ELSEVIER

Performance Evaluation 36–37 (1999) 289–306

PERFORMANCE EVALUATION
An International Journal

www.elsevier.com/locate/peva

Modeling window based congestion control protocols with many flows

Renato Lo Cigno [a,*,1], Mario Gerla [b,2]

[a] *Dipartimento di Elettronica, Politecnico di Torino, Corso Duca Degli Abruzzi, 24, 10129 Torino, Italy*
[b] *Computer Science Department, 3732F Boelter Hall, UCLA, Los Angeles, CA 90095-1596, USA*

Abstract

The Markovian analysis of telecommunication networks with several interacting connections is very difficult due to the explosion of the number of states in the model. Simulation analysis under the same conditions also fails, because of prohibitive simulation times due to the enormous amount of events that must be collected to reach statistically meaningful results. This paper presents a modeling technique based on closed queuing networks, that is suitable for the evaluation of adaptive window congestion control protocols in heavy load conditions, overcoming the problem of state dimension explosion. The technique is presented applying it to a class of protocols where the window is linearly increased when the transmission is successful, and reset to 1 when packets are lost; however, the technique is general and can be applied to almost any increase and decrease policy. Results are presented for the modeled protocol class, showing the behavior as the number of competing connections increases. Then, a modification on the protocol is presented, in order to show possible ways to ameliorate the performance and, most of all, how the modeling technique can help in evaluating protocol modifications before their deployment.

Keywords: Window protocols; Queueing networks; Many flows

1. Introduction

Window based protocols are often used as the basis of flow and congestion control in packet switched, data networks. They are to be found at different levels on the protocol architecture, either operating between adjacent nodes, on a hop-by-hop basis, or end-to-end.

We are concerned here mainly with adaptive window protocols, i.e., protocols where the transmitter can dynamically adjust the transmission window size in order to adapt the amount of information injected in the network to the network conditions themselves. TCP, the Internet Transport Control Protocol [1–3], is probably the best known among adaptive window protocols, but it is not the only one.

* Corresponding author. E-mail: locigno@polito.it

[1] This work was done while Renato Lo Cigno was on leave at the Computer Science Department of the University of California, Los Angeles under grant No. 203.15.8 issued by the Italian National Research Council (CNR).

[2] E-mail: gerla@cs.ucla.edu

0166-5316/99/$ – see front matter
PII: S0166-5316(99)00029-2

We are concerned here with general properties of protocols more than with the detailed modeling of one of them.

The design of window based protocols was traditionally based on the behavioral analysis of the protocol in isolation (i.e., considering a single connection or flow) or interacting with a small number of other protocol instances (i.e., considering a small number of connections or flows). We stress that protocol design is generally not based on *performance* analysis, but on the analysis of the *behavior* of the protocol. In practice, the protocol is tested against possible loops and deadlocks, but its performance is generally assessed only through heuristic considerations.

Recently, with the booming expansion of the Internet, some researchers raised concerns about the ability of the TCP/IP protocol suite to cope with a large number of flows at the same time, and a simulation study was presented [4] showing that TCP has serious drawbacks in coping with situations where the number of active connections M is larger than the 'pipe' storage capacity R expressed as the number of outstanding packets (i.e., in transmission, propagation, queued or waiting acknowledgment). In other words the pipe storage capacity is the total number of packets that can be memorized *within* the network, either because they are actually stored in buffers waiting for transmission, or because they are 'in flight' along the transmission line or, finally, because they have been delivered to the destination, but their acknowledgments have not yet reached the source. The definition of 'many flows' is a little fuzzy; however, it can be argued that the number of connections sharing the same network resources becomes large if the following two conditions are met: (1) the network is heavily loaded leading to a non-negligible packet loss probability P_{loss}; (2) the number of connections is at least comparable to the number of packets that can be stored in the network, for the sake of ease we assume $M \geq R/2$.

This paper presents a model for the performance evaluation of window protocols, which is based on queuing networks. The model is used to show possible countermeasures to make window based protocols more resilient to the presence of a large number of connections. Recently, some papers, like [5,6], appeared in literature reporting studies on TCP performance modeling; however, to our best knowledge, none of them ever proposed the use of queueing networks. Besides all the models are specifically tailored for the TCP protocol, and do not offer a methodological framework for the analysis of window based protocols.

The remaining part of the paper is organized as follows. Section 2 presents a short overview of the window protocols we are interested in. Section 3 derives the queuing model of a sample protocol and its solution. Section 4 presents some selected numerical results and Section 5 shows how it is possible to modify the behavior of a window protocol to make its adaptation mechanism more resilient to heavy congestion and to the interaction of many flows. Section 6 closes the paper and discusses further applications of the modeling technique.

2. Window protocols overview

As pointed out in Section 1, window based protocols are very popular and they are used in several different contexts. Their behavior is indeed deeply influenced by the application they are targeting. In this paper we are concerned with end-to-end flow and congestion control protocols, although we deem that the same modeling technique can be applied to other classes of window based protocols as well.

The detailed description of adaptive window protocols is beyond the scope of this work. Fundamental theory and general descriptions can be found in any book on data networks (see for instance [7]), while

the detailed description of specific protocols is generally found in specialized literature [1,8]. We recall here only the basic features common to most of the protocols. Based on these common features a simple, but very general protocol is devised and modeled.

End-to-end adaptive window protocols rely on the flow of acknowledgments (ACKs) sent back from the receiver to adjust the window dimension and trigger new transmissions. Even if some protocols, like TCP, are byte oriented, the window dimension W can generally be expressed as a number of packets. The PDU (Protocol Data Unit) is the packet and the transmitter and receiver exchange entire packets. The packet size can be fixed or variable, but, in order to minimize the overhead due to headers, protocols generally try to send packets of the maximum allowed dimension, so that assuming fixed size packets is a reasonable approximation.

The common feature of any window based control protocol is that they can send at maximum an entire window of packets without receiving positive ACKs, then they stop transmission and wait to receive ACKs. Timeouts are set, either one per packet or one per window, and, if the ACKs do not arrive within the timeout expiration, packets are re-sent. The window adaptation mechanism is protocol-specific; however the general rule is that a steady flow of positive ACKs makes the window increase, while a timeout expiration makes it decrease.

The retransmission policy generally depends on the amount of information carried by the ACKs. Popular schemes are Go-back-N or Selective acknowledgment, though more complex schemes are widespread, like in TCP. In TCP, ACKs are cumulative and, if only one packet is lost in a window, the result is a selective retransmission, while it may happen that successfully delivered packets are retransmitted, when more than one packet per window is dropped within the network.

3. Modeling the behavior of interacting connections

A brute force approach for modeling a communication network where several connections interact is not feasible. The state of the system is described by the individual connection protocol states (the transmission window size and the timeout state) plus the number of packets stored in the network. The number of states in the model is proportional to the number of states needed to describe the protocol elevated to the number of concurrent connections. Even a trivial protocol that can be described with ten states leads to intractable models when the number of connections are only a small fraction of what we are interested in. Even resorting to special solution techniques [9] allows only the analysis of systems with a very limited number of connections.

What is needed to obtain a useful model is a modeling process that allows the use of powerful mathematical tools for the solution, such as queuing models solvable in product form.

We remark here that the modeling process described in this section is fairly general. Different protocol details would result in different queuing networks, but all share a remarkable property: they can be solved in product form.

Let us concentrate on an adaptive window protocol with additive window increase, window reset upon packet loss and multiple timeouts. Let N be the maximum window size expressed in packets for each transmitter, and $c-1$ the maximum number of consecutive timeouts (or retransmissions of the same packet); after $c-1$ consecutive timeouts the protocol closes the connection. Each transmitter can be in $N+c$ different states.

States $\{S_{W_1} \cdots S_{W_N}\}$ correspond to phases when packets are delivered correctly to the receiver; the

receiver sends back an ACK triggering a new transmission. The transmitter moves from states S_{W_i} to state $S_{W_{i+1}}$ after transmitting correctly i consecutive packets. When the state S_{W_N} is reached the transmitter remains in S_{W_N} unless a packet is lost. Whenever a packet is lost the transmitter moves to state S_{T_1}.

