Telecommunication Systems 25:1,2, 129–155, 2004

A Markovian Model for TCP Analysis in a Differentiated Services Network

CHADI BARAKAT * and EITAN ALTMAN {cbarakat;altman}@sophia.inria.fr
INRIA, 2004, route des Lucioles, 06902 Sophia Antipolis, France

Abstract. In a Differentiated Services network, the use of TCP by an application impacts the service it gets from the network. TCP congestion control algorithms are designed to provide a fair sharing of resources in a best effort network as the current Internet. TCP is not conscious of the new services proposed by DiffServ, namely the different priorities packets are injected with into the network. Many schemes have been proposed to support TCP traffic in a DiffServ network. These schemes have been often validated with simulations. In this paper we propose an analytical model to study the performance of TCP in a DiffServ network under the different proposed schemes. The model is based on a Markovian fluid approach. We present first a general version of the model, then we specify it to the different proposed schemes. For each scheme, we compute the throughput achieved by a TCP connection. We compare then the service differentiation provided by the proposed schemes under different subscription levels, different reservations, and different round-trip times. Our model forms a good tool for the evaluation of new solutions to support TCP traffic in a DiffServ network.

Keywords: differentiated services, TCP, buffer management, Markovian modeling, performance evaluation and comparison

Introduction

There has been an increasing interest these last years in enhancing the traditional best effort service of the Internet to provide new applications with some guarantees in terms of bandwidth, losses, and end-to-end delay. Differentiated Services architecture (DiffServ) is considered as the most promising approach in this field for reasons of scalability and incremental deployment [Clark and Fang, 5; Nichols et al., 18]. Flows asking for a better service than the traditional best effort one are monitored at the edge of the network. Their parameters (rate, burst size) are compared to the contract signed between the user and the service provider. Packets compliant with the contract are marked with a high priority. Those violating the contract are shaped, rejected, or injected into the network with a low priority. The priority level of packets is carried in the DS field of the IP header. In the core of the network, high-priority packets are privileged over low-priority ones. This privilege can be in the form of a better scheduling (e.g., priority scheduling) as with the Premium service architecture [Jacobson et al., 14; Nichols et al., 18], or in the form of a lower drop probability as with the Assured service architecture [Clark and Fang, 5; Heinanen et al., 11]. The main advantage of the DiffServ framework is that packets in

* Corresponding author.

the network are treated as a function of their priority level and not as a function of the flow they belong to. No per-flow information needs to be stored in routers. This eases the task of routers which have to deal with a small number of priority levels rather than a large number of flows. Complexity is pushed to the edge of the network. This makes the framework scalable, flexible, and easy to introduce into the Internet.

The utility of such framework to uncontrolled flows is clear. An uncontrolled flow is a one whose rate is not reduced when packets are lost in the network. These flows are generated using UDP, the best effort transport protocol. An example is a real time video or audio flow. An application using UDP has an idea on the rate of its packets. It asks the network for a certain bandwidth as a function of the quality of service it desires. The network can reject or mark with low priority the packets of this application exceeding the reserved bandwidth. What the network guarantees is a small drop probability for high-priority packets. The drop probability for low-priority packets depends on the load of the network. The application gets then a minimum quality as a function of the bandwidth it reserves, and this quality improves when low-priority packets succeed to cross the network.

The problem of DiffServ appears with transfers using TCP [Jacobson, 13], the reliable connection-oriented transport protocol of the Internet. The congestion control algorithms of TCP are well suited to the current best effort service. The window increases until a packet is lost. The TCP source assumes here that the network is congested and reduces its window consequently. This guarantees a good utilization and a fair sharing of network resources. But with the proposed DiffServ architecture, an application using TCP may ask the network for a better service (e.g., more throughput) by reserving a certain bandwidth. If at the edge of the network non-compliant packets are rejected, TCP will reduce its rate when it reaches the reserved bandwidth. The application using TCP fails then to use the bandwidth it is paying for as well as any unreserved bandwidth in the network. The solution to this problem is to let non-compliant packets get into the network as low-priority packets. This improves TCP performance since the rate can now reach larger values. But, the injection of packets of different priority levels causes additional problems: TCP is not aware of the reservation. The loss of a low-priority packet is not distinguished from the loss of a high-priority packet, and the rate is reduced in the same manner regardless of the reserved bandwidth. In a DiffServ network, the loss of a packet of a certain priority level means a congestion of the resources associated to this priority level, and not of all the resources as in a best effort network. TCP does not see this partitioning of resources and it divides its window by two whenever a loss occurs.

Many works have studied this misbehavior of TCP in a DiffServ network and many improvements have been proposed [Basu and Wang, 4; Clark and Fang, 5; Feng et al., 7, 8; Yeom and Reddy, 21, 22]. The main conclusion of these studies is that TCP is unable to realize its target throughput in a DiffServ network. The target throughput of a TCP connection is defined as the reserved bandwidth plus a fair share of any unreserved bandwidth. Moreover, a connection with a small reservation has been shown to achieve better performance than a connection with a large reservation. Indeed, a connection with

a large reservation has a large window and it is more affected by the loss of a low-priority packet than a connection with a small reservation. These works have also shown the well known problem of TCP unfairness in presence of different round-trip times (RTT). A connection with small RTT achieves better performance than a connection with long RTT. Some solutions have been proposed to alleviate these problems. These solutions consist in either changing TCP sources, or marking TCP flows differently at the edge of the network, or changing the behavior of network routers.

The performance of the different schemes proposed in the literature to support TCP traffic in a DiffServ network has been often evaluated by simulations [Clark and Fang, 5; Feng et al., 7, 8; Yeom and Reddy, 21]. In [Yeom and Reddy, 22], a mathematical model has been proposed to calculate the throughput of a connection as a function of the drop probability of packets from different priority levels. Three schemes have been compared. However, the extension of this work to other possible schemes is not straightforward. Further, it does not allow to study the impact of the parameters of the other connections (e.g., RTT, reserved bandwidth) on the performance of the connection under study. All the exogenous traffic is modeled in [Yeom and Reddy, 22] with the packet drop probability, and the dynamics of the connection under study is assumed to have no impact on this probability. This might be the case of a large number of connections sharing the network, but it is not certainly the case when a small number of connections is multiplexed.

In this paper we present a general Markovian model able (i) to calculate the performance of all the connections sharing a bottleneck router, and (ii) to account for the different solutions already proposed, or to be proposed. Using this model, we compare the performance of some schemes proposed to support TCP in a DiffServ network. In the next section we present a brief overview of the different schemes we consider in this paper. In section 2 we explain our Markovian model. In section 3 we compute the throughput of a TCP connection as a function of two functions: the reaction of a TCP connection to congestion signals, and the probability that a particular connection reduces its rate upon congestion. By appropriately setting these two functions, we are able to specify our model to all proposed schemes. Section 4 explains how these two functions must be set. In section 5 we simplify the model in the case of large number of connections, and we give closed-form expressions for TCP throughput. In section 6 we present some numerical results and compare the different schemes. The paper is concluded in section 7.

1. TCP in a DiffServ network

We summarize first the main objectives of an ideal DiffServ scheme supporting TCP traffic:

- The available bandwidth must be efficiently utilized.
- In the case when the sum of reservations is less than the total available bandwidth (the under-subscription case), each connection must realize its reservation. The difference

between the total available bandwidth and the total reservation must be shared equally by the different connections.

- In the case when the sum of reservations is larger than the total available bandwidth (the over-subscription case), the available bandwidth must be distributed among the different connections proportionally to their reservations.

Note that even if an efficient admission control algorithm is deployed, the over-subscription case could happen on a certain link due to the dynamic routing inside a DiffServ domain.

The original proposition to support TCP transfers in a DiffServ network is due to Clark and Fang [5]. At the edge of the network, the transmission rate of the TCP connection is compared to the reserved bandwidth. A time sliding window mechanism (TSW) has been proposed to measure the rate of the connection. The time window determines how much the past is important, or in other words it determines the time interval over which the rate of TCP is averaged. The transmission rate of a window-based flow control protocol as TCP can be defined as the window size divided by the RTT of the connection. Thus, the time window must be in the same order of the RTT. Two priority levels have been proposed in [Clark and Fang, 5] for packet marking. A packet that arrives at the edge of the network and finds the rate of the connection smaller than the reservation, is marked with a high priority and is called an IN packet. A packet that arrives and finds the rate of IN packets equal to the reservation, is marked with a low priority and is called an OUT packet. In network routers, IN and OUT packets are buffered in the same queue and are scheduled in a FIFO manner. This guarantees an in-order delivery of packets which is necessary for TCP operation. The differentiation in the service comes from the different probabilities network routers drop these two types of packets at the onset of congestion. A variant of RED (Random Early Detection) [Floyd and Jacobson, 10] is proposed to implement this difference in the drop probability. This variant, called RIO (RED IN/OUT) [Clark and Fang, 5], has two minimum thresholds instead of one. As with RED, the average length of the queue is calculated using an exponentially weighted moving average algorithm. When this average length exceeds the lower minimum threshold, OUT packets are probabilistically dropped in order to signal congestion to TCP sources. The buffer starts to drop probabilistically IN packets when the average queue length (sometimes the average number of IN packets in the buffer) exceeds the upper minimum threshold. Note that instead of dropping packets, the router can signal congestion to TCP sources by setting the ECN (Explicit Congestion Notification) bit in the IP header [Floyd, 9].

As we described in the introduction, this scheme has some problems to satisfy the objectives we defined at the beginning of the section. Due to the saw tooth window variation of TCP (figure 1), a connection is obliged to transmit a certain amount of OUT packets in order to realize its reservation. Since OUT packets are very likely to be dropped, the connection may not realize its reservation. Moreover, a connection with a large reservation must transmit more OUT packets than a connection with a small reservation. Also, a connection with a large reservation has in general larger window

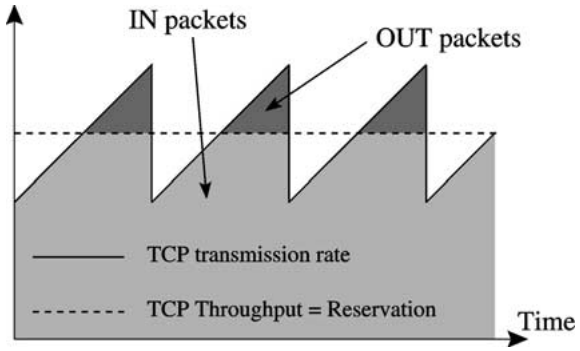

Figure 1. The saw tooth variation of TCP window.

than a connection with a small reservation which makes it more affected by the loss of an OUT packet. This explains the bias of the scheme proposed in [Clark and Fang, 5] against connections with large reservations.

