Available online at www.sciencedirect.com

ScienceDirect

Performance Evaluation 64 (2007) 507–523

PERFORMANCE EVALUATION
An International Journal

www.elsevier.com/locate/peva

Versatile stochastic models for networks with asymmetric TCP sources

Nicky D. van Foreest[a], Boudewijn R. Haverkort[a,*], Michel R.H. Mandjes[b,a,1], Werner 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*

Received 15 March 2006
Available online 27 September 2006

Abstract

In this paper we use stochastic Petri nets (SPNs) to study the interaction of multiple TCP sources that share one or two buffers. No analytical nor numerical results have been presented for such cases yet. We use SPNs in an unconventional way: the tokens in the SPN do not represent the packets being sent in the network, but merely model fractions of buffer occupancy and the congestion window sizes. In this way, we use the SPNs to obtain a discretisation of a fluid model for TCP dynamics. Thus, we pair the modelling flexibility of SPNs with the modelling efficiency of fluid models. In doing so, our approach also avoids the (numerical) solution of partial differential equations; instead, just the steady-state solution of a (large) continuous-time Markov chain is required.

We first consider two TCP sources sharing a single buffer and evaluate the consequences of two popular assumptions for the loss process in terms of fairness and link utilization. The results obtained with this model are in agreement with existing analytic models. A comparison with (more costly) simulations in ns2 shows that the real loss process is somewhere in between the two loss models.

Secondly, we consider a network consisting of three sources and two buffers and study how the sources share the capacity of the links. This leads to an interesting conjecture on fairness in large TCP networks.

Keywords: TCP; Fluid-flow model; Stochastic Petri nets; Fairness analysis

1. Introduction and related work

Many models of the interaction between TCP sources and buffers have been developed over the last few years. These models are used to obtain insight into how efficient and fair 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 a single bottleneck link. Other models, e.g., [23,26], are concerned with the performance of large networks as a whole. The latter models are, with respect to the analysis, completely deterministic.

* Corresponding author.
E-mail address: brh@cs.utwente.nl (B.R. Haverkort).
[1] Current position at Korteweg-de Vries Instituut, Universiteit van Amsterdam, Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands.

As yet, there is no model that applies well to networks of *intermediate size*, i.e., networks consisting of a few sources and a few buffers (also called nodes), while retaining a stochastic flavour. This stochastic nature is arguably not of much importance in large networks where the presence of a single flow is hardly noticeable, however, in intermediately-sized networks this *is* important, since one connection can add considerably to congestion. Hence, it is of importance 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 [18] for two TCP sources interacting with a buffer is simple, yet flexible, and can in principle be extended to networks with a few sources and buffers. It has been formulated directly in terms of a continuous-time Markov chain and allows the study of various stochastic aspects of the interaction. However, its use is limited in practice since the generator matrix of the Markov chain has to be constructed manually. A suitable framework to extend this model is provided by stochastic Petri nets (SPNs), see e.g., [1,11]. Note, however, that we do not use SPNs in a conventional way. In our models, the SPN tokens do not coincide with packets (often called segments in TCP context), but rather represent, in an abstract way, (partial) buffer occupancies and congestion window sizes. Hence, we use the SPN to describe a stochastic approximation to a fluid model of TCP dynamics. This approach has not been pursued before. Furthermore, it is very efficient, in that it resorts to a steady-state analysis of a finite Markov chain to evaluate the TCP dynamics, instead of having to solve the system of partial differential equations for the fluid model. It is noteworthy that similar discretisation approaches have recently also been applied successfully for the evaluation of signal transduction pathways in biochemical systems [6]; it is interesting to observe that in this (biochemical) context, similar advantages of the approach are reported.

In the current work we apply SPNs to study the interaction between multiple TCP sources and buffers in intermediately-sized networks. With this approach we generalize the TCP models of [2,3] and [18] considerably in that we handle larger buffers and multi-node networks. Especially the latter aspect seems difficult to incorporate in the setting of [2,3]. In contrast to [26], in which the analysis is deterministic, our model is entirely stochastic and, therefore, allows a more realistic modelling of workload and system modelling (and their interaction). Another advantage of our approach is that we can easily compute higher moments of measures of interest.

