Department of Applied Mathematics
Faculty of EEMCS

University of Twente
*The Netherlands*

P.O. Box 217
7500 AE Enschede
The Netherlands
Phone: +31-53-4893400
Fax: +31-53-4893114
Email: memo@math.utwente.nl
www.math.utwente.nl/publications

**Memorandum No. 1734**

# Versatile Markovian models for networks with asymmetric TCP sources

N.D. van Foreest, B.R. Haverkort,
M.R.H. Mandjes and W.R.W. Scheinhardt

August, 2004

ISSN 0169-2690

# Versatile Markovian Models for Networks with Asymmetric TCP Sources

N.D. van Foreest [a], B.R. Haverkort [a], M.R.H. Mandjes [b,a], and W.R.W. Scheinhardt [a,b]

[a] *Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, P.O. Box 217, 7500 AE, Enschede, The Netherlands*

[b] *Center for Mathematics and Computer Science (CWI), P.O. Box 94079, 1090 GB, Amsterdam, The Netherlands*

**Abstract**

In this paper we use Stochastic Petri Nets (SPNs) to study the interaction of multiple TCP sources that share one or two buffers, thereby considerably extending earlier work. We first consider two sources sharing a buffer and investigate the consequences of two popular assumptions for the loss process in terms of fairness and link utilization. The results obtained by our model are in agreement with existing analytic models or are closer to results obtained by ns-2 simulations. We then study a network consisting of three sources and two buffers and provide evidence that link sharing is approximately minimum-potential-delay-fair in case of equal round-trip times.

*Key words:* TCP, fluid model, Stochastic Petri Nets, fairness analysis
*AMS subject classification (2000): 60K25, 60J22, 90B18.*

## 1 Introduction

Many models of the interaction between TCP sources and buffers have been developed over the last few years. One aim of these models is to obtain insight into how efficiently and fairly TCP sources use link and buffer capacity in the Internet. Some of these models, e.g., [2,3], focus on these aspects in the setting of two sources that share one bottleneck link. Other models, e.g., [20,23], are concerned with the performance of large networks as a whole. The models for two sources and one buffer are probabilistic whereas the approaches that generalize to networks, such as, [20,23], are, with respect to the analysis, completely deterministic.

As yet, there is no model that applies well to networks of *intermediate size*, i.e., networks consisting of a few sources and buffers, while retaining a stochastic flavor.

This stochasticity at packet level is arguably not of much importance in very large networks where the presence of a single flow is hardly noticeable. However, in these intermediately-sized networks this is important; here, one connection can add considerably to congestion. Hence, it is of interest to have a stochastic model of the source and buffer processes that, at the same time, can cope with intermediately-sized networks.

The model developed in [16] for two TCP sources interacting with a buffer is simple, yet flexible, and in principle extendable to networks of the desired size, that is, a few sources and buffers. It is formulated in terms of a continuous-time Markov chain and enables us to study various stochastic aspects of the interaction. However, its use is somewhat limited in practice since the generator matrix of the Markov chain has to be constructed manually. This process is error-prone and rather time consuming. Moreover, it becomes increasingly difficult to correctly implement extensions to multiple sources or multiple buffers. To make this additional step we need a different methodology to specify the Markov chain and obtain the generator. A suitable framework to extend the model of [16] is provided by Stochastic Petri Nets (SPNs), see e.g.,[1].

A SPN is a (graphical) formalism to describe systems which exhibit complicated dynamics. It incorporates a notion of state and a set of rules describing the allowed state changes, thereby capturing static and dynamic characteristics of complex systems such as communication systems. The fact that the SPN is a graphical representation of (a model of) a system contributes to the understanding of (the dynamics of) the system. Moreover, computer tools such as the Stochastic Petri Net Package (SPNP) [12] exist that automatically map a SPN to an underlying Markov chain and generate the corresponding infinitesimal generator. Using this generator, SPNP can compute stationary and transient performance measures formulated as expected reward functions on the SPN. Clearly, tools as SPNP handle the more cumbersome aspects of the performance analysis of complicated systems while the user can concentrate on the aspects related to *modeling* and *design*.

In the current work we apply SPNs to study the interaction between multiple TCP sources and buffers in intermediately-sized networks. This approach allows us to express various performance measures of interest, such as packet loss probability, the throughput, file transfer latency, and so on, as reward functions, which can therefore be automatically computed by SPNP.

With SPNs we can generalize the TCP models of [2,3] and [16] considerably in that we can handle large buffers and networks with more than just one node. Especially the last aspect seems difficult to incorporate in the setting of [2,3]. In contrast to [23], in which the analysis is deterministic, our model is entirely stochastic. Therefore, our model permits, for instance, to switch off an FTP session at a rate depending on the source's transmission rate. Another advantage of the probabilistic approach is that we can easily compute higher moments of the distribution of, e.g.,

buffer fill levels. The authors of [5] present an interesting mixture of a deterministic and stochastic model of TCP. There, a deterministic model controls the behavior of the network (source states and queue lengths) as long as buffers are not congested; at loss epochs a simulator distributes the loss over the sources, thereby introducing stochasticity. This approach, however, still relies on simulation.

The authors of [7] also use Markovian models of TCP. However, they develop a model of a *single* TCP Tahoe source, and then consider a superposition of *statistically independent* sources that feed traffic into an M/M/1/K queue. By fixed point methods, see also e.g. [17] or [4], they compute performance measures. In [8,9] these authors consider TCP Reno instead of TCP Tahoe in the same framework. Our approach is different in that we take the source and buffer process to be dependent, which is important in intermediately-sized networks.

As our approach shows considerable resemblance to the fluid model as developed in [23], we start by summarizing this work in Section 2. Then, in Section 3, we specify a SPN of two TCP sources that share a buffer. With this model we consider two popular models for the distribution of loss over the sources: synchronized and proportional loss, cf., [3]. In the former model, during congestion all connections simultaneously reduce their window size by a factor two. In the latter model just one source is selected to suffer from loss with a probability that is proportional to the window size. We implement both loss models and, in Section 4, extensively compare the results to, on the one hand, theoretical results provided by [2,3,18], and, on the other, simulation results obtained with ns-2, see [25]. In Section 5 we first present some further possible extensions of the source model. Then we consider a network consisting of three sources and two buffers. The implementation of the throughput formulas in this SPN are 'topology aware' in that they respect the order in which packets of a TCP connection traverse the buffers. Hence, effects such as shaping at up-stream buffers are taken into account. We compare the sharing of link capacity to the minimum-potential-delay fairness scheme as defined in [21] and which, according to the authors of [19], is the most appropriate for TCP. Section 6 concludes the paper. In Appendix A we introduce some necessary background on SPNs.

## 2 Fluid Markov Model of TCP

Here we apply the model developed in [23] to a network of $J$ greedy TCP sources that share a buffer of size $B$ and a link with capacity $L$. The buffer uses a Random Early Detect (RED) packet dropping scheme, see [14]. For later reference to the RED parameters we include a short description of RED. We summarize the differential equations that govern the dynamics of the source and buffer processes as derived in [23]. Then we point out two shortcomings of this model and clarify in which respects our approach circumvents these problems.

Let us first concentrate on the dynamics of the sources. Suppose that $T_i$ is the round-trip time for source $i$ when the buffer is empty. Then,

$$T_i(q(t)) = T_i + \frac{q(t)}{L} \tag{1a}$$

is the round-trip time of source $i$ when the buffer content is $q(t)$ at time $t$. Source $i$ maintains a window variable $W_i(t)$, supposed to be continuous, and sends fluid at rate $\mathrm{MSS}_i W_i(t)/T_i(q(t))$, where $\mathrm{MSS}_i$ is the packet size, into the RED buffer. The window dynamics behave according to the Additive-Increase/Multiplicative--Decrease (AIMD) scheme as described in [10]. More specifically,

$$dW_i(t) = \frac{dt}{T_i(q(t))} - \frac{W_i(t)}{2} dM_i(t). \tag{1b}$$

The first term of the right hand side corresponds to the Additive-Increase behavior of a source. The second term implements the Multiplicative Decrease at a loss epoch. Here, $M_i(t)$ models the loss arrivals as a point process, so that $dM_i(t) = 1$ at the arrival of a loss and 0 elsewhere.

The evolution of the queue depends on the rates of the $J$ sources through the differential equation

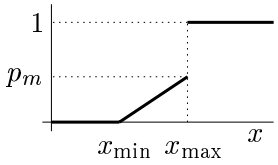
$$\frac{dq}{dt} = \sum_{i=1}^{J} \frac{\mathrm{MSS}_i W_i(t)}{T_i(q(t))} - L, \tag{1c}$$

provided $q(t) \in (0, B)$, otherwise $dq/dt = 0$.

The RED buffer operates, roughly, as follows. (Consult, for instance, [24] for a more detailed description of RED.) The RED buffer maintains an estimate $x$ of the average queue length. At each packet arrival this estimate is updated according to an exponentially weighted moving average with weight $\epsilon \in (0, 1)$. If $q_i$ is the queue length observed at the arrival of packet $i$ then

$$x_i = (1 - \epsilon) x_{i-1} + \epsilon q_i. \tag{2}$$

The RED buffer drops packet $i$ with probability $p(x_i)$, where $p(x)$ has the form




$$\text{i.e.,} \quad p(x) = \begin{cases} 0, & 0 \le x \le x_{\min} \\ \frac{x - x_{\min}}{x_{\max} - x_{\min}} p_m, & x_{\min} < x \le x_{\max} \\ 1, & x_{\max} < x. \end{cases} \tag{3}$$

In other words, the buffer drops a fraction $p(x_i)$ of the arriving packets when $x = x_i$.

