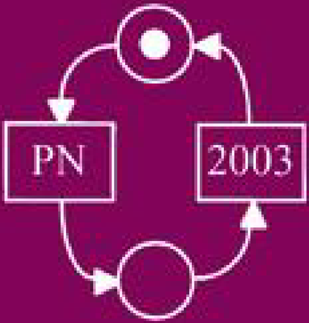
Wil van der Aalst
Eike Best (Eds.)

LNCS 2679

Applications and Theory of Petri Nets 2003

24th International Conference, ICATPN 2003
Eindhoven, The Netherlands, June 2003
Proceedings



Springer

On the Use of Petri Nets for the Computation of Completion Time Distribution for Short TCP Transfers

R. Gaeta, M. Gribaudo, D. Manini, and M. Sereno

Dipartimento di Informatica, Università di Torino,
Corso Svizzera, 185, 10149 Torino, Italia

Abstract. In this paper we describe how the completion time distribution for short TCP connections can be computed using Deterministic Stochastic Petri Net (DSPN) models of TCP protocol. A DSPN model of TCP is a representation of the finite state machine description of the TCP transmitter behavior, and provides an accurate description of the TCP dynamics. The DSPN requires as input only the packet loss probability, and the average round trip time for the TCP connections being considered. The proposed model has been validated by comparing it against simulation results in various network scenarios, thus proving that the model is accurate. Numerical results are presented to prove the flexibility and the potentialities of the proposed methodology.

Keywords: Performance Evaluation, Communication Network Planning, Transport Control Protocol, Deterministic Stochastic Petri Nets, Completion Time Distribution

1 Introduction

The performance in the Internet network is mainly dominated by physical network parameters, the IP routing strategies, the incoming traffic parameters, as well as all the aspects related with the used protocols.

In particular, the Quality of Service (QoS) perceived by end users in their access to Internet services is often driven by TCP, the connection oriented transport protocol, whose congestion control algorithm dictates the latency of information transfer.

TCP has been subject to numerous performance studies based on simulations, measurements, and analytical models. Modeling the TCP behavior using analytical paradigms is the key to obtain more general and parametric results and to achieve a better understanding of the TCP behavior under different operating conditions. At the same time, the development of accurate models of TCP is difficult, because of the intrinsic complexity of the protocol algorithms, and because of the complex interactions between TCP and the underlying IP network.

Although numerous papers proposed models and/or methods to obtain the *expected* throughput rates or transfer time, the variability of QoS requirements for different types of services and the need to provide *Service Level Agreements (SLA)* to end users, require the computation of more sophisticated performance metrics, such as completion time distributions time for TCP connections.

In this paper we show how transfer time distributions can be computed (and quantiles can be derived) using Deterministic Stochastic Petri Nets (DSPN). The DSPN model we developed requires two input parameters: the *packet loss probability*, and the *average round trip time*.
The packet loss probability is the probability that a TCP packet is dropped by one of the IP routers along the path from the TCP transmitter to the TCP receiver. Buffers in IP router have large but finite capacity therefore a packet is discarded (lost) either if an IP router has exhausted its buffering capacity for an incoming packet or if the buffer implements an Active Queue Management (AQM) algorithm where early packet drops are used to prevent network congestion (see [7] for an example of AQM scheme).
The round trip time (RTT) is a measure of the time it takes for a packet to travel from the source to the destination plus the time required for the acknowledgement packet to reach the source. The total time it takes for a packet to reach the destination (D) is the sum of three main components, i.e., $D = D_q + D_p + D_t$ where:

- D_q is the queuing delay; at each IP router it has to go through, a packet experiences a queuing delay as it waits its turn to be transmitted onto the proper link.
- D_p is the propagation delay; once a bit is transmitted onto the link, it needs to propagate to its destination. The time required to propagate depends on the link physical medium type and on the link length.
- D_t is the transmission delay; all bytes of a packet must be transmitted onto the link. The time required to push all bytes of a packet on to the link depends on the packet size and on the link capacity.

The two parameters required by the DSPN model, i.e., packet loss probability, and average round trip time, can be obtained in three different ways; interesting relations exist between the method used for deriving packet loss probability and average round trip time and the possible utilization of the proposed DSPN model:

- they can be measured over an actual network or an experimental setup; in this case, completion time distributions can also be measured, although this is significantly more complex, but the advantage of a model like the one described in this paper lies in the possibility of performing a *what-if* analysis, e.g., to assess the effectiveness of network upgrades.
- they can be derived from simulation experiments; in this case, completion time distribution can be also estimated from the same simulation experiments, but the CPU times needed to obtain reliable estimates for distributions from simulation experiments could be exceedingly high. The DSPN model of TCP thus allows simulation to be used only to obtain parameters (packet loss probability, and average round trip time) that are known to

be relatively easy to accurately estimate, and then to exploit the model to obtain completion time distributions.

– they can be estimated with an analytical model of the underlying IP network. In this case, the DSPN model of TCP can be used to compute completion time distributions.

The balance of this paper is outlined as follows: Section 2 provides a short overview of previous works that are closely related to our proposal. Section 3 presents a short description of TCP. In Section 4 we present our modeling assumptions and the description of the DSPN model of TCP. In Section 5 we present numerical results to validate the model and for providing some examples of the possible uses of the proposed methodology. Finally, Section 6 contains some concluding remarks, and outlines possible directions for future developments.

2 Related Works

The literature on analytical models of TCP is vast therefore it is difficult to provide here a comprehensive summary of previous contributions. In this section we only summarize some of the approaches that have been successfully used so far in this field and that are most closely related to our work.

One of the first approaches to the computation of the latency of short file transfers is presented in [3]. In this paper, an analytical model is developed to include connection establishment and slow start to characterize the data transfer latency as a function of transfer size, average round trip time and packet loss rate.

Works in [4] [1] too cope with finite TCP connections; their peculiarity is the analysis of connections that exhibit a on-off behavior, following a Markov model. In these works, the description of the protocol behavior is decoupled from the description of the network behavior and the interaction between the two submodels is handled by iterating the solution of the submodels until the complete model solution converges according a fixed point algorithm. In particular, [1] uses a GSPN based approach for developing the TCP behavior models.
The approach presented in [14] is based on the use of queueing network models for modeling the behavior of the window protocol. This approach has been extended in [8]. These works are based on the description of the protocol using a queueing network model that allows the estimation of the load offered to the IP network (represented using a different queueing network model). The merit of the queueing network approach lies in the product form solution that these models exhibit thus leading to more efficient computation of the interesting performance indexes.

On the other hand the approach presented in [12], allows the computation of the completion time distribution by using an analysis based on the possible paths followed by a TCP connections. The main problem of this approach is that the number of possible paths grows according to an exponential law with respect to the number of packets to be transferred over the connection. To overcome this problem an approximate method has been proposed. The approach presented in

[12] allows to account for independent losses, but other type of losses cannot be easily considered.