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. Using fixed-point methods, see, e.g., [19] or [4], they compute performance measures. In [8] and [9], the 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 developed in [26], we start by summarizing this approach in Section 2. Then, in Section 3, we specify an 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 [3]. The results, as presented in Section 4, convincingly show that the key aspects of earlier models are captured very well. In Section 5 we then present an important extension of the model, in that we consider a network consisting of three sources and two nodes. We compare the sharing of link capacity to theoretical results. This provides us with an interesting conjecture on how to estimate the fairness in more complicated network topologies.

Section 6 concludes the paper.

2. Background: A fluid model for TCP

Here we summarize the fluid model [26] for a network of J greedy TCP sources that share a buffer of size B and a link with capacity (or speed) L. The buffer uses a Random Early Detection (RED) packet dropping scheme [15]. At the end of this section we point out two shortcomings of this model and clarify how our approach circumvents these.

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,

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

is the round-trip time of source i when the buffer content is $\bar{Q}(t)$ at time t; the addend T_i accounts for the propagation delay in an otherwise empty system, and $\bar{Q}(t)/L$ accounts for the transmission delay at the node.

Source i maintains a window variable $\bar{W}_i(t)$, supposed to be continuous, and sends fluid at the rate $\text{MSS}_i \bar{W}_i(t)/T_i(\bar{Q}(t))$, where MSS_i is the message segment size (or packet size) into the RED buffer. The window dynamics behaves according to the Additive-Increase/Multiplicative-Decrease (AIMD) scheme as described in [10]. More specifically, the change to the window size is governed by the differential equation

$$\mathrm{d}\bar{W}_i(t) = \frac{\mathrm{d}t}{T_i(\bar{Q}(t))} - \frac{\bar{W}_i(t)}{2}\mathrm{d}M_i(t), \tag{1b}$$

that can be explained as follows. The first term of the right-hand side of this equation corresponds to the Additive-Increase behavior of a source; if $T_i(\bar{Q}(t))$ is the round-trip delay, its reciprocal value is the frequency with which acknowledgments are received, hence, the window is increased. The second term implements the Multiplicative-Decrease at a loss epoch, where $M_i(t)$ models the losses as a point process, so that $\mathrm{d}M_i(t) = 1$ when a loss occurs and 0 elsewhere. Hence, whenever a loss occurs, the window size is halved.

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

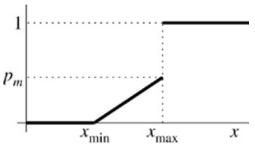
$$\frac{\mathrm{d}\bar{Q}(t)}{\mathrm{d}t} = \sum_{i=1}^{J} \frac{\text{MSS}_i \bar{W}_i(t)}{T_i(\bar{Q}(t))} - L, \tag{1c}$$

with $\bar{Q}(t) \in (0, B)$, otherwise $\mathrm{d}\bar{Q}(t)/\mathrm{d}t = 0$.

The RED buffer operates as follows; consult, for instance, [27] for a more detailed description. 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 the ith estimate for x is obtained as

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

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



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

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

To analyze the differential equation system (1) and (2), the authors of [26] take expectations at both sides of (1) and (2) and make a number of simplifying assumptions to obtain a numerically tractable system of differential equations, which they solve using Matlab to obtain the expected transient behavior of, for instance, the queue.

The main advantages of the above approach are its flexibility and scalability. Still, two fundamental problems do exist with it. 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. This is, however, an important point to incorporate if the behavior of a source depends on its actual state. For instance, the rate at which file transfers stop, depends on the actual transmission rates of the sources; the higher the rate, the sooner the transfer is ready. Thus, the rate at which a source changes from an on-state to an off-state may depend on the source's actual transmission rate. The second problem relates to a more technical aspect of the analysis. In [26] there is no mention of how to *compute* the expectations that are taken. Technically speaking, 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 the second point, we also take expectations, but with respect to a stationary distribution of a Markov chain, so that no problems about the interpretation remain. Loosely speaking, we *first* solve the system and *then* take expectations, whereas the authors of [26] take expectations first and then solve the system. Reversing this ordering is not a mere technicality, due to the fact that the expectation operator is a non-linear operator. A disadvantage of our approach as compared to [23,26] is that our model does not scale as well as theirs does.