To analyze (1–2) the authors of [23] take expectations at both sides of (1–2) and make a number of simplifying assumptions to obtain a (numerically) tractable system of differential-equations. They solve the resulting system of differential alge-

braic equations with Matlab from which they find the expected transient behavior of, for instance, the queue.

The main advantage of this approach is its flexibility, e.g., timeouts can be taken into account, and its scalability, e.g., thousands of sources can be easily handled. Still, two fundamental problems with this approach exist. In the first place, the entire analysis is set up in terms of *averages* of connections. Thus, it does not include any knowledge of individual connections. However, this is an important point, for example, when sources switch off with a rate that depends on their transmission rate, which is typically the case with file transfers. The second problem relates to a more technical aspect of the analysis. In [23] there is no mention of how to *compute* the expectations that are taken. (In technical terms, the probability space is not provided.)

Our model does not suffer from these problems. More specifically, with regard to the first point our stochastic model maintains a notion of the momentary window size and buffer content so that the momentary (fluid) transmission rate is known. With respect to our second comment, we also take expectations, but with respect to a stationary distribution of a Markov chain, so that no problems about the interpretation remain. (In a loose sense, we *first* solve the system and *then* take expectations, while the authors of [23] take expectations first and then solve the system. Reversing this ordering is not a mere technicality.) A disadvantage of our different approach as compared to [20,23] is that our model does not scale to networks of sizes studied there.

## 3 A SPN Specification of the TCP Fluid Model

In the model to be introduced now we aim to avoid some of the drawbacks of the fluid model of Section 2. The central idea is to discretize the source and queue processes; this is the topic of Section 3.1. Then, in Section 3.2, we represent the discretized joint source and buffer processes as a Stochastic Petri Net (SPN). Here we assume familiarity with basic concepts of SPNs; we refer to Appendix A for a brief survey. The first SPN we define contains a buffer that uses the proportional loss scheme, the details of which are explained in Section 3.2.3 below. The resulting SPN is similar to the model summarized in Section 2; in fact most of (1) carries over. However, we do not take expectations to simplify the analysis but compute the steady state probabilities $\pi$ of the SPN instead. In Section 3.3 we use $\pi$ to define the throughput in terms of a reward function on the SPN. To obtain insight into the time required to compute $\pi$ we discuss some computational issues in Section 3.4. Finally, Section 3.5 shows that it is nearly trivial to modify the SPN with proportional loss to a SPN that implements synchronized loss, that is, a loss model according to which all sources react in synchrony to congestion by all reducing their rate simultaneously.

To facilitate the presentation of the SPN we first model the source window size and buffer content as discrete processes and discuss some immediate consequences. Mainly to keep the model concise we reduce the RED buffer of Section 2 to a drop-tail buffer by choosing $x_{\min} = x_{\max} = B$ in (3) and $\epsilon = 1$ in (2). We assume that the sources use a TCP version, such as TCP New-Reno [13] or TCP Sack [22], that does not (frequently) resort to timeouts and slow starts when multiple losses occur in one window. The notation is the same as in the previous section, e.g., $J$ is the number of sources, etc.

The buffer process as determined by (1c) can take any value in the range $[0, B]$. In the sequel we modify this process to a corresponding *discrete* process $\{C(t)\}$ with state space $\{0, 1, \ldots, K\}$. When $C(t) = k$, $0 \leq k \leq K$, the corresponding buffer content equals $kB/K$. Thus, in this case the round-trip time for source $i$ becomes, cf. (1a):

$$T_i(k) = T_i + \frac{k}{K}\frac{B}{L}. \tag{4}$$

The window process $\{W_i(t)\}$ of source $i$ is a discrete process with state space $\{0, 1, 2, \ldots, N_i\}$. Here $N_i$ denotes the maximum window of source $i$, which is an important parameter in characterizing the performance of TCP. When $W_i(t) = n_i$ and $C(t) = k$ the source sends traffic at rate $\mathrm{MSS}_i n_i / T_i(k)$. In the sequel we often use the shorthand

$$r_i(k) = \frac{\mathrm{MSS}_i}{T_i(k)}, \tag{5}$$

so as to write the source rate as $n_i r_i(k)$. We remark that the source peak rate is given as $N_i r_i(0)$.

Observe that if the buffer content were a continuous variable, the net input rate at time $t$ is given by, cf. (1c),

$$\mathbf{r}(k) \cdot \mathbf{n} - L \equiv \sum_i r_i(k) n_i - L, \tag{6}$$

where $\mathbf{n} = (n_1, \ldots, n_J)$.

Finally, we introduce the processes $\{l_i(t), t \geq 0\}$, $i = 1, \ldots, J$. When $l_i(t) = 0$ source $i$ is allowed to increase its rate, while when $l_i(t) = 1$ it should decrease its rate. The next section provides further clarification for these processes.

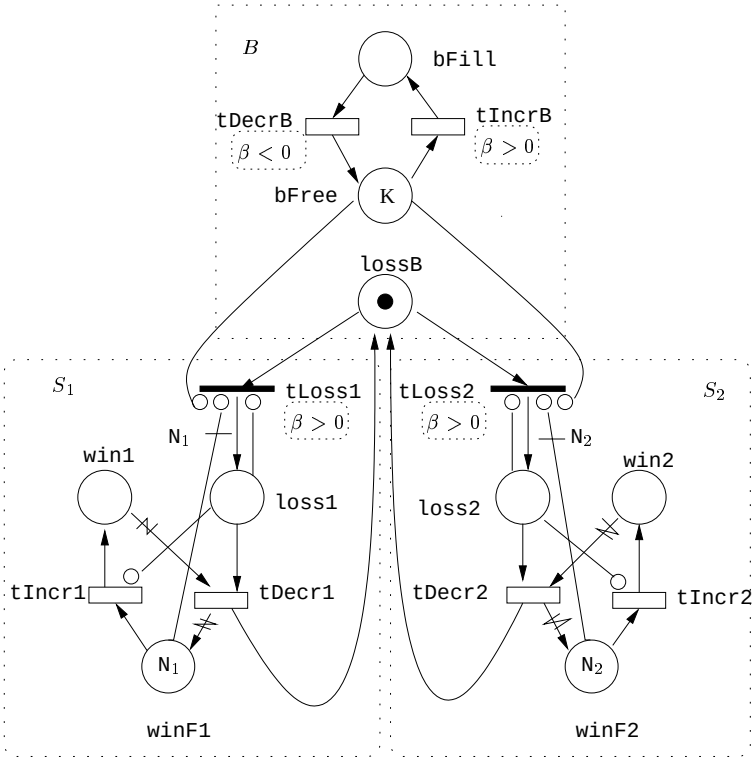



Fig. 1. A Petri Net model of two TCP sources sharing a buffer. The loss model is taken according to the loss proportional scheme. We indicate guards by means of dashed boxes around strings, such as $\beta < 0$ appearing immediately below `tDecrB`.

### *3.2 SPN for Two Sources and One Buffer with Proportional Loss*

In this section we provide a specification of a SPN consisting of two TCP sources sharing one buffer. In other words, we specify the behavior of the joint process $\{W_1(t), W_2(t), l_1(t), l_2(t), C(t), t \geq 0\}$ introduced in Section 3.1 by means of the SPN shown in Figure 1. The SPN contains three 'subnets' indicated by dashed boxes around a number of places and transitions. The subnets $S_1$ and $S_2$ represent the sources while the subnet $B$ represents the buffer. We describe these subnets first, then we focus on the dynamics of the complete SPN.

#### *3.2.1 The Subnets*

Subnet $S_1$ contains three places: `win1`, `winF1` and `loss1`; one immediate transition: `tLoss1`; and two timed transitions: `tIncr1` and `tDecr1`. Subnet $S_2$ is, except for the naming, identical; as such the rest of the discussion applies equally well to source 2. The state of source 1 is given by the markings of `win1`, `winF1` and `loss1`, respectively. Here, the number of tokens in `win1`, i.e., #`win1`, denotes the momentary congestion window of source 1. The marking of `winF1` denotes how much further the window can increase. Initially, #`winF1` $= N_1$, so that at all

$$W_1(t) = \#\texttt{win1} \quad W_2(t) = \#\texttt{win2} \quad C(t) = \#\texttt{bFill}$$
$$l_1(t) = \#\texttt{loss1} \quad l_2(t) = \#\texttt{loss2} \quad l_b(t) = \#\texttt{lossB}.$$

Table 1
The correspondence between the stochastic processes of Section 3.1 and the markings of places in Figure 1.

times during the evolution of the SPN it holds that $\#\texttt{win1} + \#\texttt{winF1} = N_1$. The loss state of source 1 is indicated by $\#\texttt{loss1}$. When $\#\texttt{loss1} = 0$ the source is allowed to increase $\#\texttt{win1}$, while when $\#\texttt{loss1} = 1$ the source has to reduce $\#\texttt{win1}$ by a factor 2.

The buffer subnet contains three places: `bFill`, `bFree`, and `lossB`; and two timed transitions: `tDecrB` and `tIncrB`. The marking of the place `bFill` is the fill level of the buffer. The place `bFree` has initially $K$ tokens. Its marking corresponds to the free space, i.e., the maximum buffer level $K$ minus the fill level $\#\texttt{bFill}$; thus, $\#\texttt{bFill} + \#\texttt{bFree} = K$. Finally, when $\#\texttt{lossB} = 0$ ($\#\texttt{lossB} = 1$) the buffer is (not) congested.

Observe that, for instance, the window size process $\{W_1(t), t \geq 0\}$ and the process $\{\#\texttt{win1}, t \geq 0\}$ are identical processes. For notational brevity and consistency with the previous sections we use in the sequel $W_1(t)$ instead of $\#\texttt{win1}$ to denote the marking of `win1`, etc. Table 1 shows the relation between the variables and the markings of the other places in the SPN. We also occasionally drop the dependency on $t$ of the processes $W_1(t)$, etc.