The use of Petri nets for performance evaluation of TCP and/or of issues related with this protocol is not new. In particular, in [16] a model of TCP, based on the Infinite-State Stochastic Petri Nets, is used to show the impact of lower-layer protocols (TCP and its windowing mechanisms) on the perceived performance at the application level (HTTP). In [5] a TCP model, based on Colored Petri Nets, is used to compare the behavior of two different TCP versions (Reno and Tahoe) when they face different packet loss situations, and to evaluate the impact of packet loss on the throughput. In [24] a TCP model, based on Extended Fuzzy-Timing Petri Nets, is used to evaluate the impact of QoS parameters on virtual reality systems.

To appreciate the contribution of the present paper is intersting to observe that the analysis presented in [5] and [24] are performed by simulation. On the other hand, the models presented in [1,5,16] allow to obtain first order measures such as throughput, average buffer occupation, and packet loss probability. It is important to point out that in network planning activities, the variability of QoS requirements for different types of services and the need to provide SLA to end users, require the computation of more sophisticated performance metrics, such as completion time distribution for TCP connections.
The contribution of the present paper is a DSNP model that allows the computation of completion time distribution for TCP connections. In general the computation of completion time distribution by means of simulation requires CPU times that could be exceedingly high. The complexity of the solution of the DSPN model is quite limited (see Table 5). Therefore the presented model can be effectively used in the context of planning and performance evaluation of IP networks.

3 The Transport Control Protocol: A Short Description

In this paper we mainly focus the attention on the TCP variant called *TCP Tahoe*. This version of TCP has been the first modification of the early implementation of TCP [19] and was proposed at time when Internet suffered from congestion. The TCP Tahoe implementation added a number of new algorithms and refinements. The new algorithms include: Slow Start, Congestion Avoidance, and Fast Retransmit [11]. The refinements include a modification to the round trip time estimator used to set retransmission timeout values [11]. The three algorithms provide the core algorithm of congestion control of TCP which is based on additive-increase/multiple-decrease principles. The congestion control of TCP gets the level of congestion from the arriving data-acknowledgement packets (ACK). When ACKs do not arrive at the sender, the TCP source waits for a timeout period: when new data is acknowledged by the TCP receiver, the congestion window (*cwd*) is increased (additive-increase); when congestion occur (congestion is indicated by packet loss detected by a retransmission timeout or the reception of three duplicate ACKs) one-half (multiple-decrease) of the current window size is saved in the slow start threshold (a congestion estimation value), and additionally the *cwd* is set to one packet. The technique to manage timeout is called *timer backoff* strategy. This technique computes an initial timeout value using the estimated round trip time. However, if the timer expires and causes a retransmission, TCP increases the timeout. In fact, each time it must retransmit a packet, TCP increases the timeout. To keep timeouts from becoming ridiculously long, most implementations limit increases to an upper bound. There are a variety of techniques to compute backoff. Most choose a multiplicative factor γ and set the new value equal to $= \gamma \cdot \text{timeout}$. Typically, γ is equal to 2. When the TCP transmitter does not have yet an estimation of the round trip time (in case of the transmission of the first packet) the initial value of the timeout is set to a given value.

The Slow Start mechanism is initiated both at the start of a TCP connection or after a retransmission timeout. It increases the *cwd* by one packet every ACK until *cwd* reaches the slow start threshold. During the slow start the sender is able to identify the bandwidth available by gradually increasing the number of data packets sent in the network. When *cwd* reaches the slow start threshold TCP uses the Congestion Avoidance mechanism which increases the *cwd* by a fraction of *cwd* ($1/cwd$) for each ACK received until *cwd* reaches the maximum congestion window size. After three duplicate ACKs are received the fast retransmit algorithm infers that a packet has been lost, retransmits the packet (without waiting for a timeout) and sets the slow start threshold to half the current window, and additionally, the current window is set to one packet. The Reno TCP version includes a further algorithm named Fast Recovery [2] which instead of reducing the *cwd* to one packet size after a three-duplicate ACK event, allows Reno to reduce the *cwd* to half its current value.

4 The Modeling Approach and Assumptions

In this section we describe the model assumptions and provide a detailed description of the DSPN model we propose.

4.1 Model Assumptions

The main goal of this work is the study of completion time distribution of finite TCP connections; to this aim we choose to represent in very detailed manner the TCP protocol and in an abstract way the remaining part of the communication network. In particular, our model captures aspects such as congestion window (*cwd*) evolution, slow start and congestion avoidance phases, TCP packet transmissions (and re-transmissions), management of packet losses due to time-out, and due to triple duplicate. As in most TCP modeling proposals, we do not model the connection setup and the connection closing phases. The network aspects are represented by two parameters: *round trip time* (RTT), and *packet loss probability* (p).

We model the TCP behavior in terms of *rounds*. A round starts with the back-to-back transmission of *cwd* packets, where *cwd* is the current size of the TCP congestion window. Once all packets falling within the congestion window have been sent in this back-to-back manner, no other packets are sent until the ACKs are received for these *cwd* packets or packet loss/losses are detected. These ACKs receptions mark the end of the current round and the beginning of the next round. In this model, the duration of a round is equal to the round trip time and is assumed to be independent of the window size. Note that we have also assumed that the time needed to send all the packets in a window is smaller than the round trip time. All these assumptions are quite common in the TCP modeling literature (see for instance [17]) and in the most cases are justified by observations and measurements of the TCP behavior.
In our model we assume that the duration of the round trip time is deterministic. This is an assumption that holds in communication networks where the fixed propagation delay is the dominant component of the round trip time. This is a common situation in wide area network.

Loss Packet Assumptions. The selection of a loss model is a key question in designing models of TCP. We assume that packets and ACKs are sent in groups over rounds, and that losses are independent from round to round. This assumption, which is made in most analytical studies (see for instance [3,17]), is partially justifiable with the understanding that TCP tends to send packets in bursts in a manner similar to how our models send packets in rounds. The independence of packet losses occurring in different rounds is especially likely to hold for connections with moderate to high round trip times since the time needed to send all the packets in a window is then much smaller than the round trip time [2]. On the other hand the independence of packet losses occurring within the same round is much stronger and in some cases is not realistic. For instance, this assumption is not realistic when a congested router uses the classical *drop-tail* policy.[1] In our model we focus on the following intra-round loss models:

- *Bernoulli*: Each packet is independently lost with a fixed probability p.
- *Drop-tail*: In each round, we consider the data packets sequentially. The first packet in the round is lost with probability p; for every other packets, if the previous packet was not lost, the packet is lost with probability p; if a packet is lost all subsequent packets in the round are lost.
- *Correlated*: In each round, we consider the data packets sequentially. The first packet in the round is lost with probability p; for every other packets, if the previous packet was not lost, the packet is lost with probability p; otherwise, it is lost with probability q.