3. An SPN-based TCP fluid model

In this section we introduce our SPN-based model; we assume familiarity with SPNs. Our key idea is to discretise the source window and queue processes, which is described in Section 3.1. In Section 3.2 we then present a two source, single buffer model as an SPN, where we assume a so-called proportional loss model. In Section 3.3 we then define the measures of interest (throughputs and utilization). Then, in Section 3.4 we extend our model to the so-called synchronized loss case. We briefly touch upon issues of computational complexity in Section 3.5.

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 [14] or TCP Sack [25], that does not (frequently) resort to timeouts and slow starts when multiple losses occur in one window.

3.1. Discretising source and buffer model

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 $\{Q(t), t \geq 0\}$ with state space $\{0, 1, \ldots, K\}$; notationally, we have removed the bar. Observe that K in general does not correspond to the maximum number of packets the buffer can contain, rather it is a discretisation parameter of the buffer. When $Q(t) = k, 0 \leq k \leq K$, the corresponding buffer content in the fluid model, i.e., $\bar{Q}(t)$, would then equal $Q(t)B/K$. Thus, in this case the round-trip time for source i becomes (compare (1a)):

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

Similarly, we discretise the window process of source i to obtain the discrete process $\{W_i(t), t \geq 0\}$ (again, we have removed the bar) with state space $\{0, 1, 2, \ldots, N_i\}$, where N_i denotes the maximum window of source i, which is an important parameter in characterizing the performance of TCP. When the window process at time t for source i takes the value n_i, that is, $W_i(t) = n_i$ and similarly $Q(t) = k$, the source sends traffic at the rate $\text{MSS}_i n_i / T_i(k)$. In the sequel we often use the shorthand

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

so as to write the source rate in that state as $n_i r_i(k)$. Note 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 times t when $(W_1(t), W_2(t)) = (n_1, n_2)$ and $Q(t) = k$ would be given as

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

with $\mathbf{n} = (n_1, \ldots, n_J)$; we use boldface for vectors.

Finally, we introduce the processes $\{l_i(t), t \geq 0\}$, $i = 1, \ldots, J$, as indicators for those sources that have experienced a loss but still need to adapt their rate. 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.

3.2. Two sources and one buffer with proportional loss

In this section we provide a specification of an SPN consisting of two TCP sources sharing one buffer, as shown in Fig. 1. The SPN contains three "subnets" indicated by dashed boxes. The subnets S_1 and S_2 represent the sources, whereas the subnet B represents the buffer. We first describe these subnets, before we focus on the dynamics of the complete SPN.

3.2.1. The subnets S_1, S_2 and B

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.

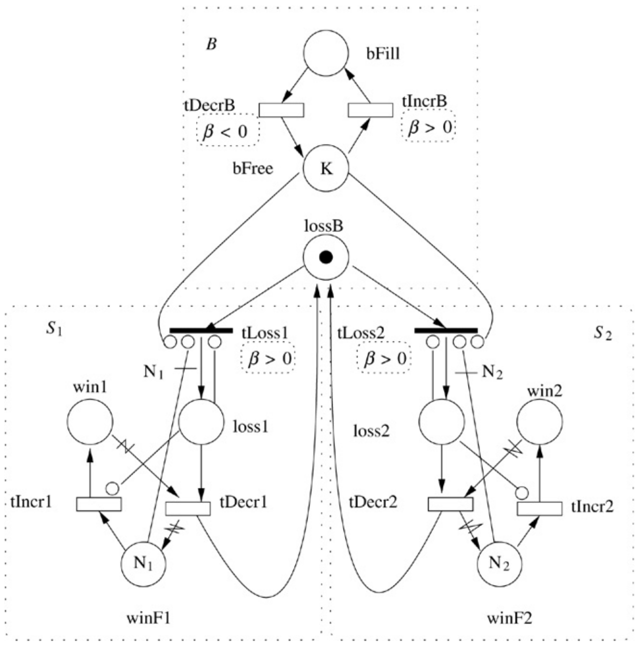


Fig. 1. An SPN model of two TCP sources sharing a buffer. The loss model is taken according to the proportional loss scheme. We indicate guards by means of dashed boxes around strings, such as $\beta < 0$ appearing immediately below `tDecrB`.