The first and the most intuitive solution to this problem is to change TCP in a way that the source reduces differently its window when OUT or IN packets are lost [Feng et al., 7; Yeom and Reddy, 21]. To do that, the source needs to know the priority level of the lost packet. Also, it needs to know the bandwidth reserved by the connection. The loss of an IN packet is an indication that the network is congested and that the congestion window needs to be divided by two as in standard TCP. The loss of an OUT packet is an indication that the unreserved bandwidth in the network is congested. The source divides then its window into two parts. The first part corresponds to the number of unacknowledged IN packets and the second part to the number of unacknowledged OUT packets. The number of unacknowledged IN packets is estimated as the product of the reserved bandwidth and the RTT. The window is reduced by half the number of unacknowledged OUT packets. The main problem with this solution is that it requires a change at the source and a knowledge of the priority level of the lost packet, which is difficult to implement.

The other solutions try to give some advantage to connections with large reservations over connection with small reservations. The objective is to help the former connections to send more OUT packets than the latter ones, and this is without changing the TCP protocol. The first solution proposed in the original paper describing RIO [Clark and Fang, 5] is based on the saw tooth variation of TCP window. On average, the transmission rate of a TCP connection varies between $2/3$ and $4/3$ of its throughput. To be able to realize its reservation, a TCP connection must be protected from the other connections until it reaches $4/3$ of its reservation. The idea of [Clark and Fang, 5] is to change the marker so that it marks packets as OUT when the rate of the connection exceeds $4/3$ of the reserved bandwidth. We call this proposition the *Saw Tooth Marking* scheme. It has the drawback of injecting into the network during some periods more IN packets than what is promised.

The second solution [Yeom and Reddy, 21] proposes to drop OUT packets in network routers according to the reserved bandwidth. The authors in [Yeom and Reddy, 21]

show that dropping OUT packets with a probability inversely proportional to the bandwidth reserved by the connection improves the performance. We call this solution the *Inverse Drop Probability* scheme. Its main drawback is that it requires that network routers know the bandwidth reserved by every connection.

The last scheme we consider is the one that proposes to mark packets with three priority levels instead of two [Heinanen et al., 11, 12; Yeom and Reddy, 21]. A RED buffer with three thresholds is used in routers. The idea is to protect the OUT packets of a connection transmitting at less than its fair share from the OUT packets of a connection exceeding its fair share, and this by giving packets of the former connection some medium priority while giving low priority to those of the latter connection. However in this case, and in contrast to Saw Tooth Marking, we are not injecting into the network more high-priority packets than what is promised. This solution can be considered as a means to give priority to some OUT packets over other OUT packets but not over IN packets.

2. The Markovian fluid model

We outline in this section our Markovian model for the evolution of the rate of a TCP connection sharing a path with other TCP connections in a DiffServ network. Consider N TCP connections sharing a bottleneck of bandwidth μ. Let $X_i(t)$ be the transmission rate of connection i at time t. It is equal to the window size divided by the RTT of the connection. The N connections increase their rates (by increasing their windows) until the network gets congested. The congested router starts then to drop packets in order to signal the congestion to TCP sources. A source receiving a congestion signal reduces its rate, then it resumes increasing it. The other sources continue increasing their rates. This continues until the next congestion event.

Let t_n denote the time at which the nth congestion event occurs, and let $D_n = t_{n+1} - t_n$. Denote by $X_{i,n}$ the transmission rate of connection i at time t_n and by $X_{i,n}^+$ its transmission rate just after t_n (after the disappearance of the congestion). $X_{i,n}^+$ is equal to $X_{i,n}$ if connection i did not reduce its rate at t_n, and to $R_i(X_{i,n})$ otherwise. $R_i(X_{i,n})$ is a function of $X_{i,n}$ usually equal to $X_{i,n}/2$ [Jacobson, 13].

We introduce now some assumptions in order to analyze the process $\{X_{1,n}, X_{2,n}, \ldots, X_{N,n}\}$. In the following, we use this process to calculate the throughput achieved by each connection.

Assumption 1. We assume first that queueing times in network nodes are small compared to the propagation delay. This holds with active buffer management techniques as RED [Floyd and Jacobson, 10]. Among many others, RED aims to reduce the length of queues in network routers in order to reduce the end-to-end delay. A RED router starts to drop packets before the overflow of the buffer. The remaining space in the buffer is used to absorb bursts of packets. The RTT of a connection, say i, is then approximately constant denoted by T_i, and the transmission rate of the connection varies linearly with the window size. The congestion appears when the sum of the rates of all connections

reach the total available bandwidth μ. Thus, instants t_n are given by[1]

$$\sum_{i=1}^{N} X_i(t_n) = \mu. \tag{1}$$

Assumption 2. We consider long TCP transfers and we suppose that the sources are always sending in the congestion avoidance mode [Jacobson, 13; Stevens, 20]. Slow start phases are ignored. This is possible since the new versions of TCP avoid this phase in most of the cases [Fall and Floyd, 6]. Using assumption 1 and the results from [Lakshman and Madhow, 17], the rate of a connection can be considered to increase linearly as a function of time. This increase continues until the source receives a congestion signal, where it reduces its rate and starts again its linear increase. Thus, $X_{i,n+1} = X_{i,n}^{+} + \alpha_i D_n \cdot \alpha_i$ is a constant function of T_i and of the frequency of ACKs sent by the TCP receiver. During congestion avoidance, the congestion window increases by one packet when a window's worth of ACKs are received [Stevens, 20]. Thus, α_i is equal to $1/T_i^2$ when the receiver acknowledges every data packet and to $1/(2T_i^2)$ when it acknowledges every other data packet (Delay ACK mechanism enabled [Stevens, 20]).

Assumption 3. The third assumption we make is that only one connection reduces its rate upon a congestion, and that the probability that a connection reduces its rate is a function of its rate and the rates of the other connections at the moment of congestion. This is again the aim of the new buffer management techniques (e.g., RED [Floyd and Jacobson, 10]) that implement random drop in order to send congestion signals to connections consuming more than their fair shares of the bottleneck bandwidth while protecting connections consuming less than their fair shares [Floyd and Jacobson, 10]. We further assume that the reaction of the connection receiving the first congestion signal is quick so that the congestion in the network disappears before other packets from other connections are dropped.

Let $U_{i,n}$ be a random variable equal to 1 if source i reduces its rate at time t_n and to 0 otherwise. We always have $\sum_{i=1}^{N} U_{i,n} = 1$, since only one connection is supposed to reduce its rate upon congestion (assumption 3). The probability that $U_{i,n}$ is equal to one is a function of the all rates at time t_n. Let $p_i(X_{1,n}, X_{2,n}, \ldots, X_{N,n})$ denote this probability. It represents the probability that the dropped packet upon congestion belongs to connection i. This probability together with $R_i(X_{i,n})$ form the two functions of our model that need to be specified in order to cover all the proposed schemes. Later, we explain how to specify these two functions.

Theorem 1. The process $\{X_{1,n}, X_{2,n}, \ldots, X_{N,n}\}$ can be described as a homogeneous Markov process of dimension $N - 1$.

[1] Equation (1) assumes that rates of connections are fluid. If rates of connections are discrete, t_n will be the first instant at which the sum of rates is larger than μ.

Proof. For any congestion event n, the transmission rates of the N connections are related by (1). Thus, the problem can be analyzed by considering only the rates of $N-1$ connections. In the particular case of $N=2$, we get a one-dimensional model.

Concerning the Markovian property of the model, it is easy to show that the state of the process of rates at time t_{n+1} depends only on its state at time t_n. Indeed, for any i and any n we have,

$$X_{i,n+1} = X_{i,n} + U_{i,n}\big(R_i(X_{i,n}) - X_{i,n}\big) + \alpha_i D_n. \tag{2}$$

Summing over all the i and using (1), we get

$$D_n = \frac{\sum_{i=1}^{N} U_{i,n}(X_{i,n} - R_i(X_{i,n}))}{\sum_{i=1}^{N} \alpha_i}. \tag{3}$$

Given that $R_i(X_{i,n})$ and the value taken by $U_{i,n}$ are only a function of the process state at time t_n, $X_{i,n+1}$ is therefore only a function of the process state at time t_n and the Markovian property holds. The process is homogeneous since the process state at time t_{n+1} depends only on its state at time t_n and not on n. □

Using the recurrence (2) and the probability function p_i, we can define all the transitions of our Markov chain. Denote by $\mathcal{X}$ the state space of this chain upon congestion. For each state $X = (x_1, \ldots, x_N) \in \mathcal{X}$ (x_i is the transmission rate of source i upon congestion), the chain can jump to N different states at the next congestion event. This depends on which source reduces its rate at the current congestion event. Denote by $F_i(X) = (f_{i,1}, \ldots, f_{i,N})$ the next state of the Markov chain, given that the Markov chain is in state X and that the source which reduces its rate is i. Using (2), for $j = 1, \ldots, N$, and for any $X = (x_1, \ldots, x_n) \in \mathcal{X}$, we can write

$$f_{i,j} = \begin{cases} x_j + \dfrac{(x_i - R_i(x_i))\alpha_j}{\sum_{m=1}^{N} \alpha_m} & \text{if } j \neq i, \\ R_i(x_i) + \dfrac{(x_i - R_i(x_i))\alpha_i}{\sum_{m=1}^{N} \alpha_m} & \text{if } j = i. \end{cases}$$

Denote by $\Pi = (\pi_X)_{X \in \mathcal{X}}$ the stationary distribution of our Markov chain. To compute this distribution, we discretisize the Markov chain by supposing that the rate of a TCP connection takes a finite set of values between 0 and the total bandwidth μ. The transmission rate of a TCP connection at the next congestion event predicted by (2) is then rounded to the closest value in this set of values. We get then a discrete-time disrecte-space Markov chain for which we can compute numerically the stationary distribution by writing the system of balance equations. We did different runs for many scenarios and we always found that this stationary distribution exists and is unique. From now on, we put ourselves in the stationary regime, and we remove the time index n from all random variables and processes, e.g., $X_{i,n}$ becomes X_i.