The Bernoulli model is arguably the most basic model for packet loss. Owing to its simplicity, it lends to an easier analysis that the other loss models. The Bernoulli model may be appropriate for modeling congestion arising in routers

[1] A router that implements a drop-tail policy discards IP packets when its buffer is full.

that implement the *Random Early Detection (RED)* policy [7], since such routers respond to congestion by dropping IP packets uniformly at random. The drop-tail model is an idealization of the packet loss dynamics associated with a FIFO drop-tail router. It is assumed in this model that during congestion, routers drop packets in bursts [18], thus causing packets in the "tail" of a round to be lost. The Correlated model is somewhat less stringent. It characterizes the loss pattern as a Bernoulli distribution of loss episodes, each episode consisting of a group of consecutive packets, the length of which is approximated by a geometric distribution. Recent evidence for such correlated packet loss include [18,23]. We note that the Correlated model actually includes both the Bernoulli case (when $q = p$) and the drop-tail model ($q = 1$) as extreme cases.

4.2 DSPN: Definition and Notation

The modeling assumptions that we use, suggest to use the Stochastic Petri nets variant that allows to use deterministic, immediate, and exponential transitions (DSPNs) [9,13]. We also use other Petri net features such as marking dependent arcs and transition guard functions. For the model we propose, these features are not mandatory but they allow to obtain a more compact model.
In the following model description transition labels are written in italic style lower case *characters* for immediate transitions and upper case letter *CHARACTERS* for timed transitions. Places labels are written in sans serif style and upper case CHARACTERS.
The marking of place $\mathsf{P_i}$ is denoted as $\#\mathsf{P_i}$, the cardinality of marking dependent arcs is denoted as: $\langle expr \rangle$, where *expr* is the expression that represent the cardinality of the arc.

4.3 Model Description

In this section we illustrate the developed model for the computation of the completion time distribution of finite TCP connections. The specification of the model of Figure 1, requires the definition of guard functions, priorities, and weights, for immediate transitions, definition of delays for the timed transitions, and specification of the initial marking. Tables 1, 2, and 3 reports all these information. The subnet composed by places STEP1, STEP2, STEP3, STEP4, and transitions *s1*, *s2*, *s3*, *RTT* models the *round evolution*, i.e., since our model represents the TCP behavior in terms of round this subnet can be considered "the clock of the model". In this subnet there is one deterministic transition, *RTT*, whose firing delay is equal to the round trip time. Place CWD models the congestion window, i.e., the number of tokens in this place represents the current value of the congestion window. If the protocol is in Slow Start phase place SS_FLAG is marked, while if it is in Congestion Avoidance phase place CA_FLAG is marked. Marking of place SS_THRESH represents the threshold that triggers the change of the TCP state from Slow Start to Congestion Avoidance phase. The tokens in place PCKTOSEND represent the packets that have to be transmitted. The initial marking of places STEP1, SS_FLAG, and CWD is equal to 1,

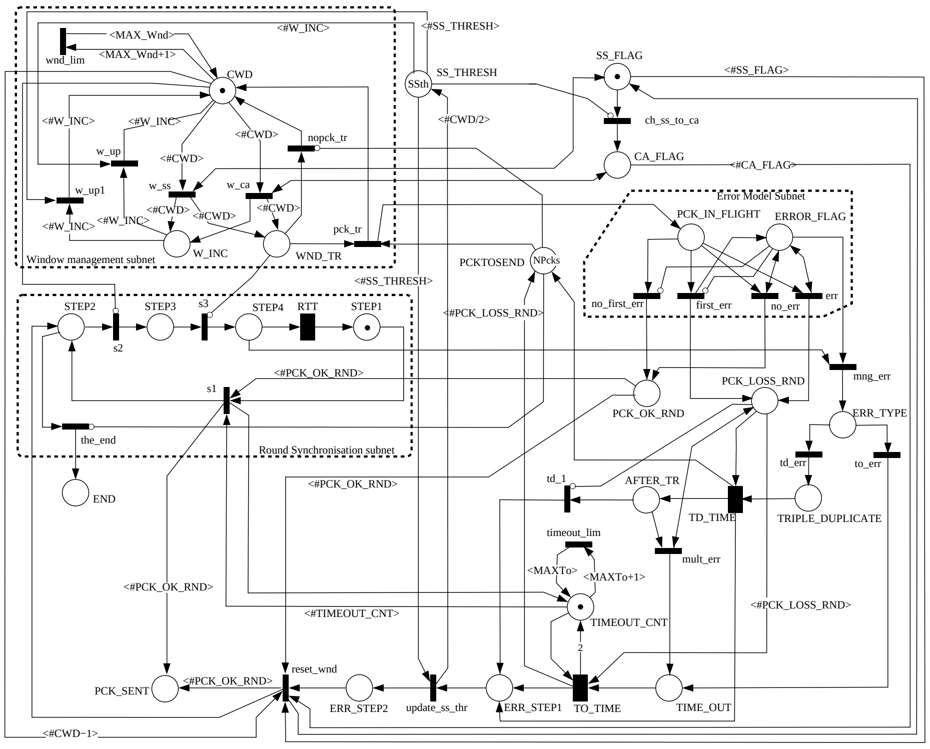


Fig. 1. DSPN model of TCP Tahoe

while sum of markings of places SS_THRESH and CWD represents the threshold that allows TCP to change from Slow Start to Congestion Avoidance phase. The subnet composed by immediate transitions *w_ss*, *pck_tr*, *nopck_tr*, *w_up*, *w_up1*, and places W_INC, and WND_TR models the congestion window evolution during the Slow Start phase. In the initial marking only transition *s1* is enabled and may fire (note that transition *w_ss* and transition *w_ca* are not enabled because their guard functions $f(w\_ss) = [(\#\mathsf{STEP2} = 1) \wedge (\#\mathsf{CWD} > 0)]$ and $f(w\_ca) = [(\#\mathsf{STEP2} = 1) \wedge (\#\mathsf{CWD} > 0)]$). When place STEP2 and place SS_FLAG are marked transition *w_ss* becomes enabled and may fire. The cardinality of arcs connecting place CWD to transition *w_ss*, and *w_ss* to places W_INC and WND_TR depends on the marking of place CWD. In this manner, the effect of the firing of *w_ss* "moves" the marking of CWD to places W_INC and WND_TR while place CWD is emptied. In the resulting marking, places W_INC, WND_TR, and STEP2 are marked, transitions *s2*, *pck_tr*, *nopck_tr*, *w_up*, and *w_up1*, have concession; due to guard functions (see Table 1) only transition *s2* is enabled and may fire. When place STEP3 is marked there are two transitions that can be enabled in mutual exclusion, *pck_tr*, or *nopck_tr*. These transitions allow to