The *marking-dependent* firing rates and guards associated with all transitions are summarized in Table 2. Here $T_i(k)$ is given by (4). The function

$$\beta(n_1, n_2, k) = \frac{K}{B}(\mathbf{r}(k) \cdot \mathbf{n} - L) \tag{7}$$

is the transition rate at which in- and decrements of the buffer level process occur. By comparing this expression to (6) it is clear that $\beta$ is equal to $K/B$ times the net input rate of the buffer. The constant $K$ appears in the numerator to ensure that the average time required to fill an empty buffer (or drain a full buffer), given that the window size does not change, is approximately independent of $K$. (Loosely speaking, when $K$ is large, the average time spent at one fixed buffer level should be short.)

### *3.2.2 From Initial State to Congestion (Congestion Avoidance)*

The initial marking of the SPN is as shown in Figure 1. Sources 1 and 2 are 'off' and not in a loss state while the buffer is empty and in possession of the loss token. In the initial state only `tIncr1` and `tIncr2` are enabled and fire at rate $1/T_1$ and $1/T_2$, respectively. Each firing increases $W_1$ or $W_2$ by one, which clearly models the

| Transition | Rate | Guard |
|---|---|---|
| tIncrB | $\beta(n_1, n_2, k)$ | $\beta(n_1, n_2, k) > 0$ |
| tDecrB | $-\beta(n_1, n_2, k)$ | $\beta(n_1, n_2, k) < 0$ |
| tLoss1 | $\infty$ | $\beta(n_1, n_2, k) > 0$ |
| tLoss2 | $\infty$ | $\beta(n_1, n_2, k) > 0$ |
| tIncr1 | $T_1^{-1}(k)$ | — |
| tDecr1 | $T_1^{-1}(k)$ | — |
| tIncr2 | $T_2^{-1}(k)$ | — |
| tDecr2 | $T_2^{-1}(k)$ | — |

Table 2
Rate functions and guards for the transitions in Figure 1.

Additive-Increase phase of TCP. Note that on average source $i$ spends an amount $T_i(k)$ in state $n_i$, given $C = k$. In this way the SPN incorporates feedback delay.

As $W_1$ and $W_2$ increase, the scaled net input rate (7) increases as well. After a number of firings of tIncr1 and tIncr2, $W_1$ and $W_2$ are so large that $\beta(W_1, W_2, 0)$ becomes positive. This will set the guard at tIncrB to true, so that tIncrB becomes enabled. Each firing of tIncrB increments $C$ by one. After $K$ firings of tIncrB, the buffer is completely filled, i.e., $C = K$. As a result, the inhibitor arcs from bFree to tLoss1 and tLoss2 are now no longer active, so that the random switch consisting of the immediate transitions tLoss1 and tLoss2 becomes enabled.

Suppose tLoss1 fires first so that source 1 receives the loss token. As such, the loss token represents the congestion signal that the buffer sends to a source. Clearly, in this case the inhibitor from loss1 to tIncr1 will prevent further increments of the window of source 1. Note that, as source 1 receives the loss token, loss2 does not become marked (unlike the synchronous case, to be discussed later), and consequently, tIncr2 can still fire.

It is evident that when source $i$ is inactive, i.e., when $W_i = 0$, it cannot suffer from loss. To prevent the loss token from being sent to a quiet source there is a multiple inhibitor arc from winF1 (winF2) to tLoss1 (tLoss2) with multiplicity $N_1$ ($N_2$). The multiplicity is indicated in Figure 1 at the inhibitor arc.

### *3.2.3 The Proportional Loss Model*

The buffer uses a proportional loss model, according to which the buffer chooses only one connection to suffer from loss during overload. The probability to select a particular connection is proportional to its momentary transmission rate. We im-

| | $r_1(K)W_1 \leq L$ | $r_1(K)W_1 > L$ |
|---|---|---|
| $r_2(K)W_2 \leq L$ | $p_1 = \frac{r_1(K)W_1}{\mathbf{r}(K)\cdot\mathbf{W}}$ | $p_1 = 1$ |
| $r_2(K)W_2 > L$ | $p_1 = 0$ | $p_1 = \frac{r_1(K)W_1}{\mathbf{r}(K)\cdot\mathbf{W}}$ |

Table 3
Firing probability $p_1$ of `tLoss1`.

plement this behavior by means of the random switch consisting of `tLoss1` and `tLoss`.

The marking-dependent weights of the random switch are chosen such that `tLoss1` fires with probability $p_1$ while `tLoss2` fires with probability $p_2 = 1 - p_1$. Table 3 shows the values of $p_1$ when $r_1(K)W_1 > L$ and $r_2(K)W_2 > L$, etc. The motivation behind this loss model is based on the insight that if, for instance, $r_1(K)W_1 > L$ and $r_2(K)W_2 \leq L$, connection 1 certainly loses traffic. Thus, in this case connection 1 should surely receive the loss token. Due to the proportional loss model there is just one loss token, so that connection 2 cannot receive a loss token. Therefore, in this case, $p_1 = 1$ and $p_2 = 0$. When $r_1(K)W_1 \leq L$ and $r_2(K) \leq L$ (but $\mathbf{r}(K) \cdot \mathbf{W} > L$) both sources can be hit by a loss with a probability proportional to their sending rates. In the (very) rare case that $r_1(K)W_1 \geq L$ and $r_2(K)W_2 \geq L$ both sources should reduce their rate. However, as `lossB` contains just one token, it cannot simultaneously send both sources a loss token. Therefore, we again take the loss probabilities proportional to the sending rates. We emphasize that the impact of this inconsistency will be small in nearly all relevant parameter settings.

### *3.2.4 Removing the Congestion (Multiplicative Decrease)*

When $l_1 = 1$ the timed, variable transition `tDecr1` is enabled. Once it fires, it moves the loss token from `loss1` to `lossB`, removes half of the tokens from `win1`, and adds these to `winF1`. In other words, $W_1$ is reduced by a factor two, reflecting the Multiplicative Decrease after the detection of loss. If, with the new marking, still $\beta(W_1, W_2, K) > 0$ either `tLoss1` or `tLoss2` will immediately fire again. After a sufficient number of multiplicative decrements of $W_1$ and $W_2$ the net input rate becomes negative. When this is the case, firings of `tDecrB` decrement the buffer content. Note that another consequence of $\beta(W_1, W_2, C) < 0$ is that the guards at `tLoss1` and `tLoss2` prevent the loss token from being passed on to either of the sources. Thus, their windows cannot decrease any further.

We have not yet discussed two arcs: the inhibitors from `loss1` and `loss2` to `tLoss1` and `tLoss2` respectively. Their role will be clarified in the synchronous loss model presented in Section 3.5. In the proportional model they have no function, but they do not influence the performance measures in any way either.

We now express in terms of the reward functions, as defined in equation (A.1), three performance measures: a connection's (average) transmission rate, its throughput and the utilization of the link.

**1.** The expected transmission rate for connection $i$ is easy:

$$\tau_i = \mathbb{E}\left\{r_i(C)W_i\right\}.$$

**2.** The throughput is not as simple to specify in exact terms in the present setting. We discuss two proposals and compare these numerically in Section 4.

**2a.** For the first proposal we consider the fluid that *enters* the buffer. As long as the buffer is not full it accepts fluid, but when it is full it drops the excess fluid. We assign this excess to the connection that receives the loss token. There is only excess traffic when $C = K$ and $\beta(W_1, W_2, C) > 0$. As these conditions imply, and are implied by, the condition that either $l_1 = 1$ or $l_2 = 1$ (due to the proportional loss assumption), we define

$$\gamma_i^{\text{in}} = \tau_i - \mathbb{E}\left\{(\mathbf{r}(C)\cdot\mathbf{W} - L)1_{\{l_i=1\}}\right\}, \tag{8}$$

where $1_A = 1$ if the condition $A$ is true and 0 otherwise. This definition assigns *all* excess fluid $e = \mathbf{r}(C)\cdot\mathbf{W} - L$ to *one source*: the source that receives the loss token. As a consequence we need to verify whether indeed $r_i(K)W_i > e$ when source $i$ receives the loss token, for otherwise we subtract too much in the above. In view of Table 3 observe that, for instance, the condition $r_1(C)W_1 < L$ is equivalent to $r_2(C)W_2 > e$. Thus, the problem of subtracting too much may only occur when both source rates exceed $L$, As this happens (very) rarely, we neglect it, as we did in Section 3.2.3.

**2b.** The other possibility is to consider the fluid that *leaves* the buffer. When the buffer is empty the departure rate at time $t$ is equal to the arrival rate. When at time $t$ the buffer contains $k > 0$ units of fluid the departure rate of source $i$ at time $t$ equals the service capacity $L$ times the fraction of traffic of source $i$ that arrived at time $t - kB/(KL)$. As the Markov process $\{W_1(t), W_2(t), l_1(t), l_2(t), C(t)\}$ does not maintain the history of the source states as supplementary variables, the source rates at time $t - kB/(KL)$ are (principally) unknown. Hence we cannot incorporate the

effect of buffering delay on the throughput. We therefore approximate the output process by the arrival process and neglect the impact of the delay. This yields

$$\gamma_i^{\text{out}} = \mathbb{E}\left\{ r_i(C)W_i 1_{\{C=0\}} + L\frac{r_i(C)W_i}{\mathbf{r}(C)\cdot\mathbf{W}} 1_{\{C>0\}} \right\}. \tag{9}$$

To see that this approximation is acceptable we reason as follows. Observe that the round-trip times of all sources include the buffering delays along the route. Hence, it always takes less time to refresh the buffer content than it takes for a source to change its rate. Consequently, while the buffer content is refreshed the input rates are nearly constant. We conclude that neglecting the delay only shifts the output process backward in time, but does not substantially change its shape or the ratio of fluid of the sources.