States $\{S_{T_1} \cdots S_{T_{c-1}}\}$ correspond to timeout phases. When a packet is dropped by the network, the transmitter waits for a timeout to expire before retransmitting it; τ_i is the value of the timeout corresponding to state S_{T_i}. The transmitter moves from states S_{T_i} to state $S_{T_{i+1}}$ if the ith retransmission of the same packet gets lost. Whenever a retransmission is successful the transmitter moves to state S_{W_1} and transmit a new packet, starting a new cycle. After $c-1$ unsuccessful retransmissions the protocol closes the connection and goes into state S_{T_c}. Since the application must eventually transmit the data, we assume that the connection is re-opened after τ_c s; whether it is the application protocol that re-opens the connection, or the user himself, may only change the value of τ_c, but not the fact that the transmitter will try to re-open the connection. Under this assumption the number of connections in the system remains constant. A new, or re-opened, connection starts in state S_{W_1}.

Our approach is to represent the state of the system with the number M_s of transmitters that are in state s. For instance M_{W_2} is the number of transmitters in state S_{W_2}. The possible number of states is still prohibitive, but, since we obtain a closed queue network solvable in product form, we can resort to mean value analysis to find a viable solution.

Consider the queuing network of Fig. 1. There are $n = N + c$ queues, each one representing a possible state for a transmitter and labeled after it (i.e., $S_{W_1}, S_{W_2}, \ldots, S_{W_N}, S_{T_1}, S_{T_2}, \ldots, S_{T_c}$). All the queues are $M/M/\infty$ since there are no limitations to the number of connections that can be in any given state. It is easy to verify that the network of queues in Fig. 1 is a closed queuing network that can be solved in product form (see for instance Chapter 3 in [7] for the relative theory).

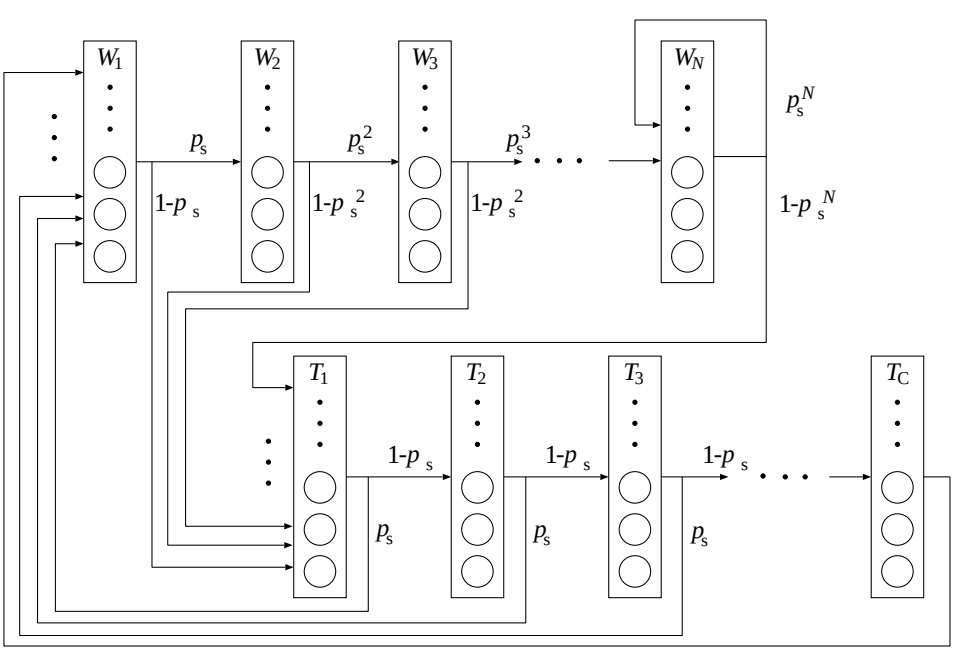

Fig. 1. Queuing model for an adaptive window protocol with linear window increase and repeated timeouts.

Queuing networks solvable in product form allow the computation of arrival rates to the individual queues solving the set of linear equations defined as

$$\lambda_i = \lambda_i^g + \sum_{j=1}^{n} p_{j,i}\lambda_j\,; \quad 1 \le i \le n \tag{1}$$

where λ_i is the arrival rate to the ith queue, $p_{j,i}$ is the probability of going to queue i after service at queue j and λ_i^g is the arrival rate from outside the network at queue i. When the queuing network is closed, as in our case, $\lambda_i^g = 0\,\forall i$, the system defined by Eq. (1) is homogeneous and admits infinite solutions. The exact solution is found forcing to 1 the sum of the state probability distribution. Equivalently, the normalization can be obtained forcing the number of customers in the system to be constant and equal to M. If the number of states in the system is large (as in our case), the computation of the normalizing factor can be cumbersome.

In the case of the presented model, most of the figures of interest can be derived from the knowledge of the mean values $\overline{M}_{W_i}$ and $\overline{M}_{T_i}$ of the number of customers in each queue. We can resort to mean state analysis and avoid the computation of the normalizing factor. Since all queues are $\mathrm{M/M/\infty}$ the dwelling times τ_i do not depend on the total number of customers in the system and are known a priori. Mean state analysis recursively computes the average number of customers $\overline{M}_i$ in each queue when the total number of customers is M, starting from the knowledge of $\overline{M}_i$ when the total number of customers is $M-1$, with the obvious observation that $\overline{M}_i = 0\,\forall i$ if $M = 0$. When the dwelling time is not dependent from the total number of customers M, mean state analysis does not require recursion and reduces to impose that

$$\sum_{i=1}^{n} \lambda_i \tau_i = M\,; \quad 1 \le i \le n. \tag{2}$$

Before proceeding further in solving the model, let us comment about the assumption that dwelling times are independent from the number of customers in the queue. Intuitively, one would expect that the service rates of queues representing the protocol state decreases as the network congestion increases because of increased queuing delays. Indeed we are modeling a network that is under very heavy overload, so that the queues can be expected to be always full and the queuing delay roughly constant.

The service rates of the individual servers of queues are readily derived from the physical and protocol characteristics of the systems. Customers leave a queue corresponding to a timeout state when the timeout expires, hence

$$\mu_{T_i} = \frac{1}{\tau_{T_i}}. \tag{3}$$

The service time of the servers in queues corresponding to non-timeout states, is instead related to the time needed to transmit W_i packets and receive the relative acknowledgments. Since we are considering a system with a large pipe, that generally implies high speed links, the transmission delay is negligible compared to propagation and queuing delays, so that the service time for these states is directly related to the pipe storage capacity R, that, we recall, is the total number of packets that can be stored within the network. In other words, it is related to the round trip time τ_r experienced by the transmitters

$$\mu_{W_i} = \frac{1}{\tau_\mathrm{r}} = \frac{C_\mathrm{p}}{R \cdot D_\mathrm{p}}\,; \quad 1 \le i \le N \tag{4}$$

where C_p is the transmission speed along the pipe in bits/s and D_p is the packet size in bits.

Eq. (4) is an approximation that holds if and only if $N < R$, otherwise the transmission delay must be taken into account.

Let p_s be the probability of a successful packet transmission. The arrival rates λ_{W_i} and λ_{T_i} to the individual queues in Fig. 1 are readily computed.

$$\lambda_{W_1} = \sum_{i=1}^{c-1} p_s \lambda_{T_i} + \lambda_{T_c} = \lambda_{T_1} \left[p_s \sum_{i=1}^{c-1} (1-p_s)^{i-1} + (1-p_s)^{c-1} \right], \tag{5}$$

$$\left.\begin{aligned} \lambda_{W_2} &= p_s \lambda_{W_1} \\ \lambda_{W_3} &= p_s^2 \lambda_{W_2} \\ &\vdots \\ \lambda_{W_{N-1}} &= p_s^{N-2} \lambda_{W_{N-2}} \end{aligned}\right\} \lambda_{W_i} = p_s^{i(i-1)/2} \cdot \lambda_{W_1}; \quad 2 \le i \le N-1, \tag{6}$$

$$\lambda_{W_N} = p_s^{N-1} \cdot \lambda_{W_{N-1}} + p_s^N \cdot \lambda_{W_N} = \frac{p_s^{N(N-1)/2}}{1-p_s^N} \cdot \lambda_{W_1}, \tag{7}$$

$$\lambda_{T_1} = (1-p_s)\lambda_{W_1} + \sum_{i=2}^{N-1} (1-p_s^i)\lambda_{W_i} + (1-p_s^N)\lambda_{W_N}, \tag{8}$$

$$\left.\begin{aligned} \lambda_{T_2} &= (1-p_s)\lambda_{T_1} \\ \lambda_{T_3} &= (1-p_s)\lambda_{T_2} \\ &\vdots \\ \lambda_{T_c} &= (1-p_s)\lambda_{T_{C-1}} \end{aligned}\right\} \lambda_{T_i} = (1-p_s)^{(i-1)} \cdot \lambda_{T_1}; \quad 2 \le i \le c. \tag{9}$$