Table 1. Definition of weights, priority levels, and guard functions for the immediate transitions (p is the packet loss probability)

Immediate transitions			
Label	Weight	Priority	Guard Function
s1	1	2	
s2	1	1	
s3	1	1	
w_ss	1	1	$\#\mathsf{STEP2} = 1 \wedge \#\mathsf{CWD} > 0$
w_ca	1	1	$\#\mathsf{STEP2} = 1 \wedge \#\mathsf{CWD} > 0$
pck_tr	1	1	$\#\mathsf{STEP3} = 1$
nopck_tr	1	1	$\#\mathsf{STEP3} = 1$
w_up	1	4	$\#\mathsf{STEP1} = 1 \wedge \#\mathsf{W\_INC} > 0 \wedge \#\mathsf{SS\_THRESH} \geq \#\mathsf{W\_INC}$
w_up1	1	4	$\#\mathsf{STEP1} = 1 \wedge \#\mathsf{W\_INC} > 0 \wedge \#\mathsf{SS\_THRESH} < \#\mathsf{W\_INC}$
ch_ss_to_ca	1	4	$\#\mathsf{STEP2} = 1$
wnd_lim	1	5	
no_first_err	$1 - p$	2	$\#\mathsf{STEP3} = 0$
first_err	p	2	$\#\mathsf{STEP3} = 0$
no_err	$1 - p$	2	$\#\mathsf{STEP3} = 0$
err	p	2	$\#\mathsf{STEP3} = 0$
mng_err	1	1	
to_err	3	1	
td_err	$\#\mathsf{CWD} - 3$	1	$\#\mathsf{CWD} > 3$
update_ss_thr	1	3	
reset_wnd	1	1	
the_end	1	5	
timeout_lim	1	6	

Table 2. Definition of delays for the deterministic transitions

Deterministic transitions	
Label	Delay
RTT	round trip time
TO_TIME	(see Eq. 1)
TD_TIME	round trip time

move tokens from place WND_TR to place CWD; furthermore, transition *pck_tr* models the transmission of TCP packets. If there is no packet to send, i.e., place PCKTOSEND empty, transition *nopck_tr* is enabled while *pck_tr* is not. In this case, the firings of *nopck_tr* only move the tokens from place WND_TR to place CWD. When place WND_TR is empty, only transition *s3* is enabled and may fire because of the guard functions. When place STEP4 is marked, the deterministic transition *RTT* is enabled and it fires after a round trip time elapses. When the token reaches place STEP1, four transitions have concession (*s1*, *w_up*, *w_up1*, and *w_ss* if place SS_FLAG is marked or *s1*, *w_up*, *w_up1*, and *w_ca* if place CA_FLAG is marked). Due to priorities and guard functions, only *w_up* or *w_up1* can be enabled. These two mutually exclusive transitions, model the doubling of the congestion window size when TCP is in slow start phase and the linearly increasing of the congestion window size when TCP is in congestion avoidance phase.

In particular, when $\#\mathsf{SS\_THRESH} \geq \#\mathsf{W\_INC}$ transition *w_up* is enabled; its firing empties place W_INC, puts #W_INC tokens in place CWD, and removes #W_INC tokens from place SS_THRESH. When $\#\mathsf{SS\_THRESH} < \#\mathsf{W\_INC}$ tran-

Table 3. Initial Marking Definition

Initial marking	
Label	Marking
STEP1	1
CWD	1
PCKTOSEND	Number of packets to send
SS_FLAG	1
SS_THRESH	Initial Slow Start Threshold value
TIMEOUT_CNT	1

sition *w_up1* is enabled, instead; its firing empties place W_INC, puts #W_INC tokens in place CWD, and empties place SS_THRESH.

When SS_THRESH becomes empty and place STEP2 is marked, transition *ch_ss_to_ca* becomes enabled and may fire. Its firing removes the token from place SS_FLAG and puts it in place CA_FLAG.
The immediate transition *w_ca* models the congestion window evolution during the Congestion Avoidance phase. In this case the behavior is similar to the Slow Start case. When transition *w_ca* fires it puts #CWD tokens in place WND_TR and only one token in place W_INC. In this manner the size of the congestion window size is increased by one each round.
The maximum number of tokens in place CWD is bounded by the transition *wnd_lim* and the by the input and output arcs that connect this transition to place CWD (the constant *MAX_WND* denotes the maximum congestion window size).

The immediate transition *pck_tr* models the transmission of TCP packets. This transition removes tokens from place PCKTOSEND and puts them in place PCK_IN_FLIGHT. The four immediate transitions *no_first_err*, *first_err*, *no_err*, and *err*, represent different error model behaviors. Due to priorities and guard functions, when the token is in place STEP4 transitions *no_first_err* and *first_err* become enabled. These immediate transitions model the occurrence of the first packet loss in the round. The weights of these transitions are such that the probability that *first_err* fires (resp. *no_first_err* fires) is equal to p (resp. $1-p$), where p is the loss probability of the first loss in a round. If a loss occurs these two transitions are no longer enabled in the same round, while *no_err* and *err* become enabled. These immediate transitions model the occurrence of the packet losses that may occur after the first loss. The firings of transitions *no_first_err* and *no_err* put tokens in place PCK_OK_RND. These tokens represent successful packet transmissions. On the other hand, transitions *first_err* and *err* put tokens in place PCK_LOSS_RND. These tokens represent lost packets that have to be re-transmitted.
When a packet loss occurs in a round, place ERROR_FLAG becomes marked and hence transition *mng_err* becomes enabled. Its firing puts a token in place ERR_TYPE thus enabling transitions *to_err* and *td_err*. These immediate transitions model the two different types of losses that may occur: losses detected by time-out expiration, and losses detected by triple duplicates [11]. There are situations where only the first case may occur; in particular when the size of