**3.** We define utilizations as

$$u_i = \frac{\gamma_i}{L}, \quad i = 1, 2 \qquad\qquad u = \frac{\gamma_1 + \gamma_2}{L} = u_1 + u_2.$$

With respect to the existence of the stationary distribution $\boldsymbol{\pi}$, which is needed in the computation of the expectations above, we remark that the Markov chain associated with the proportional SPN may be reducible, depending on the choice of parameters. This has, however, no consequence for the existence of $\boldsymbol{\pi}$. While some states may belong to transient classes, the other states form *one* recurrent class so that a unique stationary distribution still exists.

### *3.4 Computational Issues*

It is of interest to estimate the size $|\mathscr{M}|$ of the Markov chain as this gives insight into the time required to solve for the stationary distribution. We have not been able to find an accurate, yet simple, expression for $|\mathscr{M}|$, mainly due to the fact that the number of recurrent states depends critically on the values of the system parameters and the presence of guards. A simple, coarse estimator is obtained below under some additional assumptions.

Observe that the model possesses some scaling freedom in the parameters $\text{MSS}_i$, $i = 1, 2$, $L$ and $B$. To remove this freedom assume henceforth that source 2 is the distant one, i.e., $T_2 \geq T_1$, and set $r_2(0) = 1$. Moreover, assume that the packet sizes are equal, i.e., $\text{MSS}_1 = \text{MSS}_2$, which has as a consequence that

$$r_1(k) = \frac{\text{MSS}_2}{T_1(k)} = \frac{T_2}{T_1(k)},$$

as $r_2(0) = 1$. Next, suppose that the source rates are not constrained by the receiver windows. Thus, each source can congest the link. This is achieved by setting $N_i \geq \lceil L/r_i(K) \rceil = \lceil LT_i(K)/T_2 \rceil$, where $\lceil x \rceil$ is the smallest integer larger than $x$. Thus, we see that $L$ determines the source granularity: a small (large) value of $L$ means that a few (many) source transitions are needed for buffer overflow. Note that choosing $N_i$ much larger than $\lceil LT_i(K)/T_2 \rceil$ hardly affects the value of the performance measures. Clearly, such 'high' source states are only relatively seldom visited. From numerical evaluations we conclude that choosing $N_i$ equal to $1.1\lceil LT_i(K)/T_2 \rceil$ is large enough for our purposes. Finally, in the (numerical) analysis we wish to specify the buffering delay $d_B$ instead of the buffer size $B$ itself; therefore let $B = d_B L$. As a result the parameters $T_1$, $T_2$, $L$ and $d_B$ now fully characterize the system.

To obtain an upper bound on $|\mathscr{M}|$, observe that the number of different markings of `bFill` is, obviously, $K + 1$, and that the loss token can reside in three places: `lossB`, `tLoss1`, and `tLoss2`. Combining this with the above choice for $N_1$ and $N_2$ we obtain

$$\begin{aligned} |\mathscr{M}| &\leq 3(K+1)(1.1)^2 \left( \left\lceil \frac{LT_1(K)}{T_2} \right\rceil + 1 \right) \left( \left\lceil \frac{LT_2(K)}{T_2} \right\rceil + 1 \right) \\ &= O((K+1)(N_1+1)(N_2+1)). \end{aligned} \tag{10}$$

This estimate is an upper bound as the guards in the SPN considerably reduce the number of transitions so that not all markings counted in this formula represent states in the reachability set. Observe that as the SPN contains only eight transitions the number of non-zero entries per row in the generator is also bounded by eight. Consequently, the generator is sparse.

Clearly, from the computational point of view it is of interest to choose $N_1$, $N_2$ and $K$ as small as possible without significantly affecting the overall results, i.e., the outcome of the performance measures. The following provides some insight into how small $N_1$, etc., can be reasonably chosen. As a first step we notice that as source 2 is the distant source, it follows that

$$N_2 = 1.1\lceil LT_2(K)/T_2 \rceil \geq 1.1\lceil LT_1(K)/T_2 \rceil = N_1.$$

However, the probability that $W_2 > N_1$ is small. Therefore, we can safely set $N_2 = N_1$. The value of the parameter $K$ is also of importance in the estimation of $|\mathscr{M}|$. It is plausible that for $K \to \infty$, $\{C(t)\}$ becomes a continuous process; see e.g., [15] for an application of such fluid queues to TCP modeling. Thus, we infer that the performance measures hardly change for increasing $K$ beyond a certain number. It turns out that $K = 5$, a relatively small number, is already large enough for the parameter ranges we investigate in this paper; setting $K$ to larger values makes practically no difference. We can even set $K = 1$ in the large bandwidth-delay product regime, i.e., when the maximum buffering delay $d_B$ is small in comparison to the propagation delays. In this case the time it takes for an AIMD source to 'fill

the pipe' is much longer than the buffer filling time. Thus, approximately, the buffer is either empty or congested.

We remark that the numerical results to be presented below provide support for these approximations.

### *3.5 The Synchronous Loss Model*

Here we show how to change the SPN with proportional loss to a SPN that uses the synchronized loss model.

First of all, it is clear that to signal both sources about congestion it is necessary to have *two* loss tokens initially present at `lossB`. Secondly, in the new setting the transitions `tLoss1` and `tLoss2` will no longer form a random switch. Instead, each fires with probability 1, if enabled.

Now suppose again that once `bFree` becomes empty, `tLoss1` is the first to fire. This results in one of the loss tokens to move to `loss1`. The inhibitor from `loss1` to `tLoss1` prevents this transition to fire again. Consequently, the immediate transition `tLoss2` fires so that both sources receive a loss token at the same instant. As long as `loss1` (`loss2`) is marked source 1 (source 2) cannot receive a loss token which becomes available after a firing of `tDecr2` (`tDecr1`) due to the inhibitor arcs from `loss1` (`loss2`) to `tLoss1` (`tLoss2`).

As both sources receive a loss token, both sources can take their share of the excess fluid arriving during congestion. Therefore the throughput as defined in (8) should become

$$\gamma_i^{\text{in}} = \tau_i - \mathbb{E}\left\{ (\mathbf{r}(C)\cdot\mathbf{W} - L)\frac{r_i(C)W_i}{\mathbf{r}(C)\cdot\mathbf{W}} 1_{\{C=K,\beta(W_1,W_2,C)>0\}} \right\}. \qquad (11)$$

Definition (9) does not need any modification.

Finally, with respect to the size of the state space we remark that the number of possibilities for the loss tokens is 4 rather than 3. Thus, the bound in (10) should be multiplied with $4/3$ to obtain an upper bound for the size of the reachability set of the SPN with synchronized loss.

## 4 A Comparison with Analytic Models and ns-2

In this section we compare the numerical results of the SPN to other analytic work and simulation with ns-2. We specify the investigated scenarios in Section 4.1 and present the results in Section 4.2.

| Scenario | $L$ | $d_B$ (ms) | $K$ |
|---|---|---|---|
| '25s' | 25.7 | 16 | 1 |
| '80s' | 80.7 | 16 | 1 |
| '25l' | 25.7 | 160 | 5 |
| '80l' | 80.7 | 160 | 5 |

Table 4
SPN parameter values. We use mnemonics such as '25s' to denote the scenario in which the link rate is '$25.7$' and the maximum buffering delay $d_B$ is 'small'. A buffering delay of $16$ ms corresponds to a buffer size of $5$ packets in the ns-2 simulation.

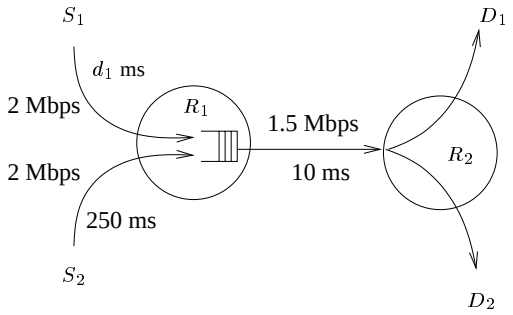



Fig. 2. The network configuration and some parameters. For the purpose of comparison we use the same parameters as in [2].

### *4.1 Scenarios*

Figure 2 shows the network we used for the numerical analysis and ns-2. To facilitate the comparison with [2] we use the same parameters. Two greedy TCP sources, $S_1$ and $S_2$, communicate with destination $D$ via a router with buffer size $B = d_B L$. The receiver windows are assumed to be so large that they do not constrain the congestion windows. Table 4 shows the values of $L$ and $d_B$ for the four investigated scenarios. For each scenario we vary the propagation delay $d_1$ of the link connecting $S_1$ and $R_1$ in $10$ steps from $40$ ms to $240$ ms. The rest of the parameters of the SPN now follow from Section 3.4.

In the simulations with ns-2 we use a RED buffer and consider a small and a large buffer case. In the former (latter) the buffer's total size is 20 (200) packets. The RED parameters in (2–3) are taken as follows. The minimum threshold $x_{\min}$ is 5 (50) packets, the maximum threshold $x_{\max}$ is 10 (55) packets, the maximum drop probability $p_m = 0.1$ and the weight $\epsilon = 0.002$. The packet size is, including IP header, $576$ Bytes. (The RED parameter values for the small buffer are also identical to the values chosen in [2].)