Eqs. (5)–(9) define an homogeneous system with infinite solutions, as it is customary in a set of balance equations of a closed queuing network. The normalizing factor for the arrival rates is obtained simply by forcing the sum of the average number of customers in all queues to be equal to M.

$$\sum_{i=1}^{N} \overline{M}_{W_i} + \sum_{i=1}^{c} \overline{M}_{T_i} = M. \tag{10}$$

The last step to solve the model is the computation of the successful transmission probability p_s, which is a function of the steady state of the system and not of its physical parameters. With a pipe storage capacity of R packets, the network can support at most R packet transmissions during any given period of time τ_r, hence the average probability of a successful transmission is given by R divided by the average load offered by the transmitters during a round trip time τ_r:

$$p_s = \frac{R}{\tau_r} \cdot \left(\sum_{i=1}^{N} i \mu_{W_i} \overline{M}_{W_i} + \sum_{i=1}^{c-1} \mu_{T_i} \overline{M}_{T_i} \right)^{-1} = \frac{C_p}{D_p} \cdot \left(\sum_{i=1}^{N} i \lambda_{W_i} + \sum_{i=1}^{c-1} \lambda_{T_i} \right)^{-1}. \quad (11)$$

Notice that when the connection is closed, i.e., in queue T_c, it does not offer any load to the network.

The system defined by Eqs. (5)–(9) with the normalization factor computed with Eq. (10), plus Eq. (11) can be solved with an iterative procedure, with the assumption that the system is a contraction and will finally converge. We can start from any reasonable value of p_s, for instance R/M or the value obtained assuming equi-distribution of the customers in the queues of the model, and we stop the iteration when the relative error of p_s is smaller than a given constant ε.

$$\left| \frac{p_s(n) - p_s(n-1)}{p_s(n)} \right| < \varepsilon. \quad (12)$$

For practical purposes we use $\varepsilon = 10^{-6}$.

4. Some numerical results

We present here some results that highlight the behavior of congestion control window protocols like the one modeled in Section 3.

For the sake of easy interpretation of the results, we consider a simple protocol with maximum window dimension $N = 10$ and six possible timeout states. Timeouts are increased with geometrical backoff. Every time a complete window is transmitted successfully the window is increased by one packet until the maximum value is reached. When a packet is lost, the window is reset to 1 and the protocol waits for the timeout to expire before retransmitting it, if it is lost again the timeout is doubled. After six consecutive losses the connection closes. We (arbitrarily) assume that connections re-opens after an average time of 180 s, independently from any other network or protocol parameter. The packet size is $D_p = 8000$ bits.

The very simple protocol we are considering is completely defined by the parameters already introduced apart from one last detail: the value for the first timeout τ_1, that defines the service rates of all the T_i queues. We consider here two possibilities. In the first one, τ_1 depends on the estimated round trip time (τ_r), and we take the case $t_1 = 4\tau_{W_i}$, i.e., the first timeout is set to 4 times τ_r. In the second one, τ_1 is independent from τ_r, and we assume here $\tau_1 = 1$ s. We examine the system performance both in a LAN-like scenario (Section 4.1) and in a WAN-like scenario (Section 4.2). Although the model of the protocol behavior is identical in the two cases, in the first case τ_r and the pipe storage capacity R are dominated by the queueing time, while in the second case the propagation delay has an important role and is often dominating both τ_r and R.

The performance measures taken into account are the following.

Packet loss probability (P_{loss}). This is the probability that a packet is discarded by the network, in our case it is the parameter $1 - p_s$ computed iteratively during the model solution.

First attempt throughput (FAT). Given our modeling assumptions based on sustained, heavy overload of the network, the pipe utilization and the global throughput are necessarily 1. FAT represents the fraction of packets that are transmitted correctly at the first attempt. It is a measure of how efficiently

overall network resources are used. Indeed a low FAT means that a large amount of traffic is offered to the network, partially transmitted and then discarded. It is computed with the following formula:

$$\mathrm{FAT} = \frac{p_s D_p}{C_p} \sum_{i=1}^{N} i \mu_{W_i} \overline{M}_{W_i}. \tag{13}$$

Hence, it is a throughput normalized to the pipe transmission speed.

Timeout probability (P_{tmo}). It is the probability of being in any one of the timeout states at any given time instant excluding connections that are closed:

$$P_{\mathrm{tmo}} = \frac{\sum_{i=1}^{c-1} \overline{M}_{T_i}}{\sum_{i=1}^{N} \overline{M}_{W_i} + \sum_{i=1}^{c-1} \overline{M}_{T_i}}. \tag{14}$$

Connection close probability (P_{cl}). It is simply the probability that a connection is closed at any given time. It is equivalent to the fraction of connections closed:

$$P_{\mathrm{cl}} = \frac{\overline{M}_{T_c}}{M}. \tag{15}$$

4.1. A LAN-like case

The LAN environment is characterized by negligible propagation delays. In our case it means that τ_r is dominated by the queuing delay. With these assumptions, the pipe storage capacity does not change with the link speed. We take a pipe storage capacity $R = 1000$ packets, assuming that this is in practice the dimension of the buffer on the bottleneck link. The number of competing connections M varies from 500 to 50,000.

Fig. 2 reports plots for the four performance figures in the case of $\tau_1 = 4\tau_r$. The speed of the link C_p varies from 50 Mbit/s to 1 Gbit/s. The results show clearly that the normalized performance of the protocol does not change substantially when the link speed is increased, apart from the obvious scaling in the absolute value of the throughput. It is interesting to notice that increasing the link speed even increases the probability that a connection is closed. This is easily explained with the fact that the dwelling time in every queue decreases while C_p increases, with the exception of the queue T_c, since we have supposed that the time required to re-open a closed connection is independent from the link speed. Indeed all the performance figures become unacceptable as soon as the number of connections becomes a few times larger than the pipe storage capacity.

Fig. 3 reports plots for the four performance figures in the case of $\tau_1 = 1\,\mathrm{s}$. Comparing Figs. 2 and 3, the change induced by a different timeout policy is remarkable, and in this case the change of the link speed leads to very different performances. Indeed the difference is due to the fact that the ratio between the dwelling time in timeout states and the dwelling time in active states, increases while C_p increases, as shown by the very high timeout probability. The behavior of the protocol in this case is somewhat more desirable than in the previous one, but it is surely far from ideal, since the high timeout probability means a great spread in transfer delays and also unfairness among connections in the short term.