3. Calculation of the throughput

The throughput of a connection say i, or equivalently the time average of its transmission rate, is equal to

$$\overline{X}_i = \lim_{t\to\infty} \frac{1}{t}\int_0^t X_i(u)\,\mathrm{d}u = \lim_{n\to\infty} \frac{\sum_{m=0}^{n-1}\int_{t_m}^{t_{m+1}} X_i(u)\,\mathrm{d}u}{\sum_{m=0}^{n-1} D_m}$$
$$= \lim_{n\to\infty} \frac{(1/n)\sum_{m=0}^{n-1}(X_{i,m} + U_{i,m}(R_i(X_{i,m}) - X_{i,m}))D_m + \alpha_i(D_m)^2/2}{(1/n)\sum_{m=0}^{n-1} D_m}.$$

Given that the system has a unique stationary regime, the limit exists and is equal to

$$\overline{X}_i = \frac{\mathbb{E}[(X_i + U_i(R_i(X_i) - X_i))D + \alpha_i(D)^2/2]}{\mathbb{E}[D]}. \tag{4}$$

Let $D_j(X)$ denote the time until the next congestion event when the system is in state $X \in \mathcal{X}$ and source j reduces its rate. Using (3), we have

$$D_j(X) = \frac{x_j - R_j(x_j)}{\sum_{m=1}^{N}\alpha_m}.$$

Thus,

$$\overline{X}_i = \frac{\int \mathrm{d}\pi_X\left(\sum_{j=1}^{N} p_j(X)(x_i D_j(X) + \alpha_i(D_j(X))^2/2) + p_i(X)(R_i(x_i) - x_i)D_i(X)\right)}{\int \mathrm{d}\pi_X \sum_{j=1}^{N} p_j(X)D_j(X)}. \tag{5}$$

This equation shows that the throughput of a connection is a function of many parameters. First, it is a function of some constant parameters as μ, N, and α_i. Second, it is a function of two other parameters that can be changed to improve the service provided to applications using TCP. These latter two parameters are the probability function $p_i(X)$ and the amount by which the rate of a connection is reduced when a congestion signal is received $R_i(x_i)$. The schemes we compare in this paper set differently these two parameters.

4. Application of the model to real schemes

Suppose that the system is in state $X = (x_1, \ldots, x_N) \in \mathcal{X}$ in the stationary regime. We find in this section the expressions of the two functions $p_i(X)$ and $R_i(x_i)$ for the different DiffServ schemes we are considering in this paper.

4.1. Standard TCP with RIO

A source i asks the network for bandwidth μ_i. Packets below the reserved bandwidth are marked as IN and those exceeding the reserved bandwidth are marked as OUT. When a congestion appears at the bottleneck, the router starts to drop OUT packets with a

certain probability. Connections transmitting at less than their reserved bandwidths are protected. The probability that a connection transmitting OUT packets reduces its rate is a function of (i) the probability at which the network drops OUT packets, (ii) the rate of its OUT packets, and (iii) the total rate of OUT packets crossing the bottleneck at the moment of congestion. If there is no OUT packets in the network (all the connections are transmitting at less than their reservations), congestion remains and the router starts to drop IN packets. When an OUT or IN packet is dropped, the corresponding connection divides its rate by two. Thus, $R_i(x_i) = x_i/2$ in this case and in all the subsequent cases where standard TCP is used.

The probability that a connection reduces its rate upon a congestion is equal to 0 when it is transmitting only IN packets and there is at least one connection transmitting OUT packets. It is equal to 1 if it is the sole connection transmitting OUT packets. Next we study the case when the connection is transmitting OUT packets together with other connections. The last case, that of all the connections transmitting only IN packets, will be directly deduced.

Upon a congestion, a RIO buffer treats OUT packets from all connections in the same way. Let q be the probability that an OUT packet is dropped at the bottleneck and let V be the result of the probabilistic drop applied to a packet. It is equal to 1 if the packet is really dropped and to zero otherwise. Denote by $Y = 1, \ldots, N$ the number of the connection to which the dropped OUT packet belongs. In the following we denote by $P_X(A)$ the probability that event A happens given that the system is in state $X \in \mathcal{X}$ upon congestion. We have

$$p_i(X) = P_X(Y = i \mid V = 1) = \frac{P_X(Y = i \text{ and } V = 1)}{P_X(V = 1)}$$
$$= \frac{P_X(Y = i) \cdot P_X(V = 1 \mid Y = i)}{\sum_{m=1}^{N} P_X(Y = m) \cdot P_X(V = 1 \mid Y = m)}$$

For $m = 1, \ldots, N$, $P_X(V = 1 \mid Y = m)$ is no other than q. Thus, $p_i(X)$ is equal to $P_X(Y = i)$ which is the probability that an OUT packet belongs to connection i given that the system is in state X. This probability is equal to the ratio of the rate at which connection i is sending OUT packets and the total rate of OUT packets. Thus,

$$p_i(X) = P_X(Y = i) = \frac{x_i - \mu_i}{\sum_{m=1}^{N} (x_m - \mu_m) \mathbb{1}\{x_m > \mu_m\}},$$

where $\mathbb{1}\{\}$ is the indicator function.

Similarly, we can calculate the probability that connection i reduces its rate when all connections are transmitting only IN packets. Again, all packets are treated in the same way by the RIO buffer. $p_i(X)$ is equal in this case to the probability that an IN packet belongs to connection i, which is equal to the ratio of the rate at which connection

i is sending IN packets (x_i) and the total rate at which IN packets are sent (μ). We can then write the general expression of $p_i(X)$. For any $X \in \mathcal{X}$ we have

$$p_i(X) = \begin{cases} \dfrac{x_i}{\mu} & \text{if } \displaystyle\sum_{m=1}^{N} \mathbb{1}\{x_m > \mu_m\} = 0, \\ \dfrac{(x_i - \mu_i)\mathbb{1}\{x_i > \mu_i\}}{\sum_{m=1}^{N}(x_m - \mu_m)\mathbb{1}\{x_m > \mu_m\}} & \text{otherwise.} \end{cases}$$

4.2. Modified TCP with RIO

Packets are marked with two priority levels and RIO buffers are used in network routers. Thus, $p_i(X)$ is the same as in the previous section. The difference is in the function $R_i(x_i)$. If an IN packet is lost, the source divides its rate by two as with standard TCP. If the dropped packet is an OUT packet, the proposed scheme [Feng et al., 7; Yeom and Reddy, 21] consists in only dividing the rate of OUT packets by two. We consider in our model that the dropped packet from connection i is an IN packet if at the moment of congestion source i is transmitting at less than its reservation, otherwise it is an OUT packet. Thus, the transmission rate of connection i just after it reduces its rate is equal to

$$R_i(x_i) = \begin{cases} \dfrac{x_i}{2} & \text{if } x_i < \mu_i, \\ \mu_i + \dfrac{x_i - \mu_i}{2} & \text{otherwise.} \end{cases}$$

4.3. Inverse drop probability scheme

Standard TCP with two priority levels is used, therefore $R_i(x_i)$ is equal to one half x_i. The difference in this case is that packets of different connections (IN and OUT) are not treated in the same way in the core of the network. The idea proposed in [Yeom and Reddy, 21] is to drop OUT packets from a connection with a probability that varies as the inverse of its reservation. However, the drop probability of IN packets is not specified. IN packets are actually dropped when all connections are transmitting at less than their reservations. In this case and according to our objectives (section 1), the throughput of a connection must be proportional to its reservation. It is known that the throughput of a TCP connection varies as the square root of the packet drop probability [Altman et al., 2; Barakat, 3; Lakshman and Madhow, 17; Padhye et al., 19]. Thus, in order to achieve the above objective, we propose to drop IN packets with a probability that varies as the inverse of the square of the reservation. We add this new feature to the proposed scheme.

As in the case of RIO with standard TCP, the router tries first to drop OUT packets. If these packets do not exist, IN packets will be dropped. Again, a connection reduces its rate with probability 1 if its the sole connection exceeding its reservation, and with probability 0 if it is transmitting only IN packets and there is at least one other connection transmitting OUT packets. For the remaining two cases, we consider first the case when

the connection is transmitting OUT packets together with other connections. The other case will be directly deduced.