Note that the buffer in the SPN is a drop-tail type buffer instead of a RED buffer, which, on the face of it, is inconsistent with the simulated network. As a motivation for using RED in ns-2 we follow an argument of [2]. It is commonly seen

in simulations with two sources sharing one drop-tail buffer that sometimes one and sometimes both sources lose packets during a congested period. Thus, at least in simulations, bursts at the packet level determine which source(s) lose(s) traffic in case a drop-tail buffer overflows. However, such rapid fluctuations at the packet level are absent in the context of fluid sources. Thus, a fluid source never perceives a 'true' drop-tail buffer. As such, comparing fluid models to simulations with (small) drop-tail buffers will not be appropriate. As RED is a queue management technique that can effectively absorb these rapid queue-length fluctuations, it is apt to use RED buffers in the simulations even when the modeled fluid buffer is a drop-tail buffer.

We remark here that a consequence of using RED in the simulations is that packets will be dropped with a probability proportional to the sending rate of a source. Thus, our proportional loss model is the more appropriate to compare against the simulations.

### *4.2 Results*

In [18] it has been derived that in case of synchronous loss and a small buffer the ratio of the throughputs $\gamma_1/\gamma_2 \approx s^{-\alpha}$, where $s = T_1/T_2$ is the ratio of propagation delays and $\alpha = 2$. The authors of [2] present a model with proportional loss in which $\gamma_1/\gamma_2 = s^{-\alpha}$ with $\alpha \approx 0.85$. As our model allows to analyze both loss models we investigate what values for $\alpha$ the model will give in either case. Moreover we analyze the impact of buffering delay. The left and right panel of all figures of this section show the results for the small and large buffer case, respectively.

In Figure 3 we plot the throughput ratio computed by the model with proportional loss as a function of $s$ for the scenarios of Table 4. We compare the differences between the 'input ratio' $\gamma_1^{\text{in}}/\gamma_2^{\text{in}}$ obtained by (8) and the 'output ratio' $\gamma_1^{\text{out}}/\gamma_2^{\text{out}}$ by (9). We also plot $s^{-\alpha}$ for several values of $\alpha$. Finally we mention an analytic estimate as derived in [3]:

$$\frac{\gamma_1}{\gamma_2} \approx \frac{1}{s}\frac{4s+3}{3s+4}. \tag{12}$$

The results for the small buffer case show that for relatively coarse-grained sources the input and output ratios are different. The input ratio is too high, as compared to the function $s^{-0.85}$, while the output ratio is too low. By increasing $L$ the two ratios seem to converge to, approximately, $s^{-0.87}$, which is close to the result of [2]. Note that, as observed in [3], (12) approximates $s^{-0.85}$ very well. Clearly, the graphs of the output ratios lie below the graphs of the input ratios, implying that the output ratios are more fair than the input ratios. We are unable to provide intuition as to why the output ratio is more fair than the input ratio. Observe also that the sharing of the link becomes more fair when the buffer size increases ($\alpha$ drops to approximately $0.65$), which is in accordance with intuition.

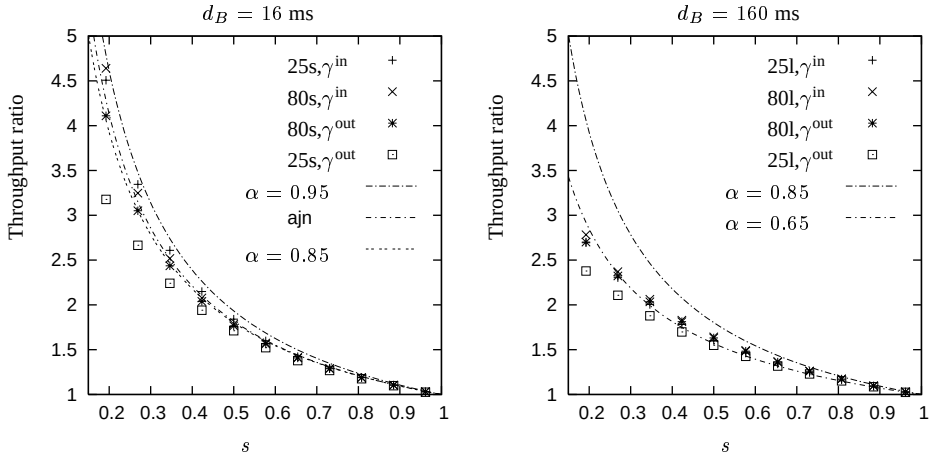



Fig. 3. Ratio of throughputs as a function of $s = T_1/T_2$ for the proportional loss model. The left (right) panel shows the ratio when $d_b = 16$ ms ($d_b = 160$ ms). (The label '25s,$\gamma^{\text{in}}$' refers to the input-related throughput (8) computed for Scenario 25s, etc.; the label 'ajn' refers to (12).)

We explain the impact of the choice for $L$ on these two ratios by considering the overload states. Suppose first that just source 1 is in state 26 when congestion occurs. When $L = 25.7$ source 1 needs $13$ round-trip times after a loss before it can fill the link by itself, whereas when $L = 80.7$, and source 1 is in state 81 it needs $41$ round-trip times. However, the congestion duration is in both cases one round-trip time. Thus, applying this insight to the situation with two sources, the fraction of time spent in congestion, i.e., $C = K$ and $\beta > 0$, is smaller when $L = 80.7$ than when $L = 25.7$ (using the scaling of the other parameters as explained in Section 3.4). As the computations of $\gamma_i^{\text{out}}$ and $\gamma_i^{\text{in}}$ mainly differ when $C = K$, and the fraction of time in congestion is less when $L = 80.7$ as compared to $L = 25.7$, the difference between $\gamma_i^{\text{out}}$ and $\gamma_i^{\text{in}}$ is smaller when $L = 80.7$.

In Figure 4 we plot similar results but now for the synchronous loss model. There is hardly any difference between the input and output ratios. For small buffers we see that the ratios according to our model behave like $s^{-2.2}$ instead of $s^{-2}$ as obtained by [18]. When the buffer size increases, the power decreases to a value smaller than $2$, in line with the results of [18].

In Figure 5 we compare the utilization $(\gamma_1^{\text{out}} + \gamma_2^{\text{out}})/L$ as computed by our model against simulation of two New-Reno sources and two TCP Sack sources, and theoretical results of [18] and [3]. In the synchronous loss model the authors of [18] estimate the utilization as $3/4$ independent of the ratio $s$. For the proportional model the authors of [3] provide in their Equation (23) the approximation

$$\frac{\gamma_1}{L} \approx \frac{2s^2-1}{2(s^2-1)}\frac{1}{s+1} - \frac{1}{4(s^2-1)} = \frac{1}{4}\frac{4s+3}{(s+1)^2}. \tag{13}$$

Combining this with (12) yields a similar expression for $\gamma_2/L$.

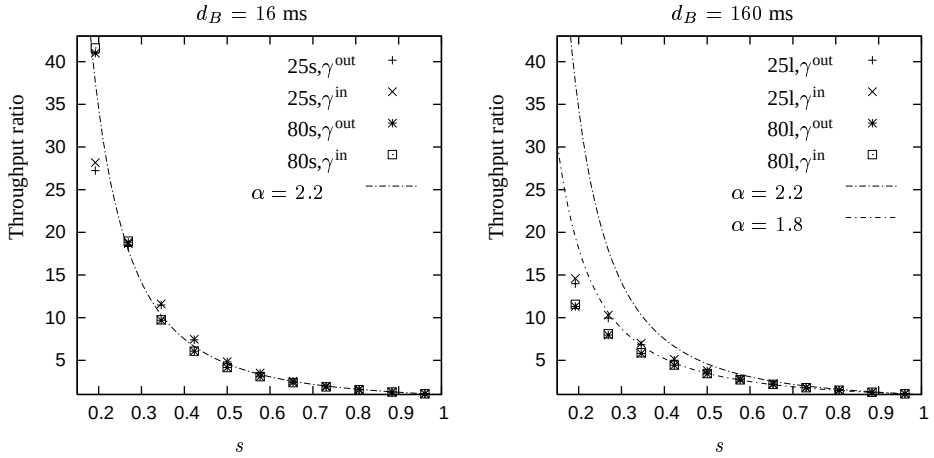



Fig. 4. Ratio of throughputs for the synchronous loss model as a function of $s = T_1/T_2$. The labeling is as in Figure 3.

We see from the graphs that most models overestimate the utilizations in comparison to simulation. Our model, contrary to (13), correctly captures the trend that the utilization decreases as a function of $s$. Note that the proportional loss model is the more appropriate model as we use a RED queue in the simulations. We include the results for the synchronous case mainly for reference. In the right panel, showing the results for large buffers, we do not include the results of [3] and [18] as these only apply to small buffers. Interestingly, in line with an observation in [2], the utilization in case of proportional loss is higher than the utilization in case of synchronized loss. Finally, we conclude that the theoretical models are too optimistic about link utilization in all cases. (The results of the TCP New-Reno simulation in the right panel are a bit odd. This behavior did not disappear by slight changes of the parameters of the RED buffer. We did not investigate large changes as this would introduce considerable differences between the model and the simulation.)

Figure 6 shows the normalized throughput of the first connection $\gamma_1^{\text{out}}/(\gamma_1^{\text{out}} + \gamma_2^{\text{out}})$ in comparison to (13) and the simulations; the results for the second connection follow immediately, as $\gamma_2^{\text{out}} = 1 - \gamma_1^{\text{out}}/(\gamma_1^{\text{out}} + \gamma_2^{\text{out}})$. Clearly, the theoretical ratios are in nearly perfect agreement. Moreover, for the small buffer case the proportional loss models are 'too fair', while the synchronous models are 'too unfair', which is in line with the findings in [16].