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

$W_1(t) \leftrightarrow$ `win1`	$W_2(t) \leftrightarrow$ `win2`	$Q(t) \leftrightarrow$ `bFill`
$l_1(t) \leftrightarrow$ `loss1`	$l_2(t) \leftrightarrow$ `loss2`	

The state of the window in source 1 is given by the markings of `win1`, `winF1` and `loss1`, respectively. Here, the number of tokens in `win1`, models the momentary congestion window of source 1, that is, it describes $W_1(t)$. The marking of `winF1` denotes how much further the window can increase. Initially, we put N_1 tokens in this place, so that at all times during the evolution of the SPN the sum of tokens in `win1` and `winF1` equals N_1. The "loss state" of source 1 is indicated by place `loss1` being marked or not: when there is no token in `loss1` the source is allowed to increase the window size (that is, increase the number of tokens in `win1`), whereas a token in `loss1` means that the source has to reduce its window size by a factor 2. Thus, the number of tokens in `loss1` describes the process $\{l_1(t), t \geq 0\}$. The relation to the places in subnets S_1 and S_2 and the discretised stochastic process is summarized in Table 1.

The buffer subnet B 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, i.e., $Q(t)$. The place `bFree` initially contains K tokens. Its marking corresponds to the free space, i.e., the maximum buffer level K minus the fill level `bFill`; thus, the sum of the number of tokens in `bFill` and `bFree` always equals K. Finally, when there is no token in `lossB` the buffer is congested; when there is a token in `lossB` the buffer is not congested.

The *marking-dependent* firing rates and guards associated with all transitions are summarized in Table 2. Here $T_i(k)$ is given by (4). When positive, 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 increments of the buffer level process occur. This expression can be understood by noting that $\beta^{-1}(n_1, n_2, k)$, the expected time that the process $\{Q(t), t \geq 0\}$ remains in state k before moving to $k + 1$, should equal the expected time that the real fluid content $\bar{Q}(t)$ takes to move from kB/K to $(k + 1)B/K$, which is

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

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`	–	$\beta(n_1, n_2, k) > 0$
`tLoss2`	–	$\beta(n_1, n_2, k) > 0$
`tIncr1`	$T_1^{-1}(k)$	–
`tDecr1`	$T_1^{-1}(k)$	–
`tIncr2`	$T_2^{-1}(k)$	–
`tDecr2`	$T_2^{-1}(k)$	–

Table 3
Firing probability p_1 of `tLoss1` in four possible cases

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

approximately B/K divided by the fluid rate in (6). A similar argument holds for states in which the rate is negative, leading to "downward transitions" at rate $-\beta(n_1, n_2, k)$.

3.2.2. *From initial state to congestion (congestion avoidance)*

The initial marking of the SPN is as shown in Fig. 1. Sources 1 and 2 are not active (their window size is zero) but are allowed to increase their rate. 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 $W_1(t)$ or $W_2(t)$ by one, which models the Additive-Increase phase of TCP. Note that on average source i spends an amount of time $T_i(k)$ in state n_i, given $Q(t) = k$. In this way the SPN incorporates feedback delay.

When $W_1(t)$ and $W_2(t)$ increase, the scaled net input rate (7) increases as well. After a number of firings of `tIncr1` and `tIncr2`, $W_1(t)$ and $W_2(t)$ are so large, i.e., `win1` and `win2` contain so many tokens, 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 $Q(t)$ (number of tokens in `bfill`) by one. After K firings of `tIncrB`, the buffer is completely filled, i.e., $Q(t) = K$. 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 in the synchronous case, to be discussed later), and consequently, `tIncr2` can still fire.