Suppose that the bottleneck router drops OUT packets of source $m = 1, \ldots, N$ with a probability q/μ_m, q is a constant. Suppose also that the system is in state $X \in \mathcal{X}$ when the congestion occurs. Then,

$$p_i(X) = \frac{P_X(Y = i) \cdot P_X(V = 1 \mid Y = i)}{\sum_{m=1}^{N} P_X(Y = m) \cdot P_X(V = 1 \mid Y = m)} = \frac{P_X(Y = i)/\mu_i}{\sum_{m=1}^{N} P_X(Y = m)/\mu_m}.$$

As in the case of RIO, $P_X(Y = m)$ is equal to

$$P(Y = m) = \frac{x_m - \mu_m}{\sum_{j=1}^{N}(x_j - \mu_j)\mathbb{1}\{x_j > \mu_j\}}.$$

Thus,

$$p_i(X) = \frac{x_i/\mu_i - 1}{\sum_{m=1}^{N}(x_m/\mu_m - 1)\mathbb{1}\{x_m > \mu_m\}}.$$

When all connections are only transmitting IN packets, the problem is similar. The difference is in the drop probability that we propose to take inversely proportional to the square of the reservation. The general expression of $p_i(X)$ for this scheme is then,

$$p_i(X) = \begin{cases} \dfrac{x_i/\mu_i^2}{\sum_{m=1}^{N} x_m/\mu_m^2} & \text{if } \displaystyle\sum_{m=1}^{N} \mathbb{1}\{x_m > \mu_m\} = 0, \\ \dfrac{(x_i/\mu_i - 1)\mathbb{1}\{\mu_i > x_i\}}{\sum_{m=1}^{N}(x_m/\mu_m - 1)\mathbb{1}\{x_m > \mu_m\}} & \text{otherwise.} \end{cases}$$

4.4. Saw tooth marking scheme

Standard TCP, two priority levels and RIO buffers are used, thus $R_i(x_i) = x_i/2$. The difference here is in the marker operation. The flow of connection i contains OUT packets when its rate exceeds $4\mu_i/3$. The rate of its OUT packets at the moment of congestion is equal to $x_i - 4\mu_i/3$ rather than $x_i - \mu_i$. The new expression of the probability function $p_i(X)$ is then

$$p_i(X) = \begin{cases} \dfrac{x_i}{\mu} & \text{if } \displaystyle\sum_{m=1}^{N} \mathbb{1}\left\{x_m > \frac{4\mu_m}{3}\right\} = 0, \\ \dfrac{(x_i - 4\mu_i/3)\mathbb{1}\{x_i > 4\mu_i/3\}}{\sum_{m=1}^{N}(x_m - 4\mu_m/3)\mathbb{1}\{x_m > 4\mu_m/3\}} & \text{otherwise.} \end{cases}$$

4.5. Standard TCP with three drop priorities

In this scheme, the source makes two reservations instead of one. Denote these reservations by $\mu_{i,1}$ and $\mu_{i,2}$ with $\mu_{i,1} < \mu_{i,2}$. Standard TCP is used at the source, therefore

$R_i(x_i) = x_i/2$. Packets are marked with three priority levels or three colors. Packets exceeding $\mu_{i,2}$ are marked with low priority (red color). Those exceeding $\mu_{i,1}$ but not $\mu_{i,2}$ are marked with medium priority (yellow color). Packets sent at a rate slower than $\mu_{i,1}$ are marked with a high priority (green color).

As in the RIO case, the network starts first to drop low-priority packets. This happens when one of the sources, say i, is exceeding its upper reservation $\mu_{i,2}$. If those packets do not exist, medium-priority packets are dropped. Medium-priority packets exist in the network when one of the sources, say i, is exceeding its lower reservation $\mu_{i,1}$. If it is not the case, the network drops high-priority packets.

All packets belonging to a certain priority level are treated in the network in the same manner. A connection reduces its rate upon congestion with probability 1 if it is transmitting alone above a certain level. It reduces its rate with probability 0 if it is transmitting below a level and there is another connection transmitting above the same level. In the other cases, the probability that a connection reduces its rate is equal to the probability that the dropped packet belongs to this connection. Similarly to the RIO case we can write,

$$p_i(X) = \begin{cases} \dfrac{x_i}{\mu} & \text{if } \displaystyle\sum_{m=1}^{N} \mathbb{1}\{x_m > \mu_{m,1}\} = 0, \\ \dfrac{(x_i - \mu_{i,1})\mathbb{1}\{x_i > \mu_{i,1}\}}{\sum_{m=1}^{N}(x_m - \mu_{m,1})\mathbb{1}\{x_m > \mu_{m,1}\}} & \text{if } \displaystyle\sum_{m=1}^{N} \mathbb{1}\{x_m > \mu_{m,1}\} > 0 \\ & \text{and } \displaystyle\sum_{m=1}^{N} \mathbb{1}\{x_m > \mu_{m,2}\} = 0, \\ \dfrac{(x_i - \mu_{i,2})\mathbb{1}\{x_i > \mu_{i,2}\}}{\sum_{m=1}^{N}(x_m - \mu_{m,2})\mathbb{1}\{x_m > \mu_{m,2}\}} & \text{otherwise.} \end{cases}$$

To compare this scheme to previous ones, the two reservations $\mu_{i,1}$ and $\mu_{i,2}$ must be set as a function of the desired throughput μ_i. If we look at the saw tooth variation of TCP rate in figure 1, we see that on average and in order to realize a throughput μ_i, the connection rate should vary between $2\mu_i/3$ and $4\mu_i/3$. Based on that, we give packets below $2\mu_i/3$ the highest priority, packets between $2\mu_i/3$ and $4\mu_i/3$ the medium priority, and packets above $4\mu_i/3$ the lowest priority. This corresponds to $\mu_{i,1} = 2\mu_i/3$ and $\mu_{i,2} = 4\mu_i/3$. The three drop priorities scheme is compared later to the other schemes with these particular values of the two reservations. Other values can be always used.

5. Case of a large number of connections

We present in this section closed-form expressions for the throughput when a large number of TCP connections share the bottleneck. We look for closed-form expressions of the throughput that are independent of the parameters of the other connections, but rather dependent on some parameters describing the state of the network (e.g., loss process, band-

width, subscription level). This is similar to the approach used in [Yeom and Reddy, 22] where the state of network is represented by the packet drop probability and the subscription level. Later, we will describe how both approaches can be related together to obtain the same results.

We focus on a particular connection i. We assume that the process of times between congestion events $\{D_n\}$ is independent of the rate of connection i. We further assume that these times are identically and exponentially distributed with intensity λ and with average $d = 1/\lambda$. This very probably holds given the large number of connections that are multiplexed at the bottleneck. We suppose that the process of congestion events is known. Consider for the moment that we are able to get the intensity of $\{D_n\}$ by probing directly the bottleneck router. Later we explain how it can be calculated on end-to-end basis by using the probability that a packet is dropped or the percentage of dropped packets. Given the assumption that the process of congestion events is Poisson, we can write (4) as follows:

$$\overline{X}_i = \mathbb{E}[X_i] + \mathbb{E}\big[U_i(R_i(X_i) - X_i)\big] + \alpha_i d.$$

Using the PASTA (Poisson Arrivals See Time Averages) theory [Kleinrock, 16, chapter 5], the expectation of the rate of connection i at moments of congestion ($\mathbb{E}[X_i]$) is equal to the time-average of the rate of the connection or its throughput ($\overline{X}_i$). Thus, using the above equation, the throughput of connection i can be written as the solution of:

$$\mathbb{E}\big[U_i\big(X_i - R_i(X_i)\big)\big] = \bar{p}_i(\overline{X}_i)\mathbb{E}\big[\big(X_i - R_i(X_i)\big) \mid U_i = 1\big] = \alpha_i d. \tag{6}$$

$\bar{p}_i(\overline{X}_i)$ is the probability that connection i reduces its rate at a congestion event. It is equal to the expectation of the conditional probability $p_i(X) = P_X(U_i = 1)$. Recall that $P_X(A)$ denotes the probability that event A holds given that the N TCP connections are in state $X = (x_1, \ldots, x_N)$.

Next, we will express $\bar{p}_i(\overline{X}_i)$ as only a function of the throughput of connection i ($\overline{X}_i$). The impact of the rates of the other connections will be eliminated by using the bottleneck bandwidth μ and the subscription level. Consider for example the RIO case and denote by ρ the subscription level or the ratio of the total bandwidth reserved and the total available bandwidth. $\rho < 1$ means under-subscription and $\rho > 1$ means over-subscription. Given the large number of connections, the total rate of OUT packets at the moments of congestion can be approximated by $(1 - \rho)\mu$ when $\rho < 1$ and by 0 otherwise. The total rate of IN packets can be approximated by $\rho\mu$ when $\rho < 1$ and by μ otherwise. Thus, there is no need for the rates of the other connections in the calculation of $p_i(X)$. We can express this probability as a function of X_i, μ and ρ. We will explain later how this calculation can be done for each scheme. Clearly, this will not work with the Inverse Drop Probability scheme where we need the bandwidth reserved by each connection rather than the total reservation.

Further simplification is required to solve the nonlinear system (6) for $\overline{X}_i$. The nonlinearity comes from the dependency between U_i and X_i. We propose two possible

approximations to solve this dependency. The first approximation is to suppose that these random variables are independent. From (6) we get

$$\bar{p}_i(\overline{X}_i)(\overline{X}_i - \overline{R}_i(\overline{X}_i)) = \alpha_i d.$$

$\overline{R}_i(\overline{X}_i)$ is the expectation of $R_i(X_i)$, hence it is equal to $\overline{X}_i/2$ in case of standard TCP. This approximation is equivalent to assuming that connection i reduces its rate at congestion moments with a constant probability $\bar{p}_i(\overline{X}_i)$. We already studied such approximation in [Altman et al., 1] where we assumed that the rate of the TCP connection decreases at some potential loss moments with a probability independent of its transmission rate. The result of this approximation is an exponentially distributed time between moments at which connection i reduces its rate (at which U_i is equal to 1) with a mean equal to $d/\bar{p}_i(\overline{X}_i)$.

The second approximation consists in considering a fixed-point approach as the one used in the literature [Lakshman and Madhow, 17; Padhye et al., 19; Yeom and Reddy, 22]. This approach consists in taking X_i constant when $U_i = 1$, i.e. when the rate of connection i is reduced. Denote this constant by X_0. The rate of connection i changes then in the stationary regime between $R_i(X_0)$ and X_0 (figure 1) with a time average $\overline{X}_i$. In our case where the probability $p_i(X)$ increases with the transmission rate of the connection, this should give better result than the previous approximation. In the case of standard TCP, $R_i(X_0)$ is equal to $X_0/2$ and thus,

$$X_0 = \frac{4\overline{X}_i}{3}. \tag{7}$$

This is also the value of X_0 in the case of modified TCP when $\overline{X}_i < \mu_i$. Now, in the particular case of modified TCP and $\overline{X}_i > \mu_i$, $R_i(X_0)$ is equal to $\mu_i + (X_0 - \mu_i)/2$ and thus,

$$X_0 = \mu_i + \frac{4(\overline{X}_i - \mu_i)}{3}. \tag{8}$$

Using (6) we can write for the case of the second approximation,

$$\mathbb{E}\big[U_i\big(X_i - R_i(X_i)\big)\big] = \bar{p}_i(\overline{X}_i)\big(X_0 - R_i(X_0)\big) = \alpha_i d. \tag{9}$$

We chose to work with this second approximation. Using (9) where X_0 and $R_i(X_0)$ can be expressed as a function of $\overline{X}_i$, we can compute explicitly the throughput of connection i. Again, by appropriately specifying the two functions $\bar{p}_i(\overline{X}_i)$ and $R_i(X_0)$, we can cover all the proposed schemes. $R_i(X_0)$ is given in section 4 for the different schemes as a function of X_0. We still have to calculate $\bar{p}_i(\overline{X}_i)$. We exclude from our calculation the inverse drop probability scheme since it requires the knowledge of the bandwidth reserved by the different connections not only the total reservation. If we assume that all the connections have approximately the same reservation, this scheme will be identical to Standard TCP with RIO.