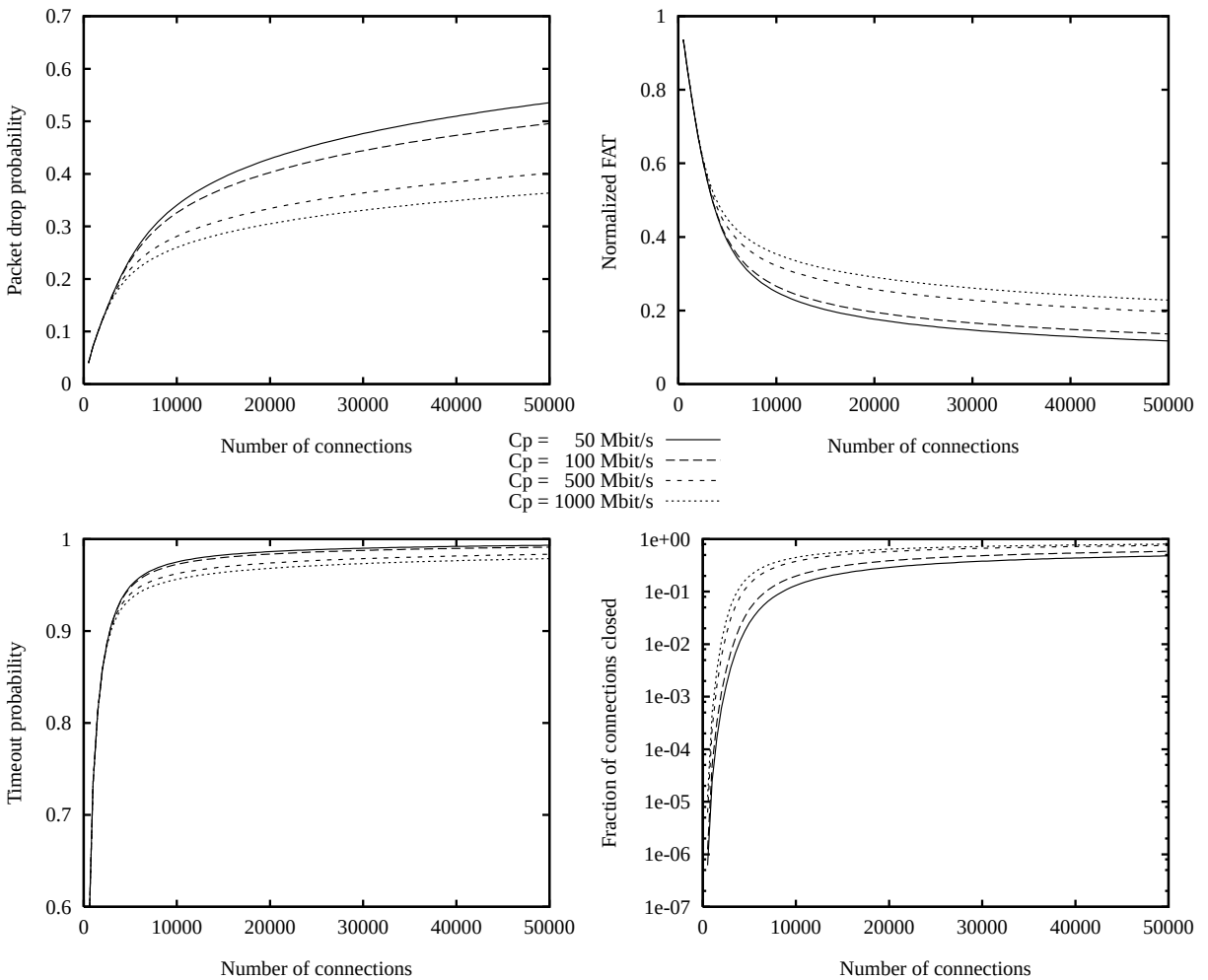

Fig. 2. P_{loss}, FAT, P_{tmo} and P_{cl} as a function of the number of connections M; LAN environment, pipe storage capacity $R = 1000$ packets and varying link speed; timeouts proportional to τ_{r}.

4.2. A WAN-like case

Let us now consider a different scenario, where the round trip time, and hence the pipe storage capacity, are dominated by the propagation delay, like in long haul connections. We consider a round trip propagation delay of 100 ms, which is suitable for intercontinental connections. Assuming optical fiber connections with signal propagation speed of 2×10^8 m/s (a typical value), this delay corresponds to 10,000 km connections.

We model SDH (Synchronous Digital Hierarchy) links and a fixed buffer size of 5000 packets. This means that the pipe storage capacity R grows from roughly 7000 packets when the link speed is 155.5 Mbit/s to nearly 130,000 when the link speed is 9.952 Gbit/s.

Fig. 4 shows the results for the case when the timeouts are based on the round trip time. As expected,

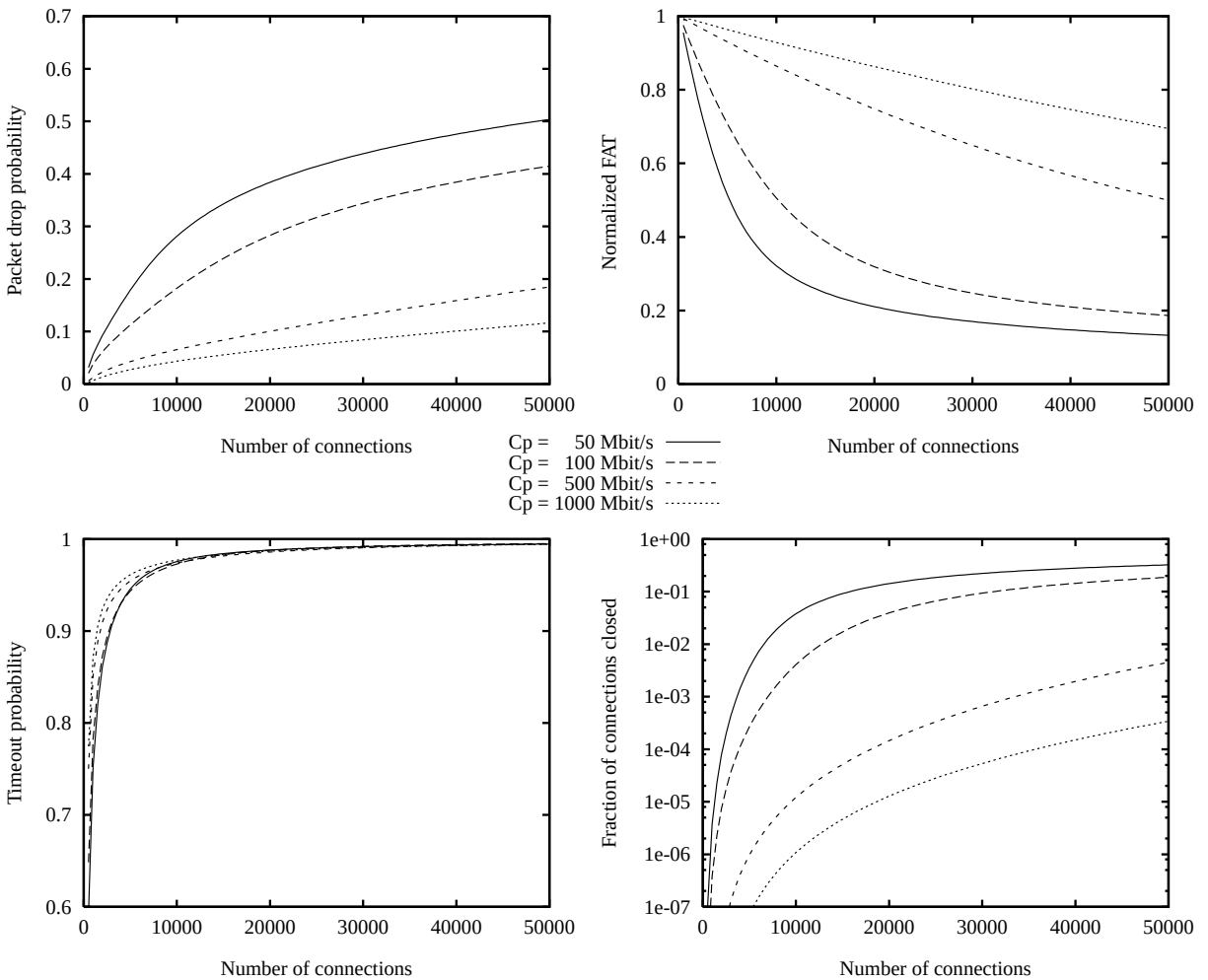

Fig. 3. P_{loss}, FAT, P_{tmo} and P_{cl} as a function of the number of connections M; LAN environment, storage capacity $R = 1000$ packets and varying link speed; timeouts independent from τ_{r}.

in this case the performance of the system increases with the link speed, mainly due to increased pipe dimension.

Fig. 5 presents the results plots for the other case, when the timeouts are independent from the round trip time. The performance difference shown by comparative analysis of Figs. 4 and 5 is in this case very limited, due to the fact that round trip times in WAN environments are much larger than in the LAN case and are also little sensitive to the link speed.

The comparison between LAN and WAN results leads to a simple, though not trivial, consideration. Increasing the link speed in LAN networks does not increase the protocol resilience to the presence of a large number of flows, while in WAN networks it does. In LANs increasing the link speed increases the performance of single connections, but not the performance normalized to the network link speed itself. Indeed this latter would be enhanced by adding buffering capacity, that increases the pipe storage capacity, rather than by increasing the transmission speed.