It is evident that whenever source i is inactive, i.e., when $W_i(t) = 0$, it should not 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 $N_1(N_2)$, as indicated in Fig. 1.

3.2.3. *The proportional loss model*

In a proportional loss model, only one connection suffers from loss during overload. The probability of selecting a particular connection is proportional to its momentary transmission rate. We implement this behavior by means of the random switch consisting of `tLoss1` and `tLoss2` in the following way. The marking-dependent weights of the random switch are chosen such that `tLoss1` fires with probability p_1, and `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 following insight. If $r_1(K)W_1 > L$ and $r_2(K)W_2 \le L$, 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 will not receive a loss token in this case. Therefore, in this case, $p_1 = 1$ and $p_2 = 0$. However, in case $r_1(K)W_1 \leq L$ and $r_2(K) \leq L$ (but $\mathbf{r}(K) \cdot \mathbf{W} > L$) both sources can perceive a loss, with a probability proportional to their sending rates. Finally, 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)*

Whenever `loss1` is marked, the timed transition `tDecr1` (with variable arc multiplicity, cf. the zig-zag symbol) 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, $W_1(t)$ (`win1`) 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 $W_1(t)$ and $W_2(t)$, 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 finally have to address 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 3.4. In the proportional model they have no function, and do not influence the performance measures in any way.

3.3. *Performance measures*

We now express four performance measures of interest as reward-based measures: a connection's expected transmission rate, the connection throughput, the connection fairness and the utilization of the link.

The *expected transmission rate* for connection i is expressed as

$$\tau_i = \mathbb{E}\{r_i(Q)W_i\} = \sum_k \sum_{n_i} r_i(k) n_i \Pr\{Q = k, W_i = n_i\},$$

where k ranges over all possible discretised buffer filling levels, and n_i over all possible window sizes. Furthermore, Q and W_i are the random variables that are (jointly) distributed according to the steady-state distribution of the process underlying Markov chain; the probabilities $\Pr\{Q = k, W_i = n_i\}$ follows directly from the numerical evaluation of the steady-state probability vector of this Markov chain.

We define the *throughput* by considering the fluid that *leaves* the buffer. When the buffer is empty the departure rate at time t is equal to the arrival rate; this case corresponds to the first additive term below. When at time t the buffer contains $k > 0$ units of fluid, the departure rate of source i at time t equals the link speed L times the fraction of traffic of source i that arrived at time $t - kB/(KL)$. However, since the Markov chain we are considering 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 exactly incorporate this 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 results in the second additive term below. This yields the following throughput for source i:

$$\gamma_i = \mathbb{E}\left\{ r_i(Q)W_i \cdot \mathbf{1}\{Q = 0\} + L \cdot \frac{r_i(Q)W_i}{\mathbf{r}(Q) \cdot \mathbf{W}} \cdot \mathbf{1}\{Q > 0\} \right\}, \tag{8}$$

where $\mathbf{1}\{\text{condition}\}$ is the indicator that evaluates to 1 when the given condition is true, and 0 otherwise. 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 merely shifts the output process backward in time, but does not substantially change its shape or the ratio of fluid of the sources.

As an alternative to the above throughput measure, we can also define the throughput as the amount of fluid *entering* the buffer. A comparison of this throughput measure with the one defined above, reveals that in practice both throughput measures do equally well; see also the technical report [16].

Next to the absolute throughput measure defined above, we also will study the so-called *fairness*, being the ratio of throughputs of different connections. Such a measure expresses how fair the AIMD mechanism allocates capacity to competing connections.

Finally, we define the *link utilization* as the ratio of the connection throughputs and the link capacity, both per connection and overall, as follows:

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

3.4. The synchronous loss model

We now show how to adapt the SPN model such that it describes a synchronized loss strategy, that is, a loss model according to which *all* sources react to congestion by all reducing their rate simultaneously.

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`. Furthermore, in the new setting the transitions `tLoss1` and `tLoss2` will no longer form a random switch. Instead, each will fire with probability 1, when enabled.