the congestion size is smaller than three a packet loss can only be detected by time-out expiration (this is accounted by means of a guard function on *td_err*). The weights of *to_err* and *td_err* are computed by using the same criterion that has been used in [17] (see this reference for details). In particular, the probability that a loss is due to time-out is $\max\{1, \frac{3}{\#\mathsf{CWD}}\}$.
We first describe the evolution of the model in case of packet loss detected by time-out expiration. The firing of transition *to_err* removes the token from place STEP4 (in this manner the round evolution is stopped) and puts a token in place TIME_OUT. In this marking, the deterministic transition *TO_TIME* is enabled. The firing time of this transition represents the TCP time-out. If the lost packet is the first packet of the connection, the TCP sender does not have an estimate of the round trip time and in this case the value of the time-out is a constant that may be dependent on the TCP implementation, we denote this time by T_{out}. In our case we set $T_{out} = 6$ sec. When the lost packet is not the first one of the connection, the TCP transmitter has an estimate of the round trip time and it can use the round trip time estimate to compute the time-out value, we denote this time by T_{out_RTT}. In both situations it may happen that more than one consecutive time-out expirations occur. In this case, the TCP transmitter exponentially increases the value of the time-out (*time-out backoff*). After a small number of consecutive time-out expirations the value of the time-out is kept constant (in the model, after six consecutive time-out expirations the marking of place TIMEOUT_CNT is not allowed to increase). The number of tokens in place TIMEOUT_CNT represents the number of consecutive time-out that have occurred. The maximum number of tokens in this place is bounded by transition *timeout_lim* and the by the input and output arcs that connect this transition to place TIMEOUT_CNT (the constant *MAXTo* denotes the maximum number of consecutive times that time-out can be increased). The marking of place TIMEOUT_CNT is set equal to one when a successful packet transmission occurs. We manage all possible cases by using a marking dependent firing time for transition *TO_TIME* defined in the following manner:

$$\text{firing time of } TO\_TIME = \begin{cases} 2^{(\#\mathsf{TIMEOUT\_CNT}-1)} T_{out} & \text{if } \#\mathsf{PCK\_SENT} = 0 \\ 2^{(\#\mathsf{TIMEOUT\_CNT}-1)} T_{out\_RTT} & \text{if } \#\mathsf{PCK\_SENT} > 0. \end{cases} \quad (1)$$

The first case accounts for the loss of the first packet of the connection, while the latter for the loss of a generic packet. When transition *TO_TIME* fires, tokens from place PCK_LOSS_RND are moved back to place PCKTOSEND. This represents the packets that have been lost and have to be re-transmitted. The firing of *TO_TIME* puts a token in place ERR_STEP1 and enables transition *update_ss_thr*. The firing of this immediate transition puts a token in place ERR_STEP2 and updates the Slow Start threshold to the value $\frac{\#\mathsf{CWD}}{2}$. This is obtained by a pair of marking dependent input and output arcs that connect place SS_THRESH to transition *update_ss_thr*. When place ERR_STEP2 is marked transition *reset_wnd* is enabled and may fire. The firing of this transition has several effects:

- it moves tokens from place PCK_OK_RND to place PCK_SENT (the packets that have been successfully transmitted);
- it moves the token from place CA_FLAG to place SS_FLAG (after a loss, TCP restarts from the Slow Start phase);
- it sets the current congestion window equal to one;
- it puts the token in place STEP2 (this resumes the round evolution).

The evolution of the model in case of tripe duplicate is quite similar to the time-out occurrence. In this case the firing of the immediate transition *td_err* enables the timed transition *TD_TIME* whose firing time is equal to the round trip time. The firing of *TD_TIME* puts a token in place AFTER_TD. When this place is marked and place PCK_LOSS_RND is empty the immediate transition *td_1* is enabled, on the other hand if place PCK_LOSS_RND is marked transition *mult_err* is enabled. These two mutually exclusive immediate transitions (*td_1* and *mult_err*) model the cases when in the round only a single triple duplicate loss occurs. On the other hand if in the round two or more losses occur (the first detected by triple duplicate) the second packet loss triggers a time-out (in this case transition *mult_err* is enabled and puts a token in place TIME_OUT).
If during the round no losses occur then immediate transition *s1* removes all tokens from place PCK_OK_RND and puts them in place PCK_SENT.
When place PCKTOSEND is empty and place STEP2 is marked transition *the_end* is enabled; its firing puts a token in place END. This represents the final absorbing state; the completion time distribution is then defined as the distribution to reach this absorbing state.

5 Results

In this section we first validate the proposed model by comparison against simulation results obtained in different network scenario; following the evaluation of the accuracy of the analytical approach we use the model we developed for analyzing cases of possible TCP scenarios. The completion time distribution is computed resorting to a transient analysis of the stochastic process underlying the DSPN model. In particular, we compute the probability that the TCP connection has completed its transfer of packets at time t by computing the probability that place END is marked at time t.

To perform the numerical experiments we use the tool TimeNet [25]. and an ad-hoc solution algorithm [10] that is faster that the one implemented in the tool TimeNet because it takes advantage from the peculiarities of the proposed model, i.e., there are only immediate and deterministic transitions (in each tangible marking only one deterministic transition is enabled).

5.1 Model Validation

In order to validate the DSPN model, we compare its results against simulation results obtained using the network simulator (ns-2) [15], which provides a detailed description of the dynamics of the Internet protocols.

We choose a simple network scenario where different type of traffic sources coexist. In particular, we consider a mix of long and short TCP connections, as well as, a mix of TCP and UDP traffic.
Indeed, numerous measurements show that the Internet traffic is now dominated by short flows involving transfer of small Web objects 10 – 20Kb in size [21]. While most Internet flows are short-lived, the majority of the packets belongs to long-lived flows, and this property holds across several levels of aggregation [21]. On the other hand, the proliferation of streaming contents, and hence of UDP-based traffic, forms a significant portion of Internet traffic [22].

– *Network Scenario.* Figure 2 illustrates a simple single-bottleneck topology that we use for our simulations. It is a simple network topology with a single bottleneck link whose bandwidth (C) is equal to $2Mbps$, and the two ways propagation delay (D_p) is equal to $120msec$. The bottleneck router has a buffer capacity equal to 100 packets and uses the RED queue management.

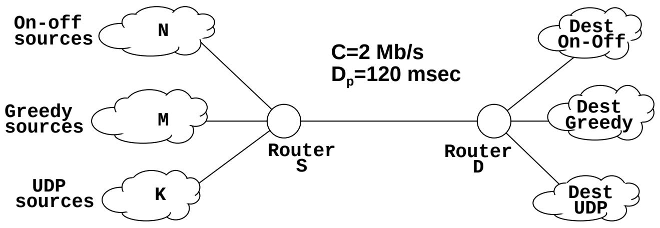


Fig. 2. Simulation topology

– *Traffic Characteristics.* We consider three types of traffic sources in our simulations:
  1. A set of N homogeneous *On-off* TCP traffic sources characterized by:
     - and alternating behavior where the silence periods are exponentially distributed with average equal to 1 sec;
     - a packet emission distribution $g = \{g_i\}$, where g_i is the probability that the number of packets that the connection has to transfer is equal to i; in our experiment we have that $g_1 = 0.3$, and and $g_i = 0.1$ for $i = 3, 5, 8, 12, 35, 120$.
  2. A set of M long-lived TCP traffic sources (greedy sources) that belong to FTP sessions with an infinite amount of data to transmit. In our simulations $M = N/10$.
  3. A set of K UDP traffic sources. In all the experiments the portion of UDP traffic is 5% of the available bandwidth. In our simulations $K = 2$.