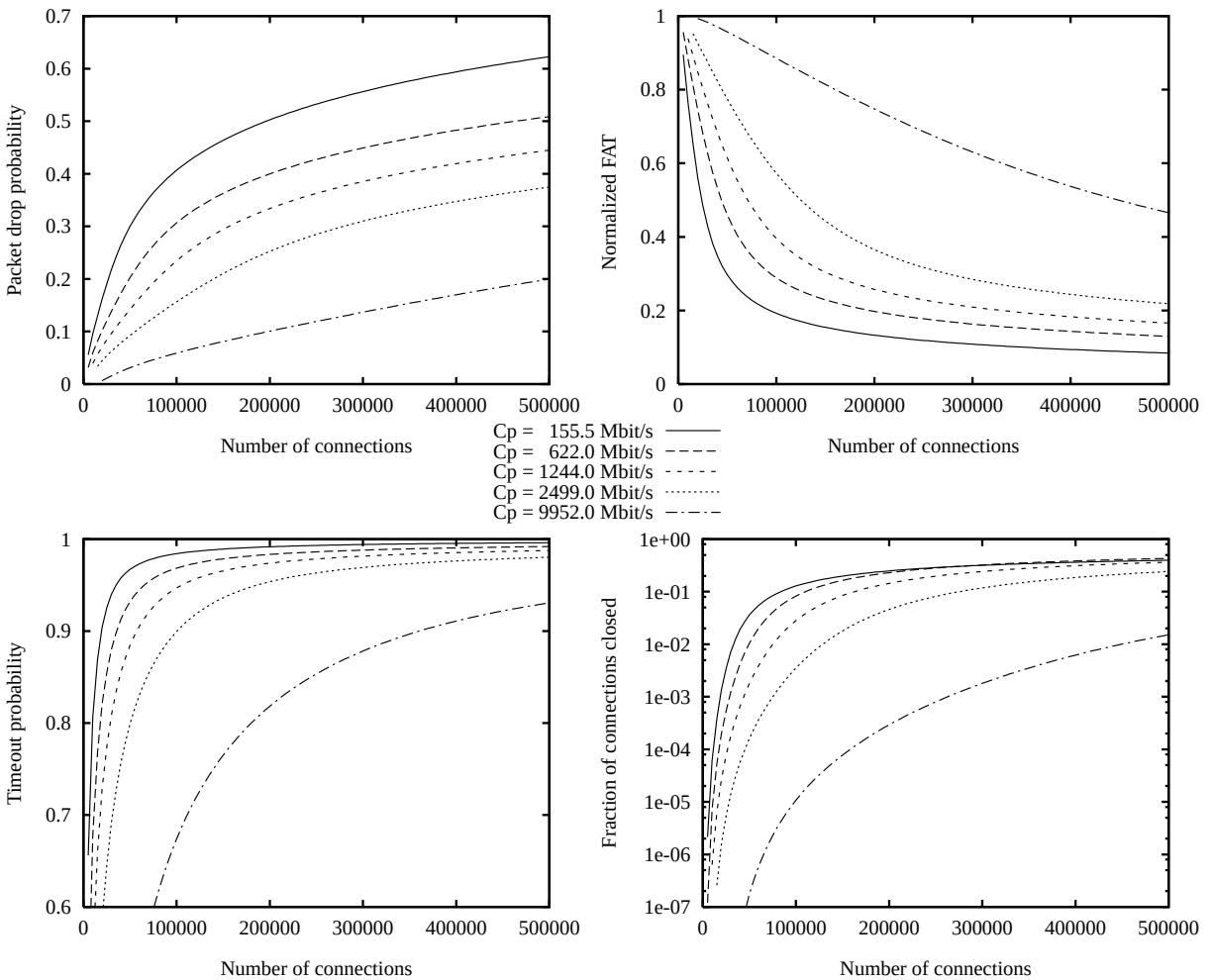

Fig. 4. P_{loss}, FAT, P_{tmo} and P_{cl} as a function of the number of connections M; WAN environment, pipe storage capacity R depending on the link speed; timeouts proportional to τ_r.

5. Window adaptation to many flows

As shown by the numerical results of Section 4, the adaptive window protocols fail to relieve congestion when the number of connections is very high. Indeed, these are cases of 'severe congestion' that need a specific adaptation to keep the congestion under control. Several directions are open to improve the network performance in this case, ranging from the introduction of connection admission control functions (but can we say that a network performs better just because we lock customers outside?) to explicit congestion control schemes. In fact, we are interested in exploring whether it is possible to improve the protocol performance with simple modifications to the protocol itself, without modifying it completely (as for instance by adding a rate control feature), and without modifying its interaction with the network infrastructure.

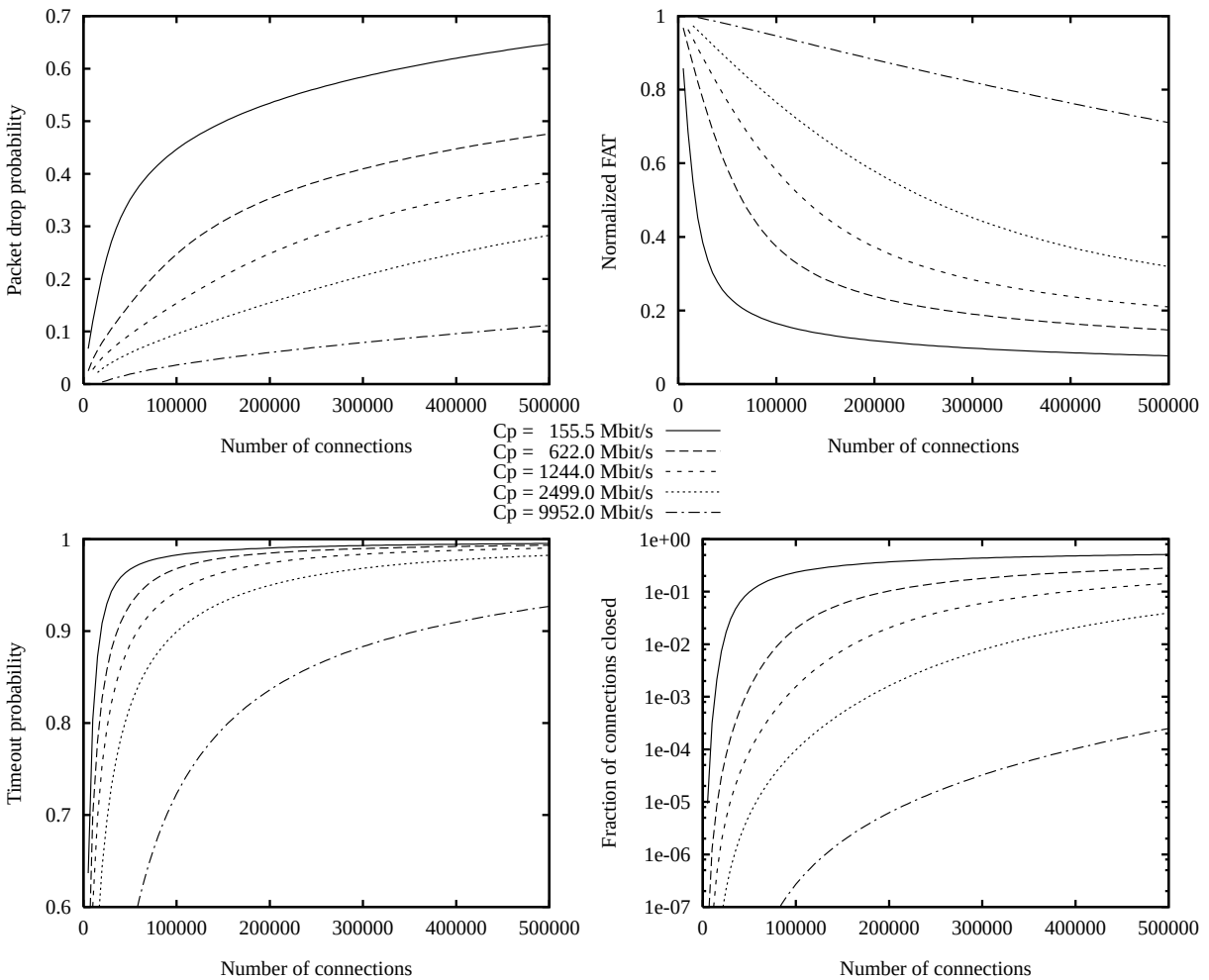

Fig. 5. P_{loss}, FAT, P_{tmo} and P_{cl} as a function of the number of connections M; WAN environment, pipe storage capacity R depending on link speed; timeouts independent from τ_{r}.

As already pointed out in [4], the oscillations between timeout states and active states where packets are transmitted every τ_{r} is a very poor control mechanism. We argue that a recovery from timeouts proportional to the length of the timeout itself would provide a better control algorithm. The rationale of this idea is that the longer the timeout has been the heavier the network congestion was (high packet loss probability), hence the return to an active state must be soft and careful.

We refer once more to the simple protocol modeled and discussed in Sections 3 and 4. In order to make their come back from timeout states softer, it is possible to introduce additional states where the transmission window is fixed to 1 packet and the transition to the next state is not immediately triggered by receiving an ACK, but only after waiting a time lapse proportional to the number of consecutive timeouts suffered.