Suppose 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`). The throughput as defined in (8) does not need any modification.

3.5. Computational complexity

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 steady-state distribution. We have not been able to find an accurate, yet simple, expression for $|\mathscr{M}|$, mainly due to the fact that the actual number of states depends critically on the values of the system parameters and the presence of guards.

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`. Furthermore, the number of different markings in place `winF`i in subnet S_i equals $N_i + 1$. Thus we obtain

$$|\mathscr{M}| = \mathscr{O}((K + 1)(N_1 + 1)(N_2 + 1)). \tag{9}$$

Observe that as the SPN contains only six timed transitions the number of non-zero entries per row in the generator matrix is bounded by six (excluding the diagonal). Consequently, the generator is very sparse.

Clearly, from a computational point of view it is of interest to choose N_1, N_2 and K as small as possible without significantly affecting the performance measures. First of all, it is plausible that for $K \to \infty$, $\{C(t), t \geq 0\}$ becomes a continuous process; see e.g., [17] for an application of such fluid queues to TCP modelling. 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. Then, approximately, the buffer is either empty or congested.

Finally, we remark that in case of the synchronized loss model, the above upper bound on the state space size remains valid; due to the increased number of possibilities with respect to the loss tokens, the number of states increases by at most a factor $\frac{4}{3}$.

4. A comparison with analytic models and NS2

In this section we compare the numerical results of our model to other analytic work and simulations with the network simulator NS2, see [28]. We specify the investigated scenarios in Section 4.1 and present the results in Section 4.2.

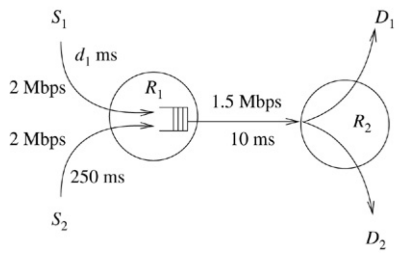


Fig. 2. The network configuration and some parameters.

Table 4
SPN parameter values

Scenario	L	d_B (ms)	K
25*s*	25.7	16	1
80*s*	80.7	16	1
25*l*	25.7	160	5
80*l*	80.7	160	5

We use mnemonics such as 25*s* to denote the scenario in which the link rate is $\underline{25}$.7 and the maximum buffering delay d_B is $\underline{s}$mall. A buffering delay of 16 ms corresponds to a buffer size of 5 packets in the NS2 simulation.

4.1. Traffic scenarios

Fig. 2 shows the network we used for the numerical analysis and NS2 simulations. To facilitate the comparison with [2] we use the same parameters. Two greedy TCP sources S_1 and S_2 communicate with destinations D_1 and D_2, respectively, via router R_1 with buffer size $B = d_B L$. We investigate two different values for both the link rate L and the maximum buffering delay d_B, thus leading to four scenarios. Table 4 shows the particular values of L and d_B we used. 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.

In the simulations with NS2 we use a RED buffer and consider a small buffer case and a large buffer case. In the small (large) buffer case, the buffer's total size is 20 (200) packets. The RED parameters in (2) and (3) are taken as follows. The minimum threshold $x_{\min}$ is 5 (50) packets, and the maximum threshold $x_{\max}$ is 10 (55) packets, resp. for the above two cases. 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 NS2 we follow an argument of [2]. It is commonly seen in simulations with two sources sharing a 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 modelled 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. Performance results

Below we present results for the fairness, the total utilization of the link and the normalized throughput for each of the connections for the network described in the previous subsection.