  The size of TCP data packets is set to 500 bytes. The size of the TCP acknowledgments is set to 40 bytes. The maximum congestion window size is set to 21 packets. The UDP sources are Constant-Bit-Rate with rate equal to 64000 bps.

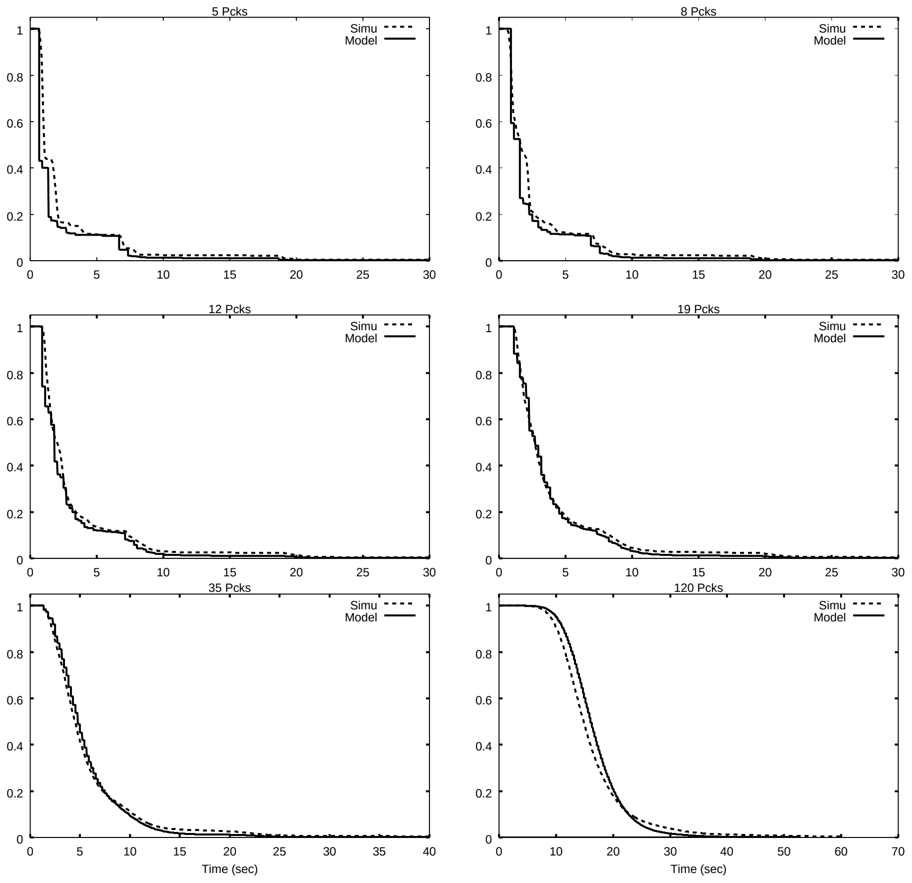


Fig. 3. Analysis vs Simulation, scenario with $N = 90$. 1−CDF of the completion time for connection with: 5 packets to transmit (upper row left column); 8 packets to transmit (upper row right column); 12 packets to transmit (middle row left column); 19 packets to transmit (middle row right column); 35 packets to transmit (lower row left column); 120 packets to transmit (lower row right column)

– *Simulation Methodology.* In our ns simulation the duration of the discared transient period is 300 sec, while the duration of each batch is 120 sec. Every simulations runs a maximum of 3000 batches, corresponding to a maximum simulated time of 360000 sec. Despite this upper bound limit we adopt a criterion to stop the simulations runs which refers to the completion time distribution measure. To estimate the completion time distribution we restrict the analysis to the interval $[0 - 1000 * D_p]$; i.e., 1000 times the propagation delay. We split this time interval into histograms whose width is 10 times the physical delay. We stop the simulation when 70% of the histograms reach the 10% accuracy. We estimate the completion time distribution as well as other measures such as 90%, 95%, and 98% quantile, packet loss probability, and average round trip time. We measure all the metrics with a confidence level of 97.5%.

Figure 3 shows the complement of the cumulative distribution function (CDF), i.e., 1−CDF, of the connection completion time for 5, 8, 12, 19, 35, and 120 packets and $N = 90$. Although the plots of the CDF of the completion time give an intuition of the accuracy of the proposed model, they only allow to conduct a qualitative analysis. To give a quantification of possible differences between model and simulative results we resort to the computation of some parameters of the completion time distributions; in particular, we present results referring to the 90%, 95%, and 98% quantiles of completion time distribution. Table 4 shows the 90%, 95%, and 98% quantiles, the loss probability, and RTT for $N = 80, 90, 100$. Results obtained for different values of N in the range $30, \ldots, 150$ show similar level of accuracy: the relative errors for the 90%, 95%, and 98% quantiles are in the range $2 \div 20\%$.

The CPU time required for any simulation run was in the order of magnitude of several hours (at least ten), while the model solution required a few minutes of CPU time except for the case of 120 packets whose solution required one hour of CPU time. Please note that CPU times are drastically reduced when the solution technique proposed in [10] is used.

5.2 Model Exploitation

In general the computation of completion time distribution by means of simulation requires CPU times that could be exceedingly high. On the other hand, the complexity of the solution of the DSPN model is quite limited (as it can be derived by the state space size presented in Table 5). Therefore the DSPN model can be effectively used in the context of planning and performance evaluation of IP networks. Figure 4 shows some results obtained with a loss probability

Table 4. Comparison of simulation vs model results with $N = 80, 90$

$N = 80$, $RTT = 0.223251$, $p = 0.093475$						
n. Pcks	90%		95%		98%	
	mod	sim	mod	sim	mod	sim
5	3.35	4.32	6.67	7.08	8.68	8.16
8	4.46	5.52	7.11	7.68	9.79	10.08
12	5.58	6.48	7.78	8.28	11.36	10.81
19	7.14	7.81	8.45	9.12	12.92	12.72
35	8.90	9.36	10.69	11.16	16.07	18.12
120	20.09	19.81	22.30	23.16	29.02	29.88

$N = 90$, $RTT = 0.225975$, $p = 0.106652$						
n. Pcks	90%		95%		98%	
	mod	sim	mod	sim	mod	sim
5	6.68	6.96	6.68	7.81	18.68	18.96
8	6.90	7.21	7.58	8.16	18.90	19.08
12	7.13	7.44	8.03	8.76	19.13	19.44
19	7.68	8.41	9.16	9.84	19.58	20.04
35	9.84	10.44	11.65	12.61	21.39	21.84
120	22.72	23.04	25.21	28.08	32.99	34.81

Table 5. State Space Size of the DSPN model with different number of packet to transfer

Num. of Packets	State Space Size
8	158
10	229
30	1799
50	5052
80	14375
100	23372
120	34479

equal to 0.05 and different values for the round trip time (from 0.45 to 0.60 sec). Figure 5 shows similar results obtained with a round trip time equal to 0.4 sec and different values for the packet loss probability (from 0.01 to 0.15).