## 5 Extensions

The extensions presented in this section provide further support for the versatility of applying SPNs to modeling TCP. We start with two relatively simple extensions of the model of Section 3. The first allows sources to switch on (e.g. 'downloading') and off (e.g. 'thinking'); the second is such that multiple sources can share the

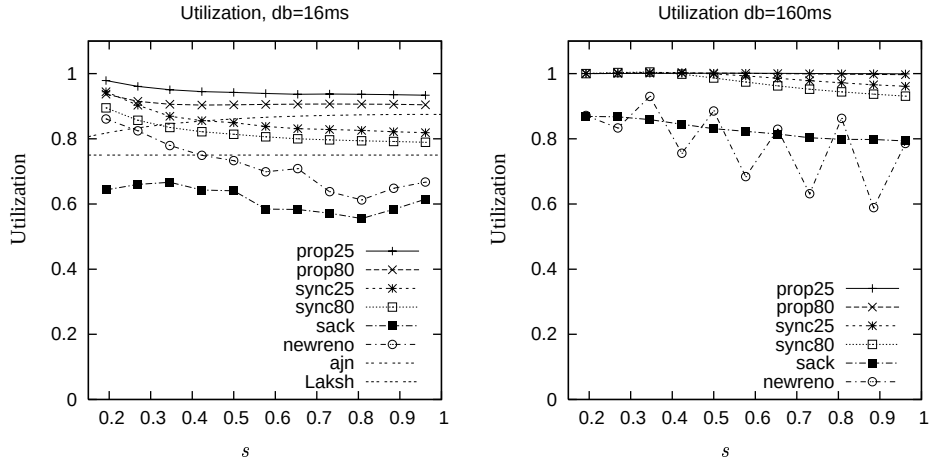



Fig. 5. The utilization as a function of the ratio of propagation delays. The label 'ajn' refers to (13) as obtained by [3], while 'Laksh' labels the line $3/4$ which is the estimate obtained by [18]. The label 'prop25' in the left (right) panel refers to scenario 25s (25l) of the proportional model, etc.

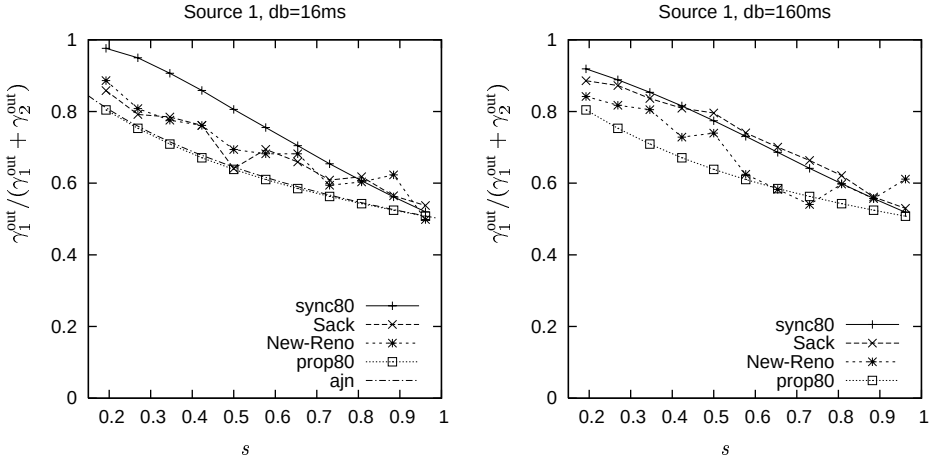



Fig. 6. The normalized throughput as a function of $s$.

bottleneck link. Then we specify and analyze a network consisting of three sources and two links. Especially this last model appears difficult to tackle 'by hand'.

### *5.1 Multiple Sources and On/Off Behavior*

Figure 7 specifies a source that can switch on and off. The extension of the source subnet consists of adding two transitions `tOn` and `tOff` that fire at rate $\lambda_{\text{on}}$ and $\lambda_{\text{off}}$. The probability that the on-time exceeds $x$ is $\exp(-\lambda_{\text{on}}x)$. We take the file sizes as exponentially distributed with average size $\mathbb{E}\{\text{file size}\}$. Consequently, the rate at which the source switches off is given as $\lambda_{\text{off}} = r(k)n/\mathbb{E}\{\text{file size}\}$, if $C = k$ and $W = n$.

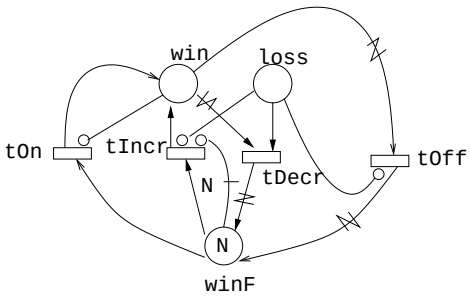



Fig. 7. An on/off source. (We do not draw the arcs that connect the source subnet to the buffer subnet.)

Suppose the source is off. Then, clearly, all its window tokens should be positioned in `winF` and the marking $W$ of `win` should equal 0. The inhibitor from `winF` to `tIncr` with multiplicity $N$ disables `tIncr` as long as $W = 0$. Thus, the only possibility to move a token from `winF` to `win` is the transition `tOn`. As soon as `win` contains one token, the inhibitor to `tOn` disables this transition. The source switches off when `tOff` removes all tokens at `win` via the variable arc from `win` and adds these tokens to `winF`. When the source is in a loss state, i.e., `loss` is marked, it cannot finish a file transfer. The inhibitor from `loss` to `tOff` prevents this. Note that this implementation of on/off behavior does not come at the cost of extra places. Hence, the set of markings $\mathscr{M}$ does not increase.

We refer the reader to [16] for an analysis of the impact of on/off behavior on the sharing and utilization of the link.

Extending the SPN of Section 3 to incorporate more than two sources is quite simple. SPNP supports arrays of places, transitions, etc. The window size $W_i$ of source $i$ corresponds then to the value of the $i$-th element of the 'window' array, etc. The size of the arrays equals the number of sources, which can be controlled, clearly, by a single variable. For the synchronous loss model the number of loss tokens initially present at `lossB` should of course equal the number of sources.

### *5.2 Three TCP Sources sharing Two Buffers*

In this section we extend the model to a network consisting of three sources and two buffers in a configuration as shown in Figure 8. We explain the SPN in which the buffers use a proportional loss scheme, define the performance measures and present some results.

The SPN for the network is shown in Figure 9. The subnets for sources 1 and 2 and the buffers $B_1$ and $B_2$ are identical to their counterparts of Section 3.2. Source 0, as shown by the middle, lower subnet in Figure 8, is different in that its connection uses both buffers. We elaborate on this now. To avoid tedious repetition we do not formally introduce variables such as $L_1, B_1$, etc., when the meaning is obvious.

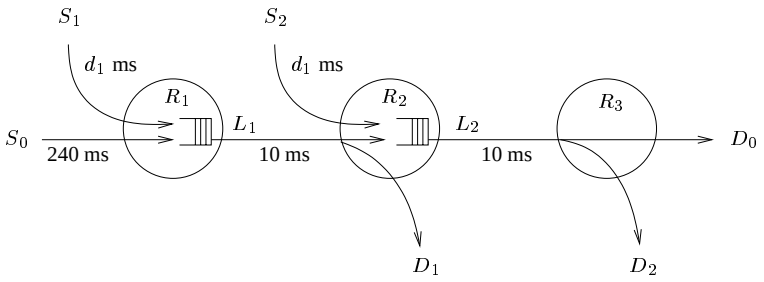



Fig. 8. A network of three sources sharing the links between routers $R_1$, $R_2$, and $R_3$. Router $R_1$ ($R_2$) contains the first (second) shared buffer in front of the link $L_1$ ($L_2$).

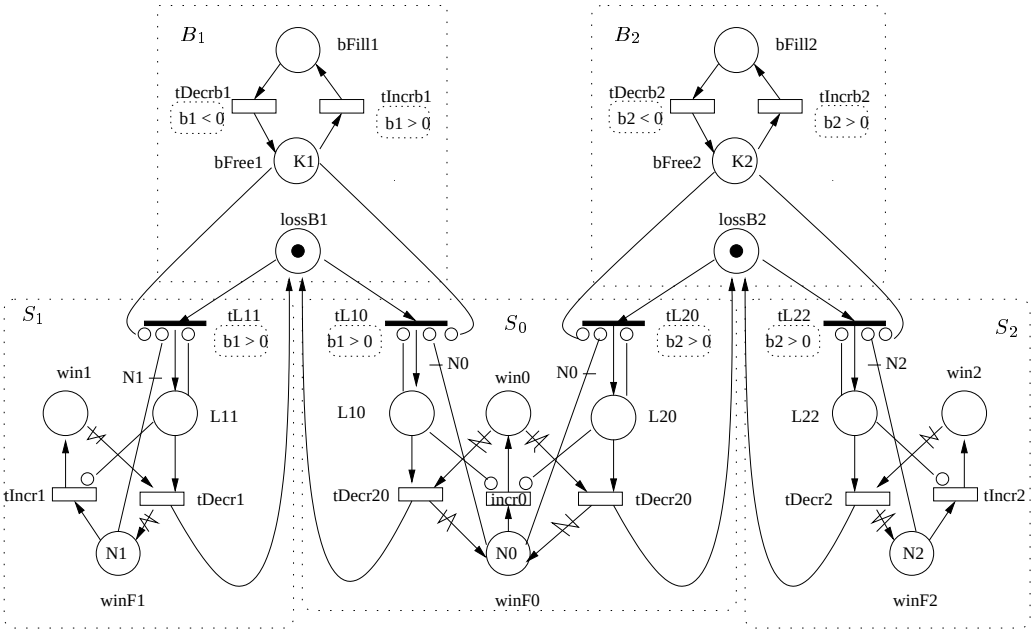



Fig. 9. TCP source 0 uses buffers 1 and 2, while source 1 (2) uses buffer 1 (2). Note that in Figure 1 we used rather descriptive names for the transitions. In the present case we turned to shorter, but less descriptive, names for presentational reasons.