Fig. 6 presents a queuing model for this class of protocols, and it also helps explaining the protocols

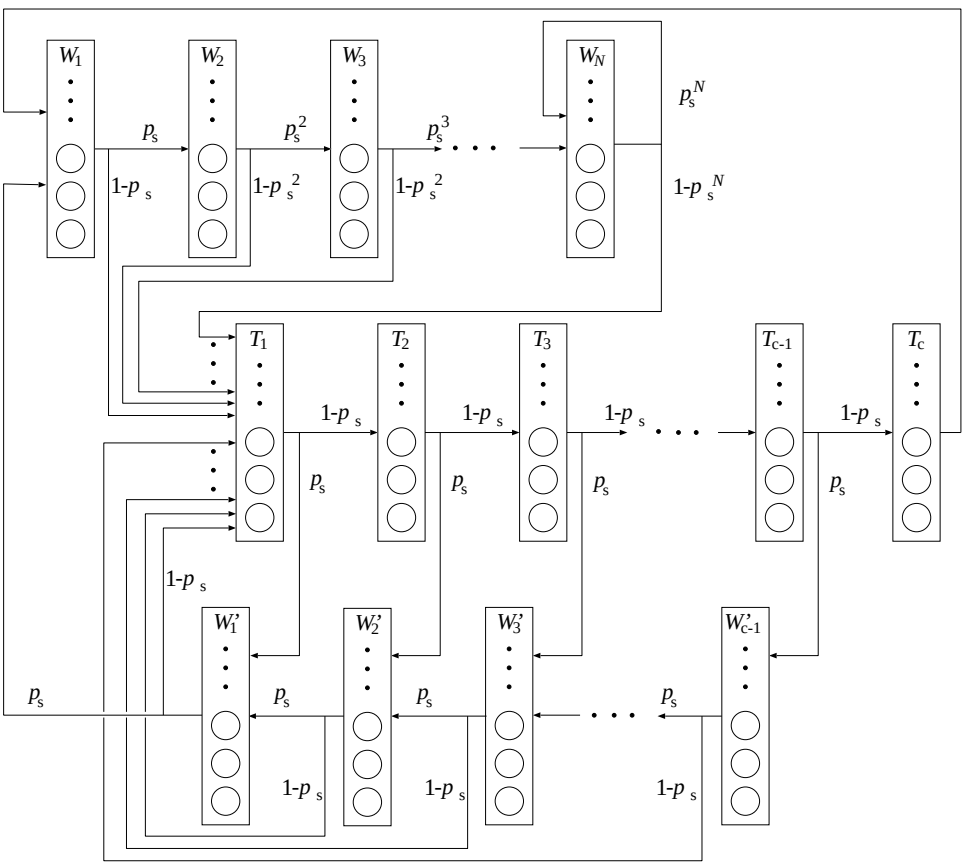

Fig. 6. Queuing model for an adaptive window protocol with linear window increase, repeated timeouts and 'severe congestion' recovery mechanism.

themselves since each queue corresponds to one possible state of the protocol. With respect to the protocols modeled in Fig. 1, the W' states are added, each one in correspondence with a timeout state and having the same transition rate $\tau_{W'_i} = \tau_{T_i}$, $1 \leq i < c$. All W' states have window dimensions equal to 1. Without going into implementation details that are not due here, we note that the protocol modification can be easily obtained with a timeout counter, that keeps track of the number of consecutive timeouts.

The queuing model defined in Fig. 6 is still solvable in product form as the one defined in Fig. 1. The equilibrium equations are the following:

$$\lambda_{W_1} = p_s \lambda_{W'_1} + \lambda_{T_c}, \tag{16}$$

$$\lambda_{W_i} = p_s^{i(i-1)/2} \cdot \lambda_{W_1}; \quad 2 \leq i \leq N-1, \tag{17}$$

$$\lambda_{W_N} = \frac{p_s^{N(N-1)/2}}{1 - p_s^N} \cdot \lambda_{W_1}, \tag{18}$$

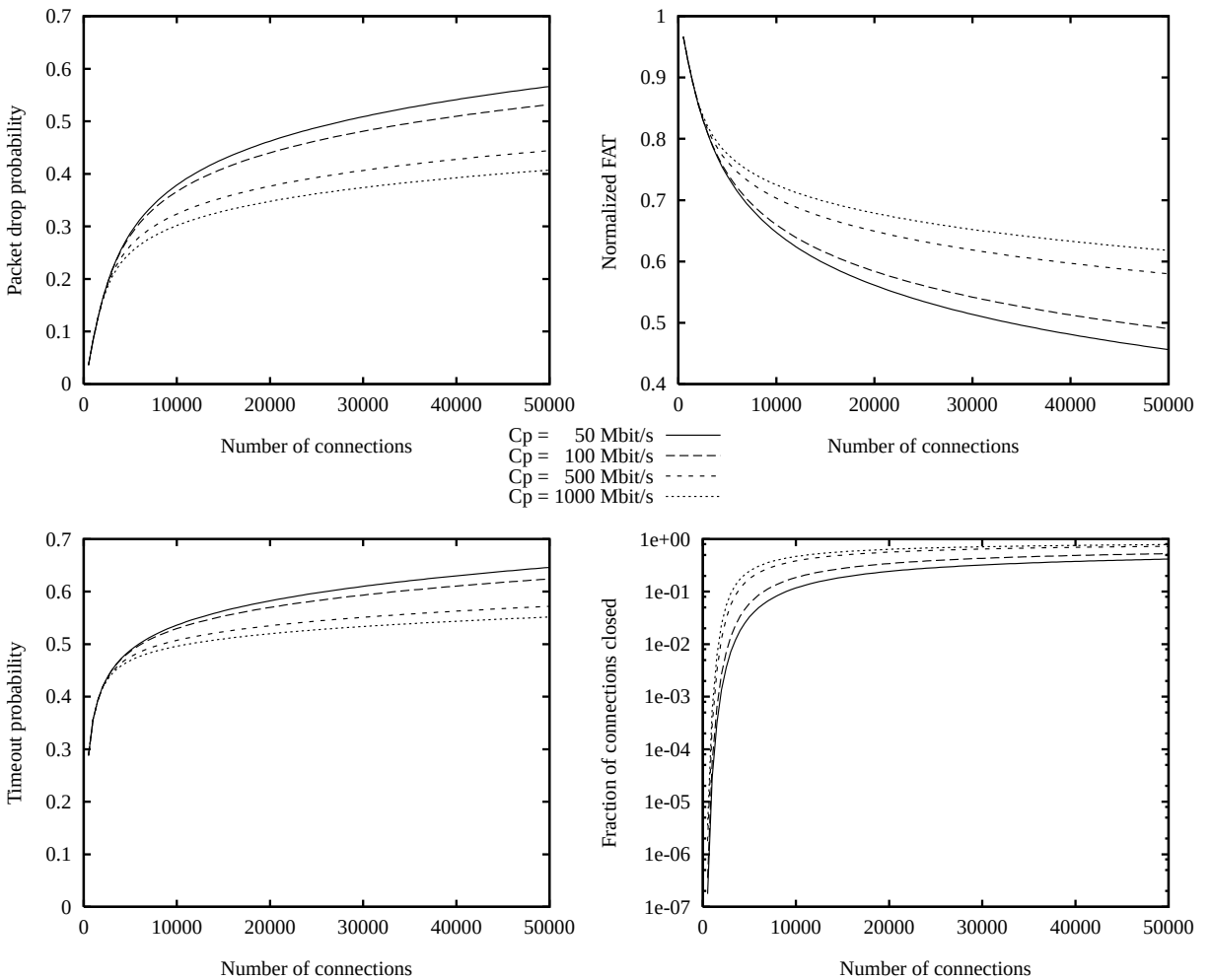

Fig. 7. Modified protocol. P_{loss}, FAT, P_{tmo} and P_{cl} as a function of the number of connections M; LAN environment, pipe storage capacity $R = 1000$ packets and varying link speed; timeouts proportional to τ_{r}.