Results similar to those presented in Figures 4 and 5 are particular meaningful in the context of network planning. For instance, assume we are interested to plan a network which guarantees with confidence equal to 95% to all TCP connections shorter than 50 packets (in this case, since we assume that the packet size is equal to 500 Bytes, for transfers of files smaller that 25 KBytes) a completion time shorter than 15 sec. From the plots depicted in Figure 5 (right column) we can see that to achieve this target we need to plan a network which guarantees packet loss probability smaller that 0.12 (12%). If instead we are interested to guarantee a confidence level equal to 98%, for the same type of TCP connections, we need to ensure loss probability smaller that 0.08 (8%).

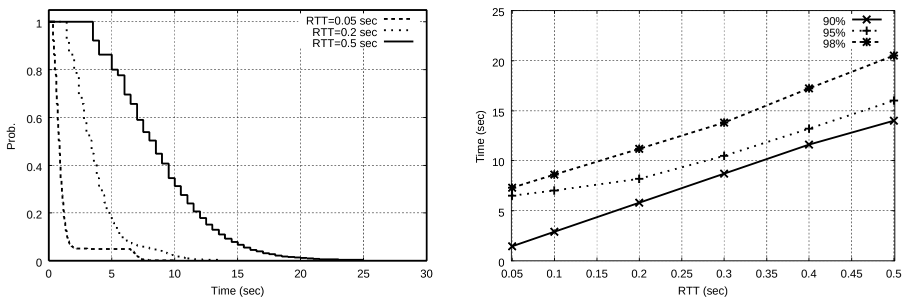


Fig. 4. Analytical results obtained for TCP connections with 50 packets, with fixed loss probability (0.05) and different values for the round trip time (from 0.05 to 0.50 sec): completion time distribution (left column) and quantiles (right column)

The DSPN model can also be used for investigating loss model effects. This could be meaningful to evaluate the impact of different IP router management policies and their effects on the correlation of loss occurrences. Figure 6 shows the completion time distributions (left column) and the 90%, 95%, and 98% quantiles (right column) obtained for a TCP connections of 120 packets, a network scenario

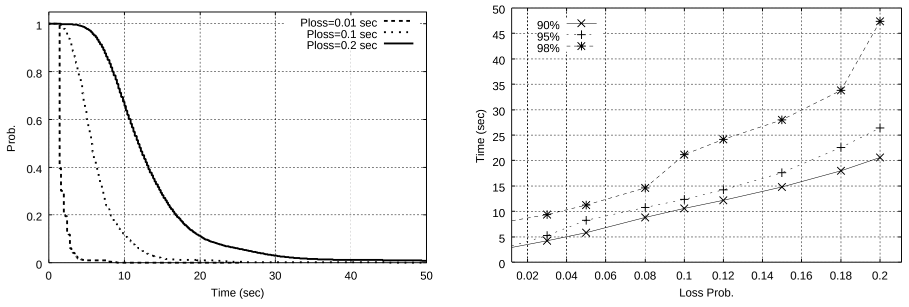


Fig. 5. Analytical results obtained for TCP connections with 50 packets, with fixed loss probability (0.05) and different values for the round trip time (0.2 sec) and different values for the loss packet probability (from 0.01 to 0.2): completion time distribution (left column) and quantiles (right column)

characterized by round trip time equal to 0.2 sec and packet loss probability equal 0.1, and different loss packet models: Bernoulli, drop-Tail, and Correlated with different correlation values ($q = 0.25, 0.5, 0.75$).

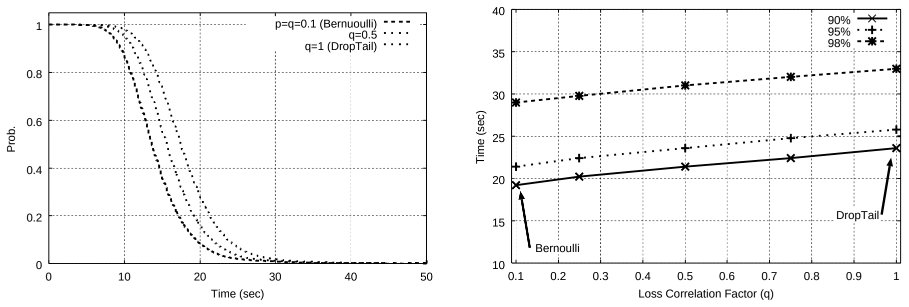


Fig. 6. Analytical results obtained for TCP connections of 120 packets, RTT equal to 0.2 sec, $p = 0.1$, and different loss packet models: Bernoulli, drop-Tail, and Correlated with different correlation values ($q = 0.25, 0.5, 0.75$). Completion time distribution (left column) and quantiles (right column).

All the plots presented so far focus on cumulative distribution functions. These functions are useful for deriving quantitative information (for instance quantiles). In the following we show two graphs that represent probability density functions (pdfs). These pdfs can give other useful information because in a certain sense the can provide a “fingerprint” of the probability distribution and the analysis of their shapes could useful for interesting considerations. Figure 7 shows two plots that present pdfs obtained for TCP connections of 50 packets (left column), and for TCP connections of 100 packets (right column). In both cases, the round trip time is equal to 0.2 sec. Each plot reports two pdfs obtained using two different packet loss probability values: 0.1 and 0.2.

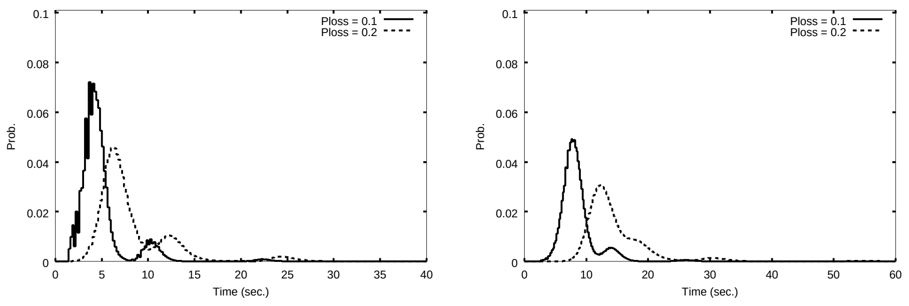


Fig. 7. Probability density functions obtained with a round trip time equal to 0.2 sec, TCP connections of 50 packets (left column), and TCP connections of 100 packets (right column)

One of the most intriguing and, probably, the most ambitious future work of this research topic is the use of the DSPN model for guessing an analytical form of the completion time distribution for TCP connections. In this case the goal would be first the fitting of the completion time distributions with some known distribution and hence the derivation of the parameters for this distribution starting from TCP and/or network parameters (number of packets to transfer, round trip time, loss probability, etc.). This very ambitious goal would yields a drastical change in performance evaluation and planning of IP networks because the TCP modeling could be replaced by the analytical form of the completion time distribution for TCP connections.