5.1. Standard TCP with RIO

Using (7) and (9) we write,

$$\bar{p}_i(\overline{X}_i)\overline{X}_i = \frac{3\alpha_i d}{2} = \frac{3\alpha_i}{2\lambda}. \tag{10}$$

Consider first the case $\rho < 1$. The total rate of OUT packets crossing the bottleneck at the moments of congestion is equal to $(1-\rho)\mu$. Thus, we write $\bar{p}_i(\overline{X}_i)$ as follows:

$$\bar{p}_i(\overline{X}_i) = \mathbb{E}\big[p_i(X)\big] = \frac{1}{(1-\rho)\mu}\mathbb{E}\big[(X_i-\mu_i)\mathbb{1}\{X_i > \mu_i\}\big]. \tag{11}$$

The term $\mathbb{E}[(X_i-\mu_i)\mathbb{1}\{X_i > \mu_i\}]$ is equal to the average area between the rate of connection i, its reservation μ_i and two reductions of its rate (i.e., the dark areas in figure 1) divided by the average duration of a TCP cycle (i.e., average time between two consecutive reductions of the rate of the connection). We use here the PASTA theory to equate moments upon congestion events to moments at arbitrary time moments. Note that a TCP cycle can include many congestion events. To compute the above expectation, we reconsider the fixed-point approach where the rate of connection i oscillates in the stationary regime between $X_0/2$ and X_0, with $X_0 = 4\overline{X}_i/3$. It is clear that X_0 is greater than the reserved bandwidth μ_i since the network is not over-subscribed and some bandwidth exists for low-priority packets. Hence,

$$\bar{p}_i(\overline{X}_i) = \begin{cases} \dfrac{(X_0-\mu_i)^2}{X_0(1-\rho)\mu} = \dfrac{(4\overline{X}_i - 3\mu_i)^2}{12\overline{X}_i(1-\rho)\mu} & \text{if } \dfrac{X_0}{2} = \dfrac{2\overline{X}_i}{3} < \mu_i, \\ \dfrac{\overline{X}_i - \mu_i}{(1-\rho)\mu} & \text{otherwise.} \end{cases}$$

Solving equation (10) for $\overline{X}_i$, we get

$$\overline{X}_i = \begin{cases} \dfrac{\mu_i}{2} + \sqrt{\dfrac{\mu_i^2}{4} + \dfrac{3\alpha_i(1-\rho)\mu}{2\lambda}} & \text{if } \mu_i < \sqrt{\dfrac{2\alpha_i(1-\rho)\mu}{\lambda}}, \\ \dfrac{3\mu_i}{4} + \sqrt{\dfrac{9\alpha_i(1-\rho)\mu}{8\lambda}} & \text{otherwise.} \end{cases} \tag{12}$$

We still have to consider the case $\rho > 1$. X_i is always smaller than the reserved bandwidth since there is no place left for OUT packets. The probability that connection i reduces its rate upon congestion is simply

$$\bar{p}_i(\overline{X}_i) = \mathbb{E}\left[\frac{X_i}{\mu}\right] = \frac{\overline{X}_i}{\mu}.$$

This gives the following expression for the throughput:

$$\overline{X}_i = \min\left(\sqrt{\frac{3\alpha_i\mu}{2\lambda}}, \frac{3\mu_i}{4}\right). \tag{13}$$

The minimum operator is used to make sure that the throughput in case $\rho > 1$ cannot exceed $3\mu_i/4$ (no OUT packets are transmitted).

5.2. *Modified TCP with RIO*

The analysis for the over-subscription case ($\rho > 1$) is similar to that in the previous section. We get the same expression for the throughput as (13). In the under-subscription case and using (8) and (9), we write

$$\bar{p}_i(\overline{X}_i)(\overline{X}_i - \mu_i) = \frac{3\alpha_i}{2\lambda}.$$

$\bar{p}_i(\overline{X}_i)$ is given by (11). The transmission rate of connection i is always above the reservation which gives

$$\bar{p}_i(\overline{X}_i) = \frac{\overline{X}_i - \mu_i}{(1-\rho)\mu}.$$

Solving for the throughput $\overline{X}_i$, we get

$$\overline{X}_i = \mu_i + \sqrt{\frac{3\alpha_i(1-\rho)\mu}{2\lambda}}.$$

5.3. *Saw tooth marking*

The calculation here is similar to that in the case of RIO with Standard TCP with a small difference in the definition of ρ. Given that the marker at the edge of the network starts to mark packets as OUT when the rate of connection i exceeds $4\mu_i/3$ rather than μ_i, we define ρ as $(\sum_{i=1}^{N} 4\mu_i/3)/\mu$. The network contains $4/3$ more IN packets than in the RIO case. The under-subscription case corresponds always to $\rho < 1$ and the over-subscription case to $\rho > 1$.

Consider the case $\rho < 1$. Using (7) and (9), we have

$$\bar{p}_i(\overline{X}_i)\overline{X}_i = \frac{3\alpha_i}{2\lambda},$$
$$\bar{p}_i(\overline{X}_i) = \frac{1}{(1-\rho)\mu}\mathbb{E}\left[\left(X_i - \frac{4\mu_i}{3}\right)\mathbb{1}\left\{X_i > \frac{4\mu_i}{3}\right\}\right].$$

The throughput can be simply obtained by substituting μ_i by $4\mu_i/3$ in (12). It follows for $\rho < 1$,

$$\overline{X}_i = \begin{cases} \dfrac{2\mu_i}{3} + \sqrt{\dfrac{4\mu_i^2}{9} + \dfrac{3\alpha_i(1-\rho)\mu}{2\lambda}} & \text{if } \mu_i < \sqrt{\dfrac{9\alpha_i(1-\rho)\mu}{8\lambda}}, \\ \mu_i + \sqrt{\dfrac{9\alpha_i(1-\rho)\mu}{8\lambda}} & \text{otherwise.} \end{cases}$$

When $\rho > 1$, the throughput can be calculated similarly to the case of RIO with Standard TCP. The difference is that in this case the throughput of connection i can go until μ_i,

$$\overline{X}_i = \min\left(\sqrt{\frac{3\alpha_i\mu}{2\lambda}}, \mu_i\right).$$

5.4. *Standard TCP with three drop priorities*

The analysis is similar to that in the RIO case. Define ρ_1 and ρ_2 as the two ratios indicating how the bandwidth μ is allocated to different priority levels ($\rho_k = \sum_{i=1}^{N} \mu_{i,k}/\mu$). We have $\rho_1 < \rho_2$. We distinguish three regions instead of two. The first region corresponds to $\rho_1 < \rho_2 < 1$. The expression of the throughput in this case is obtained by substituting μ_i by $\mu_{i,2}$ and ρ by ρ_2 in (12). The second region corresponds to $\rho_1 < 1 < \rho_2$ and the throughput is obtained by substituting μ_i by $\mu_{i,1}$ and ρ by ρ_1 in (12). The third region corresponds to $1 < \rho_1 < \rho_2$ and the throughput is given by (13) after the substitution of μ_i by $\mu_{i,1}$.

In order to compare this scheme to previous ones, we need to set $\mu_{i,1}$ and $\mu_{i,2}$ as a function of the throughput desired by the connection (μ_i). As in section 4, we set $\mu_{i,1}$ to $2\mu_i/3$ and $\mu_{i,2}$ to $4\mu_i/3$. Similarly, ρ_1 is equal to $2\rho/3$ and ρ_2 to $4\rho/3$. We have then the following expression for the throughput,

For $\rho < \dfrac{3}{4}$,

$$\overline{X}_i = \begin{cases} \dfrac{2\mu_i}{3} + \sqrt{\dfrac{4\mu_i^2}{9} + \dfrac{3\alpha_i(1-4\rho/3)\mu}{2\lambda}} & \text{if } \mu_i < \sqrt{\dfrac{9\alpha_i(1-4\rho/3)\mu}{8\lambda}}, \\ \mu_i + \sqrt{\dfrac{9\alpha_i(1-4\rho/3)\mu}{8\lambda}} & \text{otherwise.} \end{cases}$$

For $\dfrac{3}{4} < \rho < \dfrac{3}{2}$,

$$\overline{X}_i = \begin{cases} \dfrac{\mu_i}{3} + \sqrt{\dfrac{\mu_i^2}{9} + \dfrac{3\alpha_i(1-2\rho/3)\mu}{2\lambda}} & \text{if } \mu_i < \sqrt{\dfrac{9\alpha_i(1-2\rho/3)\mu}{2\lambda}}, \\ \dfrac{\mu_i}{2} + \sqrt{\dfrac{9\alpha_i(1-2\rho/3)\mu}{8\lambda}} & \text{otherwise.} \end{cases}$$

For $\dfrac{3}{2} < \rho$, $\quad \overline{X}_i = \min\left(\sqrt{\dfrac{3\alpha_i\mu}{2\lambda}}, \dfrac{\mu_i}{2}\right)$.