$$\lambda_{T_1} = \sum_{i=1}^{N} (1 - p_{\text{s}}^{i}) \lambda_{W_i} + (1 - p_{\text{s}}) \sum_{i=1}^{c-1} \lambda_{W'_i}, \tag{19}$$

$$\lambda_{T_i} = (1 - p_{\text{s}})^{(i-1)} \cdot \lambda_{T_1}; \quad 2 \le i \le c, \tag{20}$$

$$\left.\begin{aligned} \lambda_{W'_{c-1}} &= p_{\text{s}} \lambda_{T_{c-1}} \\ \lambda_{W'_{c-2}} &= p_{\text{s}} (\lambda_{T_{c-2}} + \lambda_{W'_{c-1}}) \\ &\vdots \\ \lambda_{W'_1} &= p_{\text{s}} (\lambda_{T_1} + \lambda_{W'_2}) \end{aligned}\right\} \lambda_{W'_{c-i}} = p_{\text{s}} \sum_{k=1}^{i} p_{\text{s}}^{(i-k)} \lambda_{T_{c-k}}; \quad 1 \le i \le c. \tag{21}$$

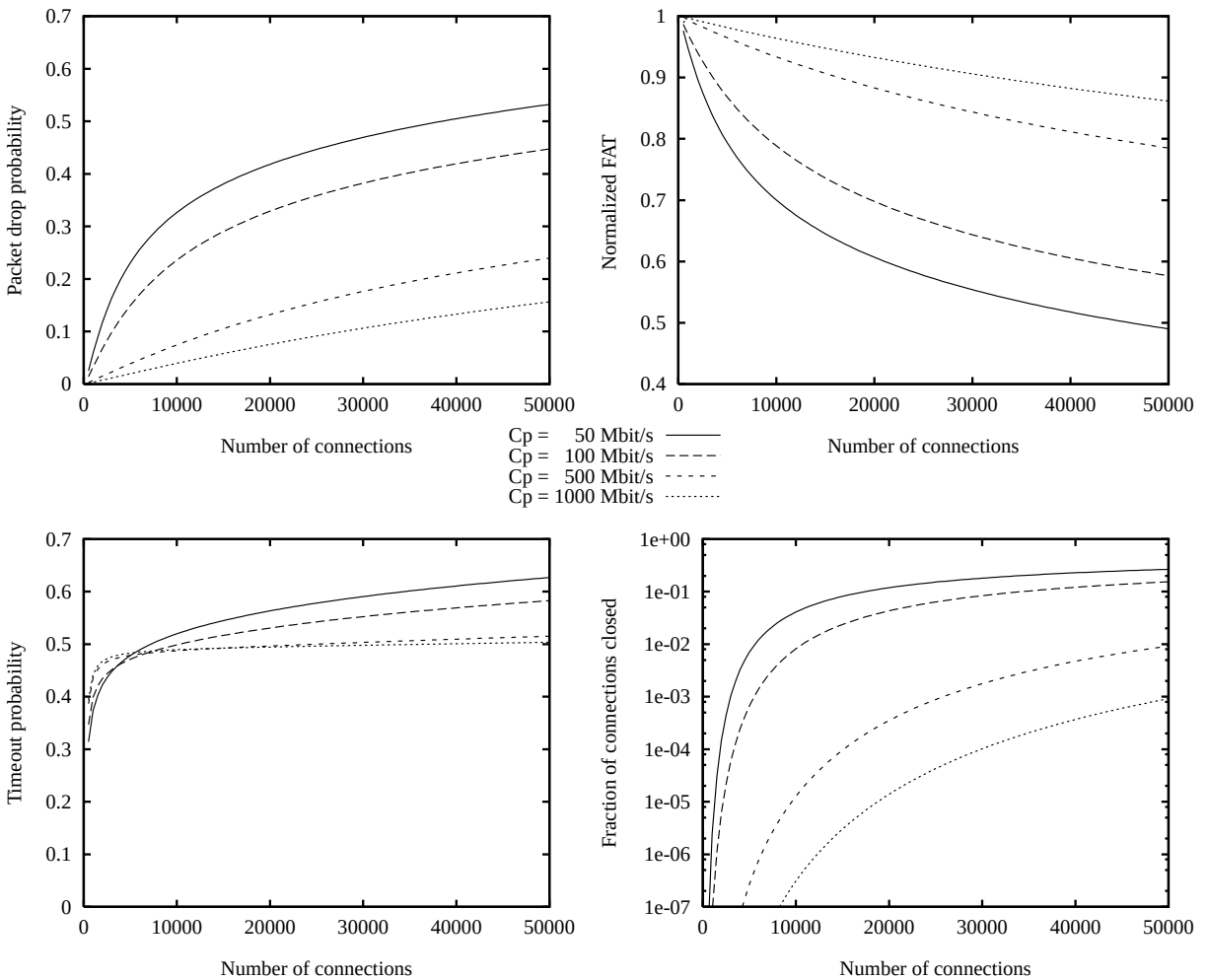

Fig. 8. Modified protocol. P_{loss}, FAT, P_{tmo} and P_{cl} as a function of the number of connections M; LAN environment, pipe storage capacity $R = 1000$ packets and varying link speed; timeouts independent from τ_{r}.

Eqs. (16)–(21) can be coupled with the two following normalizing equations and be solved iteratively as those in Section 3:

$$\sum_{i=1}^{N} \overline{M}_{W_i} + \sum_{i=1}^{c} \overline{M}_{T_i} + \sum_{i=1}^{c-1} \overline{M}_{W'_i} = M, \tag{22}$$

$$p_{\text{s}} = \frac{C_{\text{p}}}{D_{\text{p}}} \cdot \left[\sum_{i=1}^{N} i\lambda_{W_i} + \sum_{i=1}^{c-1} \lambda_{T_i} + \sum_{i=1}^{c-1} \lambda_{W'_i} \right]^{-1}. \tag{23}$$

Figs. 7–10 report the same performance results of Figs. 2–5, for the protocol modeled in this section. The performance improvement in terms of FAT and reduced timeout probability P_{tmo} is striking (notice