The average round-trip time of source 0 is

$$T_0(\mathbf{k}) = T_0 + \frac{k_1}{K_1}\frac{B_1}{L_1} + \frac{k_2}{K_2}\frac{B_2}{L_2},$$

where we write $\mathbf{k} = (k_1, k_2)$. Thus, the analog of (5) becomes

$$r_0(\mathbf{k}) = \frac{\mathrm{MSS}_0}{T_0(\mathbf{k})}.$$

With respect to the loss model we see that Source 0 can receive a loss token from both buffers. A consequence of this is that Source 0 can have both loss tokens in possession simultaneously. As such it suffers from loss twice within one round-trip time, i.e., within one window of data, and reduces its rate twice accordingly. This is inconsistent with the behavior of TCP New-Reno or TCP Sack which mostly

decrease only once even when more than one packet of a window of data are lost. We contend that this undesirable side effect has small impact. First, for this event to happen, congested periods of both buffers have to overlap. Second, source 0 should receive the loss token of both buffers. As source 0 is usually sending at a lower rate than source 1 and 2, both conditions will not often be satisfied simultaneously.

The functions $\beta_1$ and $\beta_2$ should also be adapted to the network environment. It is clear that

$$\beta_1(\mathbf{n},\mathbf{k}) = \frac{K_1}{B_1}\left(r_0(\mathbf{k})n_0 + r_1(k_1)n_1 - L_1\right).$$

To obtain a similar expression for $\beta_2$ we should account for the fact that the first buffer shapes the output process of source 0. We approximate the output rate, $\delta_0$ say, of source 0 at the first buffer similarly to (9):

$$\delta_0(\mathbf{n},\mathbf{k}) = \begin{cases} r_0(\mathbf{k})n_0, & \text{if } k_1 = 0, \\ L_1 \frac{r_0(\mathbf{k})n_0}{r_0(\mathbf{k})n_0 + r_1(k_1)n_1}, & \text{if } k_1 > 0. \end{cases}$$

Now we can define $\beta_2$ as

$$\beta_2(\mathbf{n},\mathbf{k}) = \frac{K_2}{B_2}\left(\delta_0(\mathbf{n},\mathbf{k}) + r_2(k_2)n_2 - L_2\right).$$

For reasons of consistency with the definition of $\delta_0$ above, we use the output-related definitions of throughput as in (9). Thus, omitting the superscript 'out',

$$\gamma_0(\mathbf{n},\mathbf{k}) = \begin{cases} \delta_0, & \text{if } k_2 = 0, \\ \frac{\delta_0}{\delta_0+\Delta_2}L_2, & \text{if } k_2 > 0, \end{cases}$$
$$\gamma_1(\mathbf{n},\mathbf{k}) = \begin{cases} \Delta_1, & \text{if } k_1 = 0, \\ \frac{\Delta_1}{\Delta_0+\Delta_1}L_1, & \text{if } k_1 > 0, \end{cases}$$
$$\gamma_2(\mathbf{n},\mathbf{k}) = \begin{cases} \Delta_2, & \text{if } k_2 = 0, \\ \frac{\Delta_2}{\delta_0+\Delta_2}L_2, & \text{if } k_2 > 0, \end{cases}$$

where

$$\Delta_0 \equiv r_0(\mathbf{k})n_0, \qquad \Delta_1 \equiv r_1(k_1)n_1,$$
$$\Delta_2 \equiv r_2(k_2)n_2, \qquad \delta_0 \equiv \delta_0(\mathbf{n},\mathbf{k}).$$

With this we set

$$\gamma_i = \mathbb{E}\{\gamma_i(\mathbf{W},\mathbf{C})\}, \quad i = 0,1,2, \tag{14}$$

where $\mathbf{W} = (W_0, W_1, W_2)$ and $\mathbf{C} = (C_1, C_2)$. Finally, the utilizations for the first and second link become, respectively,

$$u_1 = \frac{\gamma_0 + \gamma_1}{L_1} \qquad u_2 = \frac{\gamma_0 + \gamma_2}{L_2}.$$

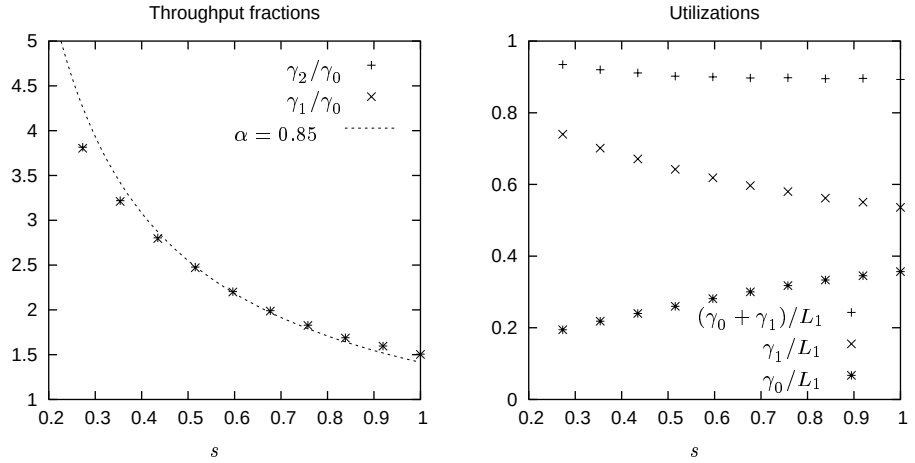



Fig. 10. The throughput ratios for both buffers and utilization for the first buffer as functions of $s = T_1/T_0 = T_2/T_0$.

We compute $\gamma_0$, etc., for this model. Here the second buffer is identical to the first, i.e., $L_2 = L_1 = 25.7$ and $d_{B_2} = d_{B_1} = 16$ ms, as in Scenario 1 of Table 4. We vary the propagation delay $d_1$ of the links connecting sources 1 and 2 to the routers $R_1$ and $R_2$, respectively, from $40$ ms to $250$ ms simultaneously in ten steps. The left panel of Figure 10 shows the ratios of the throughputs $\gamma_1/\gamma_0$ and $\gamma_2/\gamma_0$ as functions of $s = T_1/T_0 = T_2/T_0$. We see that the throughputs of source 1 and 2 are nearly the same. This is to be expected when the fraction of lost traffic at the first buffer is small. Indeed, in that case the rate of the 'thinned connection 0', i.e. the traffic of connection 0 minus the loss incurred at the first buffer, is nearly the same as the transmission rate of source 0. Hence, connection 1 and connection 2 have to compete with approximately the same connection. The fact that $\gamma_2$ is just slightly larger than $\gamma_1$ shows, in accordance with the above, that the rate of the thinned connection 0 is a bit smaller than its initial rate. As the difference between $\gamma_1$ and $\gamma_2$ is small, we neglect this aspect in the rest of the discussion.

When $T_0 = T_1 = T_2$ we can compare $\gamma_1/\gamma_0$ to some theoretical fairness results for networks as derived by [21]. The authors of [19] claim that, in the terminology of [21], the bandwidth sharing obtained by TCP in networks results in minimum-potential-delay fairness. When we apply these results to the network shown in Figure 8 we obtain that, theoretically, $\gamma_1/\gamma_0 = \sqrt{2}$. Our model, on the other hand, gives $\gamma_1/\gamma_0 = 13.81/9.21 = 1.5$, which is quite near to $\sqrt{2}$. Interestingly, in Figure 10 we plot as a reference the function $s \to \sqrt{2}s^{-0.85}$. This shows considerable agreement to the numerical results. It seems that the power of $s$ is dictated by the loss model while the pre-factor is determined by the topology. We defer an investigation of this result to future research.

## 6 Summary

We use stochastic Petri nets to specify, in a versatile way, Markovian models of TCP New-Reno or Sack (more specifically, AIMD) sources that share one or two buffers. The first model contains two connections competing for a single bottleneck link and buffer. The second model describes one connection traversing two consecutive buffers, while each buffer receives additional side traffic from other TCP connections. We also show that the first model can be simply extended to more than two sources, and present a modification of the source model to include on/off behavior.

This methodology is flexible, extendable, and enables to obtain qualitative insight into the impact of various source and network parameters on transient and long-term properties such as source throughput, link utilization and fairness. With respect to parameters as packet size, round-trip time, and buffer size, the results of our model are consistent with those of earlier models, e.g. [6,16], and therefore not reported here.

In the first model (two sources, one buffer) we implement two popular assumptions about the loss process at the buffer, viz. proportional loss and synchronized loss. We validate the Markovian models for either loss process by extensively comparing it, on the one hand, to the theory developed in [2],[3] and [18], and, on the other, to simulation by ns-2. The models provide results that are consistent with the theoretical results or improve these in that better resemblance is found to simulation results obtained by ns-2.

The second model (three sources, two buffers) shows that when the round-trip times of all connections are equal, the computed 'fairness' is approximately minimum-potential-delay fair as defined in [21]. It would be interesting to investigate the type of fairness in case buffer sizes are not small or when the round-trip times differ. To the best of our knowledge, the approach based on SPNs is one of the few theoretical approaches that enables such quantitative analysis. In [21] the impact of round-trip time differences is considered, but the window control is non-adaptive contrary to our source model.