5.5. *Relation with the probability approach*

In the literature [Lakshman and Madhow, 17; Padhye et al., 19; Yeom and Reddy, 22], the probability with which packets of different priority levels are dropped in the network is used instead of the intensity of congestion events (λ). The advantage of the former approach is that the drop probability is a measure that can be computed end-to-end. This can be done by simply counting the number of drops to the total number of packets

transmitted. The relation between the two approaches is however quite simple, and by some transformation we can prove that our results are similar to those found in [Yeom and Reddy, 22]. Of course, our study remains more general than [Yeom and Reddy, 22] and since it accounts for more DiffServ schemes. Consider, for example, the scheme of standard TCP with RIO buffers which is also considered in [Yeom and Reddy, 22]. Let p_{IN} and p_{OUT} be respectively the probabilities that IN and OUT packets are dropped in the network. Assume that the bottleneck link is fully utilized. The intensity of losses is no other than the drop probability times the rate of packets. Thus, in the under-subscription case we have

$$p_{\mathrm{IN}} = 0, \qquad \lambda = p_{\mathrm{OUT}}(1-\rho)\mu.$$

And in the over-subscription case,

$$\lambda = p_{\mathrm{IN}}\mu, \qquad p_{\mathrm{OUT}} = 1.$$

Substituting in (12) and (13), λ and ρ by their values as a function of p_{IN} and p_{OUT}, one can find the same expressions for the throughput as those found in [Yeom and Reddy, 22] for the standard TCP with RIO scheme.

6. Some numerical results

We solve numerically our model for the case of two concurrent TCP connections. This gives a Markov chain of dimension 1. The performance of the different schemes is compared according to the objectives we presented at the beginning of section 1. The two connections share a bottleneck of bandwidth $\mu = 1.5$ Mbps. TCP packets are of total size 552 packets (MSS + TCP/IP header). Reservations are expressed in kbps. The receivers are supposed to acknowledge every data packet which gives $\alpha_i = 1/T_i^2$ packets/s^2. Recall that α_i is the rate at which the transmission rate of connection i increases versus time. By using this value for α_i, we obtain the throughput in packets/s. First, we give the two connections the same RTT (100 ms) and we study the performance of the different schemes under different reservations and different subscription levels. Second, we study the impact of the difference in RTT on the service differentiation provided by the different schemes.

6.1. Impact of the reservation

We change the reservations of the two sources in a way that their sum is constant and equal to $\rho\mu$; ρ indicates how much bandwidth is reserved by the two connections. We consider three values of ρ: 0.5, 1 and 1.5. For each ρ and according to the objectives in section 1, we define a factor F that characterizes how much connection 1 is favored with respect to connection 2. For $\rho < 1$, the network is under-subscribed and the two sources must share fairly the excess bandwidth. We define F in this case as the ratio of $\overline{X}_1 - \mu_1$ and $\overline{X}_2 - \mu_2$. The optimal scheme is the one that gives the closest F to 1 [Yeom and Reddy, 21]. An $F > 1$ means that the scheme is in favor of connection 1.

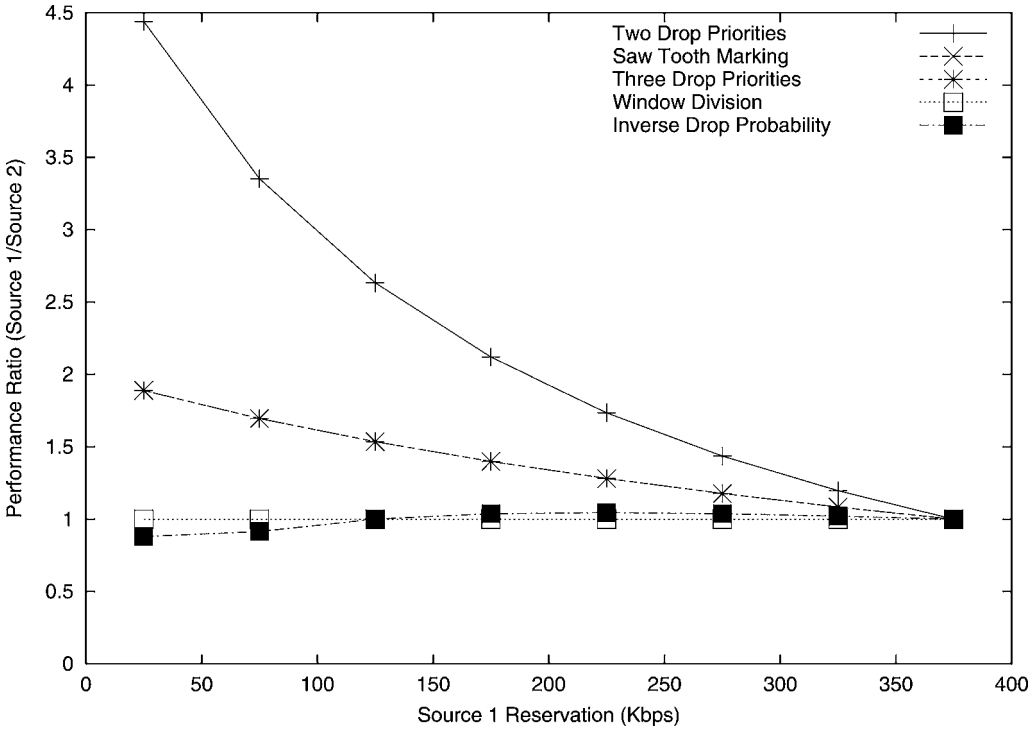

Figure 2. Performance comparison for a 50% total reservation.

For $\rho \geqslant 1$, the network is over-subscribed. The bandwidth must be shared proportionally to the reservation. We define F in this case as the ratio of $\overline{X}_1/\mu_1$ and $\overline{X}_2/\mu_2$. Again, the optimal scheme is the one that gives the closest F to 1, and an $F > 1$ means that the scheme is in favor of connection 1.

In figures 2–4, we plot the factor F respectively for the three cases $\rho = 0.5$, 1 and 1.5. The X-axis shows the reservation of source 1 in Kbps. We vary this reservation between 0 and $\rho\mu/2$. For all the schemes and as one must predict, F converges to 1 when the reservation of source 1 moves to that of source 2.

In the under-subscription case, the original RIO scheme gives the worst service. The source with the small reservation achieves better performance than that with the large reservation. The other schemes improve the service. They give connection 2 more chance to increase its rate above its reservation which improves its throughput.

In the over-subscription case the situation changes. This is more depicted in figure 4. In this case, the original RIO scheme gives better performance than the proposed solutions (except the three colors scheme). The problem here is that the source with the large reservation is transmitting almost always IN packets and rarely OUT packets. Thus, it cannot profit from the high priority we give to OUT packets. The increase in the priority of OUT packets helps the source with the small reservation which achieves better throughput.

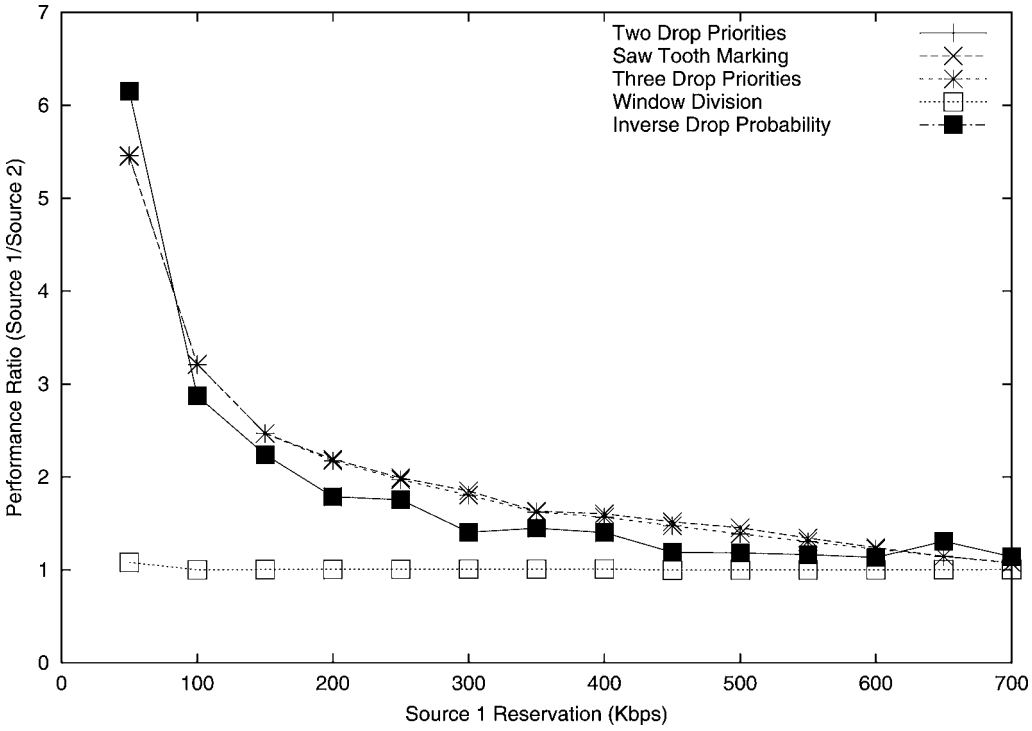

Figure 3. Performance comparison for a 100% total reservation.

The comparison between the different schemes requires also a calculation of the bottleneck bandwidth utilization $((\overline{X}_1+\overline{X}_2)/\mu)$. Two schemes with the same F will have different performances if they do not utilize equally the available bandwidth. In table 1 we give the average utilization for all schemes under different subscription levels. For a given level, we average the utilization over all the possible values of μ_1 and μ_2. The table shows that the different schemes give approximately the same utilization. The scheme proposing a change in TCP sources gives a better utilization in the cases of $\rho = 0.5$ and $\rho = 1$, but it gives also the best factor F in these two cases. The three figures showing F as a function of μ_1 and μ_2 for different ρ (figures 2–4) are then enough to compare the performances of the different schemes.