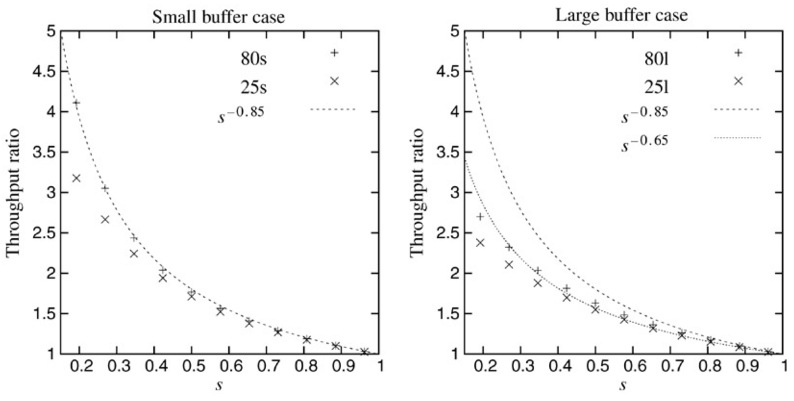


Fig. 3. Throughput ratios as a function of $s = T_1/T_2$ for the proportional loss model.

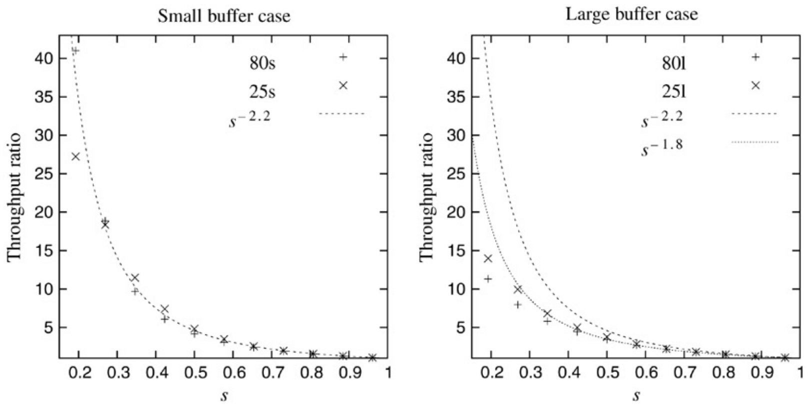


Fig. 4. Throughput ratios as a function of $s = T_1/T_2$ for the synchronous loss model.

To investigate the issue of *fairness*, we consider the ratio of throughputs γ_1/γ_2, where γ_i is the throughput of connection i. Both for the proportional loss model and the synchronous loss model we study how this ratio depends on $s = T_1/T_2$, the ratio of propagation delays. As comparison we mention that the authors of [2] present a model with proportional loss in which the throughput ratio behaves as $s^{-0.85}$, while in [21] it has been derived that in case of synchronous loss and a small buffer the throughput ratio behaves as s^{-2}. We also analyze the impact of the buffering delay, i.e., we present our results for both the small and large buffer cases introduced in the previous subsection.

In Fig. 3 we plot the throughput ratio for the model with proportional loss as a function of s for the scenarios of Table 4. We also plot $s^{-\alpha}$ for some values of α. The left and right panel present the results for the small and large buffer case, respectively, as will be the case in all figures of this section.

Notice that the throughput ratio is relatively insensitive to the link rate L. In the small buffer case, the ratio converges approximately to $s^{-0.87}$ as L increases, which is close to the result of [2]. Observe also that the sharing of the link becomes more fair when the buffer size increases (α drops to approximately 0.65). This is in accordance with intuition, since a large buffer will lead to larger delays, so that the total round-trip times of both connections will differ less than in the case of a small buffer, cf. (4).

In Fig. 4 we plot similar results but now for the synchronous loss model. For small buffers we see that the ratio according to our model behaves like $s^{-2.2}$ instead of s^{-2} as obtained by [21]. When the buffer size increases, the power decreases to a value smaller than 2, in line with the results of [21].