We finally mention that specifying the Markovian models by means of SPNs, so that the generator of Markov chain and the performance measures are computed automatically, has some noteworthy advantages over implementing the generator by hand as is done in [16]. The implementation of the SPNs is straightforward and less error-prone. Moreover, the automatically generated Markov chains usually need less states, and can therefore be solved more efficiently. Finally, it is easy to include complex behavior of the application layer or modify aspects of TCP in the SPN. In summary, we feel that using SPNs shifts the burden of the work from simple but awkward programming to the more attractive task of designing a Petri

net that behaves according to a set of pre-specified rules.

**Acknowledgments**

To obtain the performance results in this paper we made extensive use of the software package SPNP version 4. The authors thank Kishor S. Trivedi of Duke University for making this package available. The authors are grateful to Chadi Barakat for sharing the ns-2 script he used to obtain the results of [2]. Finally, the first author thanks Pasi Lassila for some help with the simulations with ns-2.

## A Some Concepts of Stochastic Petri Nets

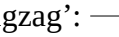
In this section we introduce the concepts of stochastic Petri nets that are relevant to this paper. We refer to Figure 1 as an example.

A SPN consists of a set of *places* and a set of *transitions*. These two sets are connected via *directed* arcs as a bipartite graph: places (drawn as circles) connect only to transitions, while transitions (drawn as bars) connect only to places. A directed arc from a place (transition) to a transition (place) is called an *input* (*output*) arc to (from) a transition. Places can contain tokens indicated as a number of black dots or an integer in the place. In case a place contains at least one token, we say that it is *marked*. The distribution of the tokens over the places represents the state of the net and is called the *marking*. When *all* input places, i.e., all places connected to the input arcs of a transition, are marked the transition is *enabled*. Once enabled, the transition can *fire*, thereby removing tokens from its input places and adding tokens to its output places. Thus, a firing nearly always changes the marking. These firings are to occur immediately (contrary to timed transitions to be introduced below) and atomically, i.e., other transitions cannot fire before the action of the firing transition is completed. Note that during firing 'conservation of tokens' is *not* necessarily implied.

Starting from an *initial* marking $M_0$, the *reachability set* $\mathscr{M}$ is the set of all different markings $M$ reachable by any succession of enabled transitions starting from $M_0$.

Besides the input and output arcs just mentioned, a Petri net can contain *inhibitor* arcs, to be drawn as an arc from a place to a transition with as arrowhead a small circle. If the place connected to an inhibitor arc is marked, the related transition is disabled.

An important property of input, output, and inhibitor arcs is their *multiplicity*. A *multiple* input (output) arc removes (adds) a number of tokens according to its

multiplicity from (to) a place, provided it is enabled. Note that the transition is only enabled if the number of tokens at each input place is larger than or equal to the multiplicity of the corresponding input arc. A multiple inhibitor arc becomes effective as soon as the place contains a number of tokens at least as large as the inhibitor's multiplicity. Besides multiple arcs we need *variable* in- and output arcs. The multiplicity of these arcs may depend on the actual marking of the net. Thus, the multiplicity of variable arcs is generally not constant. Variable arcs are shown as directed arcs with a 'zigzag':  .

Sometimes it is desirable to incorporate probabilistic behavior in the net. One mechanism for this—the other mechanism is related to time, which will be discussed presently—is a *random switch*. Such a switch consists of a set of immediate transitions which are all simultaneously enabled by the same marking. A set of weights is adjoined to the random switch. The probability that a certain transition of the random switch fires is proportional to its weight. Such weights are allowed to depend on the marking at the moment just prior to firing.

The arc types we have discussed above permit us to specify various types of conditions to enable or disable transitions. However, sometimes it is rather cumbersome to specify complicated conditions in the SPN by means of places and arcs. To avoid such awkward complications we can use *guards*. A guard is a marking-dependent enabling function attached to a transition. If the condition of the guard is satisfied, the transition is enabled; otherwise the transition is disabled. Thus, by means of guards quite complex marking dependent conditions can be imposed on the dynamics of the net. In this paper we draw a guard as a box with a dashed boundary containing a (shorthand of a) condition. (This graphical representation of a guard is by no means standard in the literature.)

Up to now the transitions discussed above are *immediate*: if a transition is enabled, and chosen when it is an element of a random switch, it fires immediately. We can introduce the concept of time in the Petri net by means of *timed* transitions, which are drawn as open rectangles. Such a transition fires, if enabled, after an exponentially distributed amount of time. A useful feature is that the transition rates of such transitions are allowed to depend on the marking.

Once we have specified the SPN the computation of performance measures is relatively straightforward. Under some mild boundedness conditions, it is possible to automatically map the SPN to a continuous-time Markov chain $\{M(t), t \geq 0\}$ with infinitesimal generator $Q$ and initial probability vector representing the initial marking of the SPN. The size of the chain equals the cardinality $|\mathscr{M}|$ of the reachability set $\mathscr{M}$. If $\{M(t)\}$ is irreducible, the stationary distribution $\boldsymbol{\pi} = (\pi_0, \ldots, \pi_{|\mathscr{M}|})$ exists and does not depend on the initial marking. The vector $\boldsymbol{\pi}$ satisfies $\boldsymbol{\pi}Q = 0$, $\sum_i \pi_i = 1$, and can be computed by Gauss-Seidel iteration, or other, more advanced, numerical procedures, cf. [26].

The performance measures of interest for the stationary limit $M$ of $\{M(t)\}$ can then be expressed in terms of a reward rate function $r : \mathscr{M} \to \mathbb{R}$ which associates with every state $m \in \mathscr{M}$ a real-valued reward rate $r(m)$. The steady-state reward is then given as

$$\mathbb{E}\{r(M)\} = \sum_{m \in \mathscr{M}} r(m)\pi_m. \tag{A.1}$$

For more information regarding SPNs consult, e.g., [1]. For details concerning SPNP, see [11].

## References


[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, 1995.

[2] E. Altman, C. Barakat, E. Laborde, P. Brown, and D. Collange. Fairness analysis of TCP/IP. In *Proc. of IEEE Conference on Decision and Control*, 2000.

[3] E. Altman, T. Jimenez, and R. Núñez-Queija. Analysis of two competing TCP/IP connections. *Performance Evaluation*, 49:43–55, 2002.

[4] K. Avratchenkov, U. Ayesta, E. Altman, P. Nain, and C. Barakat. The effect of router buffer size on the TCP performance. In *LONIIS workshop on Telecommunication Networks and Teletraffic Theory*, 2002.

[5] F. Baccelli and D. Hong. Flow level sumulation of large IP networks. In *Proc. of IEEE INFOCOM*, 2003.

[6] P. Brown. Resource sharing of TCP connections with different roundtrip times. In *Proc. of IEEE INFOCOM*, pages 1734–1741, 2000.

[7] C. Casetti and M. Meo. A new approach to model the stationary behavior of TCP connections. In *Proc. of IEEE INFOCOM*, pages 367–375, 2000.

[8] C. Casetti and M. Meo. An analytical framework for the performance evaluation of TCP Reno connections. *Computer Networks*, 37(5):669–682, 2001.

[9] C. Casetti and M. Meo. Modeling the stationary behavior of TCP Reno connections. In *QOS-IP*, pages 141–156, 2001.

[10] D.H. Chiu and R. Jain. Analysis of the increase and decrease algorithms of congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17:1–14, 1989.

[11] G. Ciardo, G. M. Fricks, J.K. Muppala, and K.S. Trivedi. *SPNP Users Manual*, 4th edition, 1994.

[12] G. Ciardo, G. Muppalla, and K. Trivedi. SPNP: Stochastic Petri Net Package. In *3rd Int. Workshop on Petri Nets and Performance Models (PNPM'89)*, pages 142–151. IEEE Comp. Soc. Press, 1989.

[13] S. Floyd and T. Henderson. The NewReno modification TCP's Fast Recovery algorithm, 1999. RFC 2582.

[14] S. Floyd and V. Jacobson. Random Early Detection gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[15] N.D. van Foreest, M.R.H. Mandjes, and W.R.W. Scheinhardt. Analysis of a feedback fluid model for heterogeneous TCP sources. *Stochastic Models*, 19(3):299–324, 2003.

[16] N.D. van Foreest, M.R.H. Mandjes, and W.R.W. Scheinhardt. A versatile model for asymmetric TCP sources. In *Proc. of ITC 18*, pages 631–640, 2003.

[17] R.J. Gibbens, S.K. Sargood, C. Van Eijl, F.P. Kelly, H. Azmoodeh, R.N. Macfadyen, and N.W. Macfadyen. Fixed-point models for the end-to-end performance analysis of IP networks. In *ITC Specialist Seminar: IP Traffic Measurement, Modeling and Management*, volume 13, 2000.

[18] 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):336–350, 1997.

[19] K.W. Lee, T.E. Kim, and V. Bharghavan. A comparison of end-to-end congestion control algorithms: the case of AIMD and AIPD. In *Globecom*, 2001.

[20] Y. Liu, F. Lo Presti, V. Misra, D. Towsley, and Y. Gu. Fluid models and solutions for large-scale IP networks. *ACM SIGMETRICS Performance Evaluation Review*, 31(4):91–101, 2003.

[21] L. Massoulié and J.W. Roberts. Bandwidth sharing: objectives and algorithms. In *Proc. of IEEE INFOCOM*, pages 1395–1403, 1999.

[22] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgment options, 1996. RFC 2018.

[23] V. Misra, W. Gong, and D. F. Towsley. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. *ACM SIGCOMM Computer Communication Review*, 30(4):151–160, 2000.

[24] L.L. Peterson and B.S. Davie. *Computer Networks*. Morgan Kaufman Publ., 2nd edition, 2000.

[25] Network Simulator. Available at: `http://www.isi.edu/nsnam/ns/`.

[26] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.