6.2. *Impact of the round-trip time*

The bias of TCP against connections with long RTT is known [Floyd and Jacobson, 10; Lakshman and Madhow, 17; Padhye et al., 19]. Long RTT connections take long time to recover from window reduction in contrast to small RTT connections which increase quickly their windows and grab most of the bandwidth. This is known to cause a problem of unfairness. In a best effort network, connections with different RTT crossing the same bottleneck fail to share equally the available bandwidth. In a DiffServ network, the increase in the RTT of a connection reduces also its throughput. This deteriorates

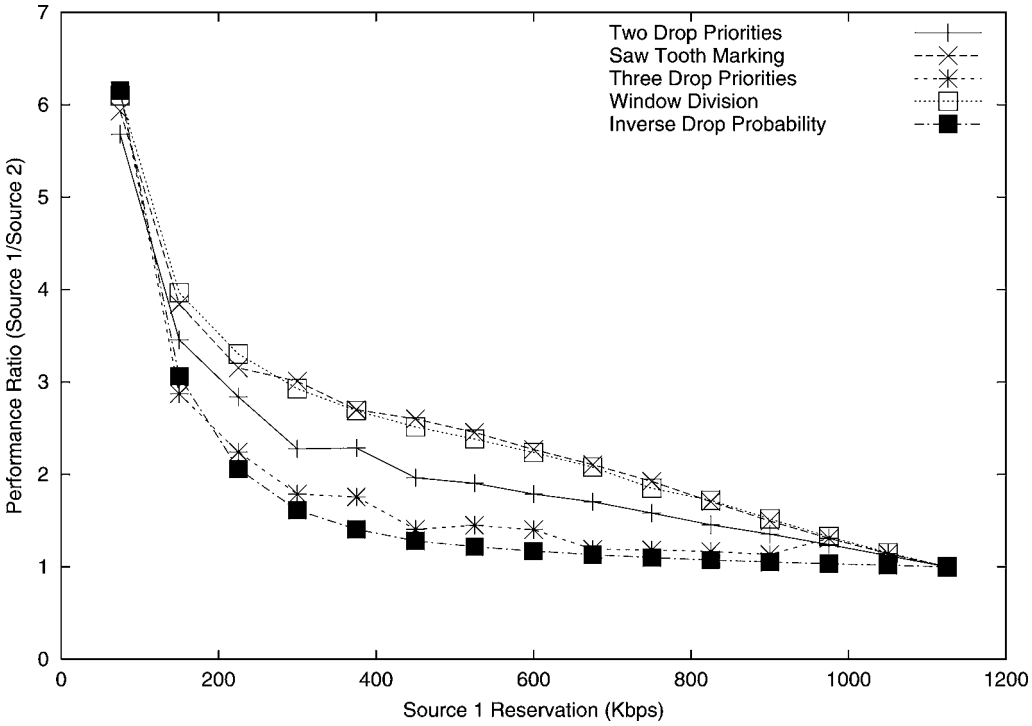

Figure 4. Performance comparison for a 150% total reservation.

Table 1
The utilization of the bottleneck bandwidth.

	Total reservation		
	50%	100%	150%
Window division	92.87%	88.57%	85.96%
Two drop priorities	85.72%	85.09%	85.68%
Three drop priorities	85.72%	85.68%	85.09%
Saw tooth marking	85.72%	85.70%	85.72%
Inverse drop probability	85.26%	85.09%	83.64%

the service if the throughput of the connection is already less than its fair share of the bandwidth given by the objectives in section 1. But, this improves the service if this connection whose RTT increases is using more than its fair share of the bandwidth.

We study in this section how much the difference in RTT impacts the service provided by the different DiffServ schemes. In other words, we study how much a scheme resists to a difference in RTT. We suppose that the two connections are asking for the same bandwidth ($\mu_1 = \mu_2$). We set T_2 to 50 ms and we vary T_1 between 50 ms and 500 ms. Ideally, the two connections must achieve the same throughput independently of their RTT. To quantify the impact of the change in T_1 on the service, we use the

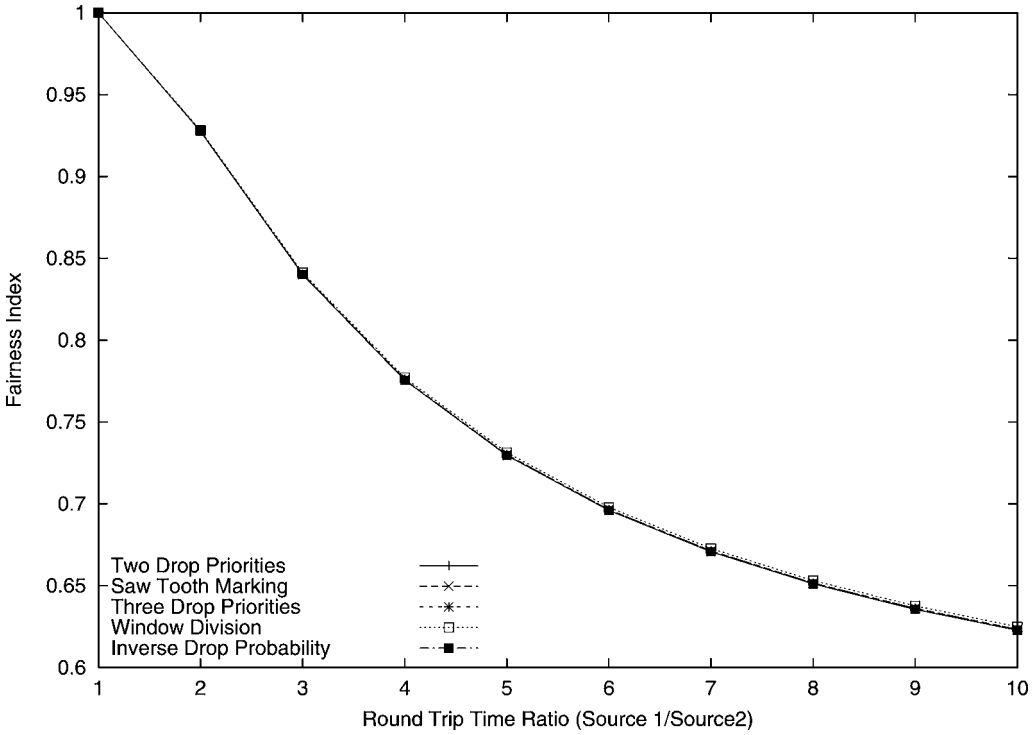

Figure 5. Fairness index for a 0% total reservation.

fairness index defined in [Jain et al., 15],

$$I = \frac{(\overline{X}_1 + \overline{X}_2)^2}{2((\overline{X}_1)^2 + (\overline{X}_2)^2)}.$$

This index is an increasing function of fairness. It varies between $1/2$ when one of the two connections is shut down and 1 when the two connections realize the same throughput. We plot in figures 5–8, the index I as a function of the ratio T_1/T_2 for four values of ρ: 0, 0.5, 1 and 1.5.

The zero reservation case corresponds to a best effort network. All the schemes achieve the same performance (figure 5). The fairness deteriorates as T_1 increases. The small RTT connection (i.e., 2) gets better and better performance. A small reservation as for $\rho = 0.5$ protects the long RTT connection and improves the service (figure 6). Indeed, as T_1 starts to increase, the throughput of connection 1 drops, but at a certain point it fells below its reservation and the connection starts here to send only high-priority packets. It becomes then protected from the other connection. This improves the fairness compared to the best effort network. All schemes other than RIO with standard TCP improve further the service. With these schemes, the long RTT connection has more chances to stay above its reservation.

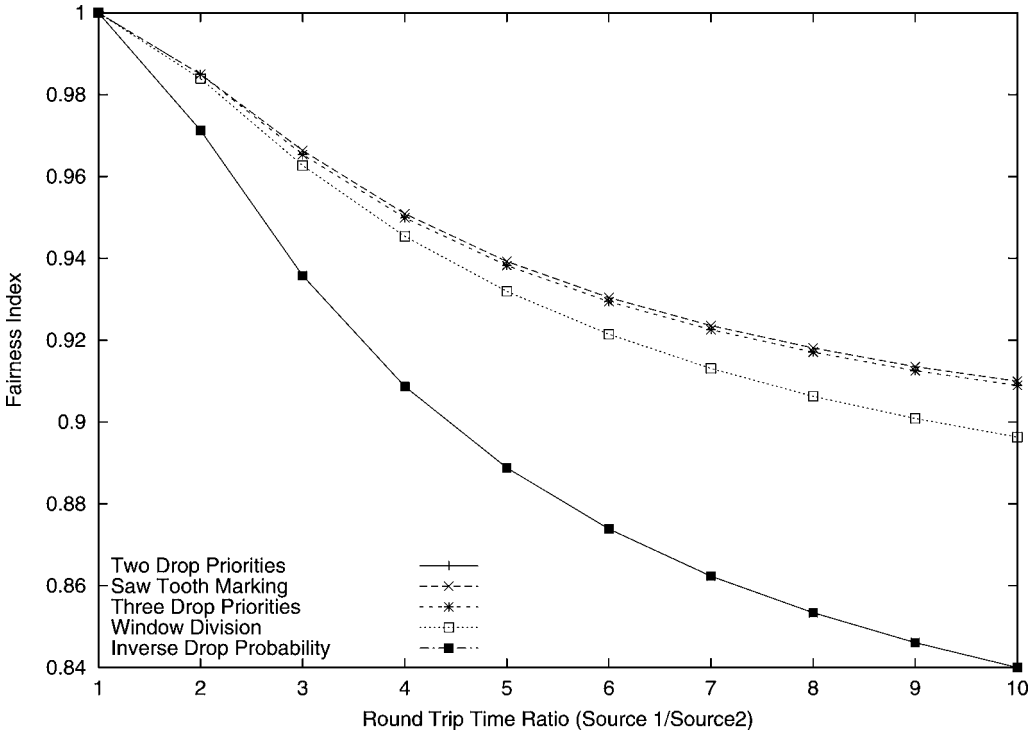

Figure 6. Fairness index for a 50% total reservation.

In the case of $\rho = 1$, the situation changes (figure 7). Connections are transmitting at approximately their reservation when $T_1 = T_2$. In this case, it is better not to help a connection to exceed its reservation because this will profit for the connection with small RTT instead of the connection with long RTT. Thus, RIO in this case gives better performance than the other schemes. We see approximately the same results in figure 8 where the total reservation is equal to 150% the available bandwidth. The connections in this case are transmitting at less then their reservations. The throughput of connection 1 deteriorates and that of connection 2 increases until the point where the throughput of connection 2 reaches its reservation. Connection 2 starts here to transmit low-priority packets. It is better here not to help this connection to increase its rate above its reservation. For this reason, the RIO scheme gives better performance than the others.