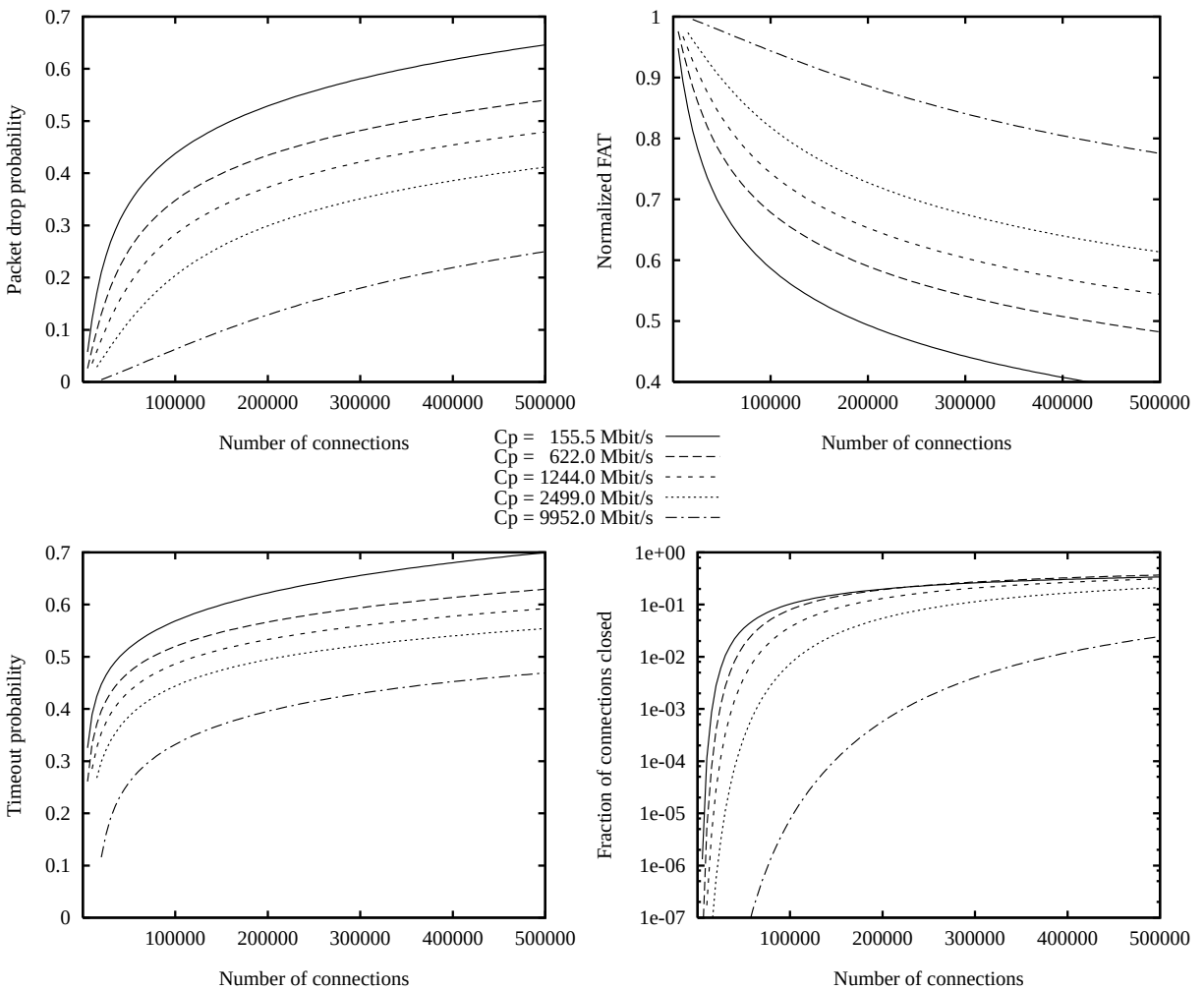

Fig. 9. Modified protocol. P_{loss}, FAT, P_{tmo} and P_{cl} as a function of the number of connections M; WAN environment, pipe storage capacity R depending on link speed; timeouts proportional to τ_{r}.

that the scale on the plots has been changed to avoid squeezing the curves). Unfortunately the packet loss probability, as well as the connection closing probability do not improve as desired.

This example does not pretend to offer a solution to any real protocol, but has been developed as a means to show how the proposed modeling technique can be used in studying protocol modifications and their dynamic in conditions where other modeling techniques or simulations fail. Other performance measures, as the window size distribution, can also be obtained from the model.

6. Conclusions and future developments

This paper presents an innovative modeling technique for the performance analysis of window based adaptive congestion control protocols, when the network is heavily loaded and the number of

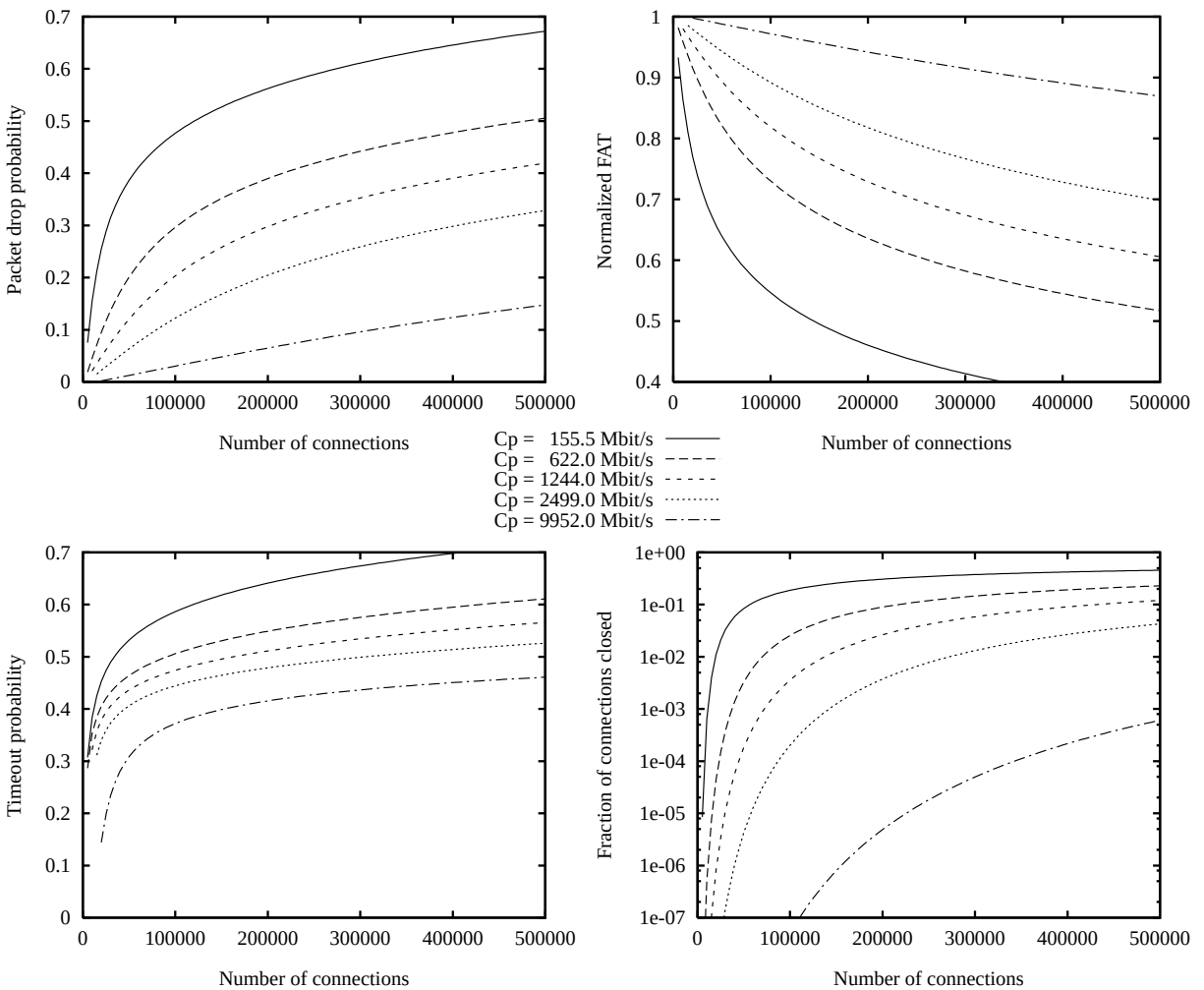

Fig. 10. Modified protocol. P_{loss}, FAT, P_{tmo} and P_{cl} as a function of the number of connections M; WAN environment, pipe storage capacity R depending on link speed; timeouts independent from τ_{r}.

connections is large. The modeling technique is based on a closed queueing network that can be solved in product form. The innovative key points of the modeling process are the following:

- Each queue in the model represents one of the possible protocol states.
- The global state of the model is represented as the number of connections that are in the same state (i.e., at any given queue), instead of trying to represent the individual connections state.
- The packet loss probability is evaluated with an iterative procedure as a function of the steady state of the system.

The third point is extremely important, since trying to explicitly represent the packets leads to queuing models that can not be solved in product form because of packet losses.

The modeling technique was applied to a simple class of protocols, discussing the performance as the number of connections sharing the network resources increases. Then, a possible modification of the protocol to make it more resilient to severe congestion conditions is proposed and briefly discussed.

The paper is focused on the methodological aspects, as well as on the possibility of obtaining results that can not be obtained either via traditional modeling or via simulation, while it does not try to model a specific protocol with all its implementation details. The prosecution of the work will be focused on the modeling of real protocols, and in particular of TCP, validating the model results against detailed numerical simulation obtained adapting the protocol code to run on the simulator or, if possible, against measures in real networks.

The model can be further enhanced by considering an open queueing model, i.e., non-greedy connections. Organizing the customers in classes, that depend on the amount of information to be transmitted, it is possible to evaluate the time needed to transfer a given amount of information.

References

[1] R. Stevens, TCP/IP Illustrated, Vol. 1, Addison-Wesley, Reading, MA, 1994.
[2] Van Jacobson, Congestion avoidance and control, ACM SIGCOMM '88, Stanford, CA, 1988.
[3] Van Jacobson, Berkeley TCP evolution from 4.3-tahoe to 4.3-reno, Eighteenth IETF, Vancouver, BC, 1990.
[4] R. Morris, TCP Behavior with many flows, Proc. IEEE ICNP'97, Atlanta, GE, 1997.
[5] A. Kumar, J. Holtzman, Comparative performance analysis of versions of TCP in a local network with lossy links, IEEE/ACM Trans. Networking 5 (3) (1998) 336–350.
[6] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, Modeling TCP throughput: a simple model and its empirical validation, Proc. ACM SIGCOMM'98, 1998.
[7] D. Bertsekas, R. Gallager, Data Networks, Prentice-Hall, Englewood Cliffs, NJ, 1992, 2nd ed.
[8] OSI Transport Protocol Specification, Standard ISO-8073, 1986.
[9] M. Meo, E. de Souza e Silva, M. Ajmone Marsan, Efficient solution for a class of Markov chain models of telecommunication systems, Performance Evaluation 27/28 (1996) 603–625.