6 Conclusions and Further Developments

In this paper we have shown how DSPN models of the TCP protocol can be used to obtain accurate estimates of the distributions of the completion time of short TCP connections. The DSPN models of TCP require as input only the packet loss probability, and the average round trip time for the TCP connections being considered.

The method presented in this paper, which is based on the use of DSPN models, allows the computation of the completion time distribution by using a transient analysis of the model, it easily allows to model different type of losses (independent, correlated, etc.), and, more important, it is amenable to possible extensions, some of which are currently underway. In particular:

– we are developing DSPN models for other, more recent, TCP versions such as Reno [20] and NewReno [6];

– The DSPN models could be used for evaluating the effect of the round trip time distribution. In this case the deterministic transition that represents the round trip time must be replaced by a timed transition with general distributed firing time.
– The DSPN models can be also modified to capture other aspects of the TCP protocol and/or the underlying IP network that have not been modeled so far. In particular, it is not too complicated the modeling of aspects such as losses of ACKs (in general in the TCP modeling literature it is assumed that the ACKs are never lost). Modeling TCP connections with asymmetric data transfer rates, i.e., different forward (data packets per second) and reverse (ACKs per seconds) rates. This feature is quite common in network technology such as ADSL.
– Another possible future work concerns the derivation of a fitting methodology to derive an analytical form for the completion time distribution for TCP connections.

References

1. M. Ajmone Marsan, C. Casetti, R. Gaeta, and M. Meo. Performance analysis of TCP connections sharing a congested internet link. *Performance Evaluation*, 42(2–3), September 2000.
2. M. Allman, V. Paxon, and W. Stevens. TCP Congestion Control. Technical report, RFC 2581, April 1999.
3. N. Cardwell, S. Savage, and T. Anderson. Modeling TCP latency. In *Proc. IEEE Infocom 2000*, pages 1742–1751, Tel Aviv, Israel, March 2000. IEEE Comp. Soc. Press.
4. C. Casetti and M. Meo. A New approach to Model the Stationary Behavior of TCP Connections. In *Proc. IEEE Infocom 2000*, Tel Aviv, Israel, March 2000. IEEE Comp. Soc. Press.
5. J.C.A. de Figueiredo and L. M. Kristensen. Using Coloured Petri nets to investigate behavioural and performance issues of TCP protocols. In *Proc. of the 2nd Workshop on the Practical Use of Coloured Petri Nets and De-sign/CPN*, 1999.
6. S. Floyd and T. Henderson. The NewReno Modification to TCP,s Fast Recovery Algorithm. Technical report, RFC 2582, 1999.
7. S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transaction on Networking*, 1(4), August 1997.
8. M. Garetto, R. Lo Cigno, M. Meo, and M. Ajmone Marsan. A Detailed and Accurate Closed Queueing Network Model of Many Interacting TCP Flows. In *Proc. IEEE Infocom 2001*, Anchorage, Alaska, USA, 2001. IEEE Comp. Soc. Press.
9. R. German. *Performance Analysis of Communication Systems: Modeling with Non-Markovian Stochastic Petri Nets*. John Wiley and Sons, 2000.
10. M. Gribaudo. Transient Solution Methods for Deterministic Stochastic Petri Nets. Technical report, Università di Torino, 2002. `http://www.di.unito.it/~marcog/res.html`.
11. V. Jacobson. Congestion avoidance and control. In *Proc. ACM SIGCOMM '88*, 1988. An updated version is avaliable via ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z.

12. E. Király, M. Garetto, R. Lo Cigno, M. Meo, and M. Ajmone Marsan. Computation of the Completion Time Time Distribution of Short-Lived TCP Connections. Technical report, Politecnico di Torino, 2002.
13. C. Lindemann. *Performance Modelling with Deterministic and Stochastic Petri Nets.* John Wiley and Sons, 1998.
14. R. Lo Cigno and M. Gerla. Modelling Window Based Congestion Control Protocols with Many Flows. In *Proc. Performance 1999*, Istanbul, Turkey, 1999.
15. S. MCanne and S. Floyd. ns-2 network simulator (ver.2). Technical report, 1997. URL http://www.isi.edu/nsnam/ns/.
16. A. Ost and B. R. Haverkort. Analysis of Windowing Mechanisms with Infinite-State Stochastic Petri Nets. *ACM Performance Evaluation Review*, 26(8):38–46, 1998.
17. J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Reno performance: a simple model and its empirical validation. *IEEE/ACM Transaction on Networking*, 8(2):133–145, 2000.
18. V. Paxon. End-to-End Internet Packet Dynamics. *IEEE/ACM Transaction on Networking*, 7(3), June 1999.
19. J. Postel. Transmission Control Protocol. Technical report, RFC 793, September 1981.
20. W. Stevens. TCP Slow Start, Fast retransmit, and Fast Recovery Algorithms. Technical report, RFC 2001, IETF, Jan 1997.
21. K. Thompson, G. Miller, and R. Wilder. Wide-area internet traffic patterns and charateristics. *IEEE Network*, 11(6), Nov-Dec 1997.
22. A. Wolman, G. Voelker, N. Sharma, N. Cardwell, M. Brown, T. Landray, D. Pinnel, A. Karlin, and H. Levy. Organization-Based Analysis of Web-Object Sharing and Caching. In *Proceedings of USENIX Symposium on Internet Technologies and Systems*, October 1999.
23. Y. Zhang, V. Paxson, and S. Shenker. The Stationarity of Internet Path Properties: Routing, Loss, and Throughput. Technical report, AT&T Center for Internet Research at ICSI, http://www.aciri.org/, May 2000.
24. Y. Zhou, T. Murata, and T.A. DeFanti. Modeling and performance analysis using extended fuzzy-timing Petri nets for networked virtual environments. *IEEE Trans. on Systems, Man, and Cybernetics; B: Cybernetics*, 30(5):737–756, 2000.
25. A. Zimmermann, R. German, J. Freiheit, and G. Hommel. TimeNET 3.0 Tool Description. In 8^{th} *Intern. Workshop on Petri Nets and Performance Models*, Zaragoza, Spain, Sep 1999. IEEE-CS Press.