6.3. Discussion of the results

Our results show the problem of TCP in a DiffServ network which is reported in the literature [Basu and Wang, 4; Clark and Fang, 5; Feng et al., 7, 8; Yeom and Reddy, 21, 22]. A connection asking for a large bandwidth is unable to realize its fair share of the bandwidth. The different proposed solutions try to solve the problem by helping this connection to exceed its reservation more than a connection with a small reservation. In an under-subscribed network, this improves the performance. However, in an over-

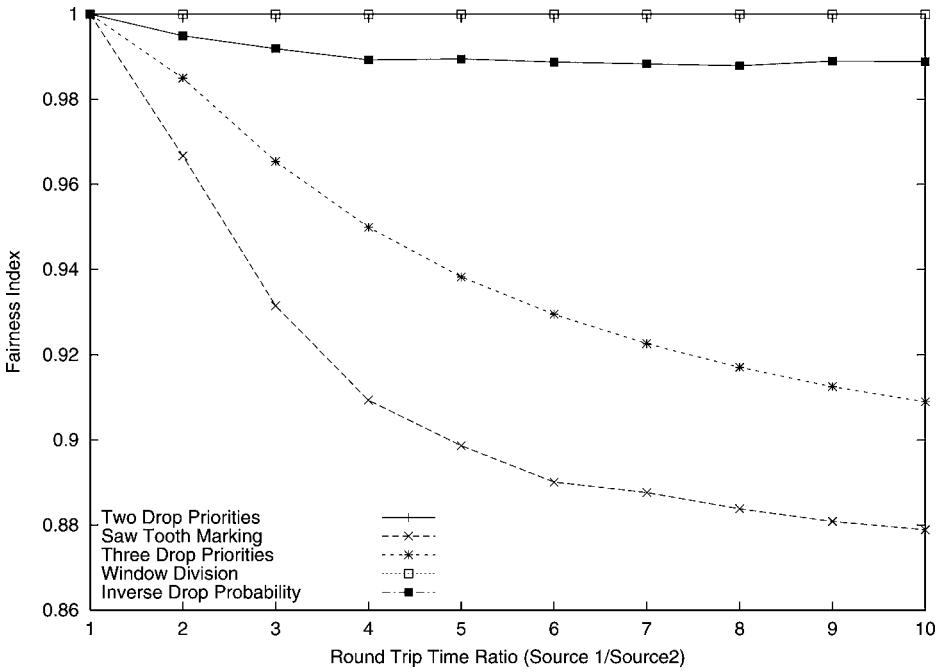

Figure 7. Fairness index for a 100% total reservation.

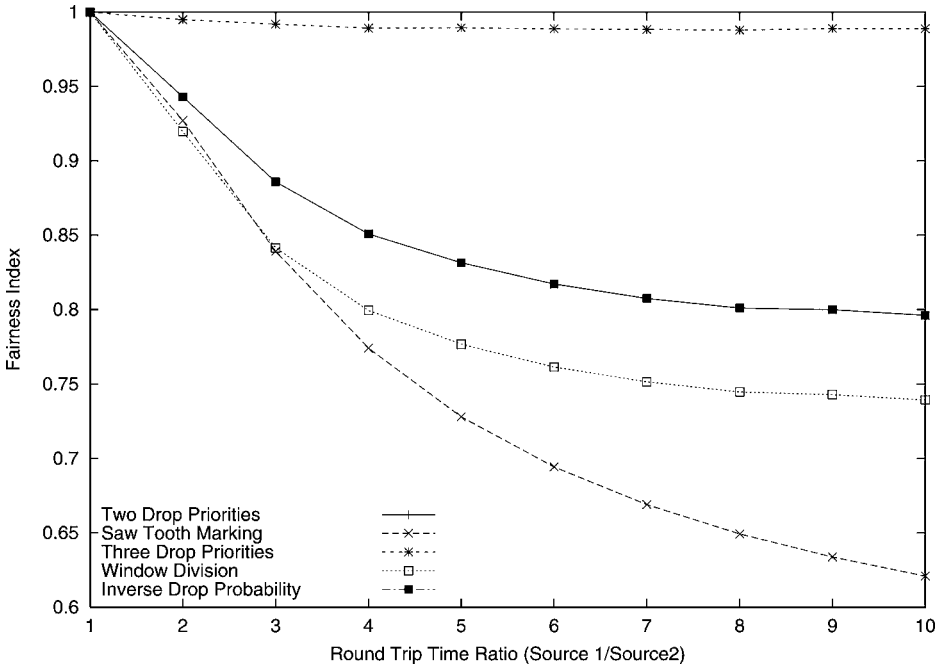

Figure 8. Fairness index for a 150% total reservation.

subscription network, the source with the small reservation profits from this help since it is the most likely to transmit above its reservation.

Our results also show that the difference in RTT deteriorates the service provided by the network. Some schemes resist better than others to this difference. Again here, the optimal scheme depends on the subscription level. In an under-subscription case, schemes helping the source to exceed its reservation give better performance. In an over-subscription case, the sources are transmitting at less than their reservations and it is better in this case not to help a connection to exceed its reservation.

When choosing a scheme, the difficulty of its implementation must also be considered. The scheme proposing a change in TCP sources is very difficult to be implemented. Moreover, it does not give the best performance in case of over-subscription. The scheme dropping packets inversely proportional to the reservation gives good performance but its implementation is also very difficult. If the network is well dimensioned so that it cannot be over-subscribed, a simple scheme such as saw tooth marking gives good performance. If over-subscription is unavoidable, the implementation of the three colors scheme allows a good performance under all subscription levels. The power of the three colors scheme comes from the fact that it gives some priority to some OUT packets over other OUT packets (necessary for the under-subscription case) but it guarantees that this does not exceed the priority of IN packets (necessary for the over-subscription case).

7. Conclusions

We present in this paper a Markovian model for the study of TCP performance in a Differentiated Services network. Our model accounts for the different mechanisms in a DiffServ architecture (marking, dropping), the parameters of the TCP connection (reserved bandwidth, round-trip time, packet size, frequency of ACKs), the parameters of the other TCP connections sharing the same bottleneck with the studied connection, and the available bandwidth in the network. We outline first a general version of the model that depends on two functions: the reaction of a connection to congestion signals, and the probability that a particular connection reduces its rate upon congestion. We calculate the general expression of the throughput of a connection. We specify then the general model to the different proposed DiffServ schemes by simply finding the expressions of the two functions. We also present a simplification of the model in case of large number of connections. Explicit expressions for the throughput are provided and the relation with the models studied in the literature is established. Finally, we solve the model for the different schemes and we show numerically how much the new propositions improve the performance with respect to the original DiffServ scheme. Mainly, we study the service differentiation provided by a scheme and how much it resists to a difference in the subscription level, in the reservation, and in the round-trip time. We believe that our model is a good tool for the validation of new solutions aiming at improving the performance of TCP in a DiffServ network.

References

[1] E. Altman, K. Avratchenkov and C. Barakat, TCP in presence of bursty losses, in: *ACM SIGMETRICS*, 2000.
[2] E. Altman, K. Avratchenkov and C. Barakat, A stochastic model for TCP/IP with stationary random losses, in: *ACM SIGCOMM*, 2000.
[3] C. Barakat, TCP modeling and validation, IEEE Network 15(3) (2001) 38–47.
[4] A. Basu and Z. Wang, A comparative study of schemes for differentiated services, Bell Labs Technical Report (1998).
[5] D. Clark and W. Fang, Explicit allocation of best effort packet delivery service, IEEE/ACM Transactions on Networking 6(4) (1998) 362–373.
[6] K. Fall and S. Floyd, Simulation-based comparisons of Tahoe, Reno, and SACK TCP, Computer Communication Review 26(3) (1996) 5–21.
[7] W. Feng et al., Understanding TCP dynamics in a differentiated services Internet, IEEE/ACM Transactions on Networking (1998).
[8] W. Feng et al., Adaptive packet marking for providing Differentiated Services in the Internet, in: *Internat. Conf. on Network Protocols*, 1998.
[9] S. Floyd, TCP and explicit congestion notification, Computer Communication Review 24(5) (1994) 10–23.
[10] S. Floyd and V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Transactions on Networking 1(4) (1993) 397–413.
[11] J. Heinanen et al., Assured forwarding PHB group, RFC 2597 (1999).
[12] J. Heinanen, T. Finland and R. Guerin, A two rate three color marker, Internet draft (1999).
[13] V. Jacobson, Congestion avoidance and control, in: *ACM SIGCOMM*, 1988.
[14] V. Jacobson, K. Nichols and K. Poduri, An expedited forwarding PHB, RFC 2598 (1999).
[15] R. Jain, D. Chiu and W. Hawe, A quantitative measure of fairness and discrimination for resource allocation in shared computer systems, DEC Research Report TR-301 (1984).
[16] L. Kleinrock, Queueing Systems, Wiley, New York, 1975.
[17] T.V. Lakshman and U. Madhow, The performance of TCP/IP for networks with high bandwidth-delay products and random loss, IEEE/ACM Transactions on Networking 5(3) (1997) 336–350.
[18] K. Nichols, V. Jacobson and L. Zhang, A two-bit differentiated services architecture for the Internet, Internet draft (1997).
[19] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, Modeling TCP throughput: A simple model and its empirical validation, in: *ACM SIGCOMM*, 1998.
[20] W. Stevens, TCP slow-start, congestion avoidance, fast retransmit, and fast recovery algorithms, in: *RFC 2001*.
[21] I. Yeom and A. Reddy, Realizing throughput guarantees in differentiated services networks, TAMU-ECE-9806 (1998).
[22] I. Yeom and A. Reddy, Modeling TCP behavior in a differentiated-services network, TAMU ECE Technical Report (1999).