In Fig. 5 we plot the *total utilization* of the link, given by $(\gamma_1 + \gamma_2)/L$, as computed by our model, and compare it with a simulation of two New-Reno sources and two TCP Sack sources. We see from the graphs that our models overestimate the utilization in comparison to simulation, but correctly capture the trend that the utilization decreases as a function of s. Although this result is not as strong as one would hope, it is in some respects better than those

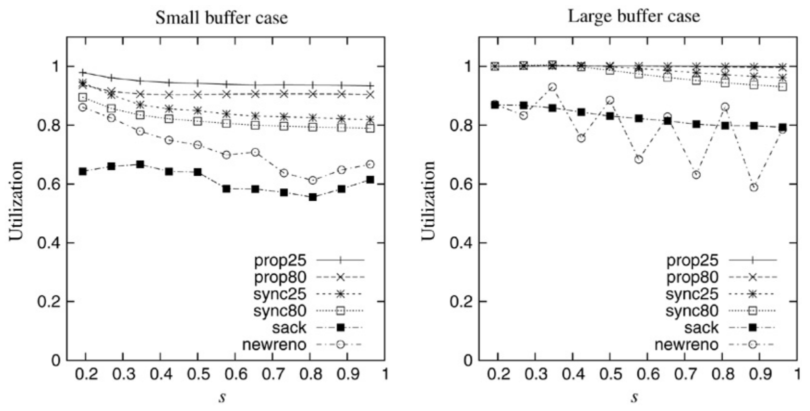


Fig. 5. The utilization as a function of $s = T_1/T_2$.

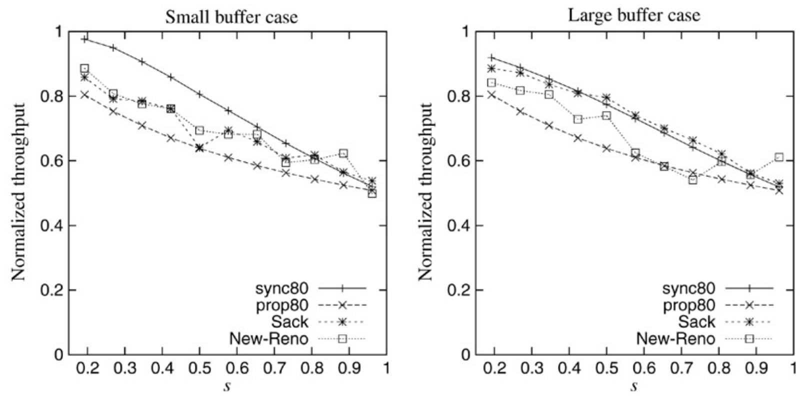


Fig. 6. The normalized throughput of source 1 as a function of $s = T_1/T_2$.

obtained earlier: in [21] (for synchronous loss and small buffers) the utilization is estimated as 0.75, independent of the ratio s. The analytic estimate from [3] in our (proportional loss) scenario also leads to an overestimation of the utilization, which actually increases from 0.79 to 0.875 as s grows from 0.1 to 1. Note that the proportional loss model is the more appropriate model to compare with the RED queue used in the simulations; we include the results for the synchronous case mainly for reference. Interestingly, in line with an observation in [2], the utilization in case of proportional loss is indeed higher than the utilization in case of synchronized loss. Finally, we mention that the results of the TCP New-Reno simulation in the large buffer case seem a bit odd, but, although they are not understood properly, they are correct. The oscillating behavior did not disappear by slight changes of the parameters of the RED buffer and as larger changes would introduce considerable differences between the model and the simulation, we did not investigate this further.

The last results of this section are in Fig. 6 and show the *normalized throughput* of the first connection, given by $\gamma_1/(\gamma_1 + \gamma_2)$, in comparison to the simulations; the results for the second connection follow immediately, as $\gamma_2 = 1 - \gamma_1/(\gamma_1 + \gamma_2)$. Clearly, for the small buffer case the proportional loss models are 'too fair', while the synchronous models are 'too unfair'. One explanation for these observations could be as follows. In the proportional loss model just one source loses traffic during periods with congestion, whereas in the synchronous model both sources always lose packets. However, in the simulation sometimes just *one* suffers from loss, and sometimes *both*. Hence, the loss process of a RED buffer is neither strictly proportional nor strictly synchronized, so that the utilization obtained by simulation should be in between the results of the two models, which is the case. We therefore infer that a more detailed model of the loss process might yield better resemblance to the simulation results.

We mention that the theoretical ratio from [3] is in nearly perfect agreement with our proportional loss curve for the small buffer case.

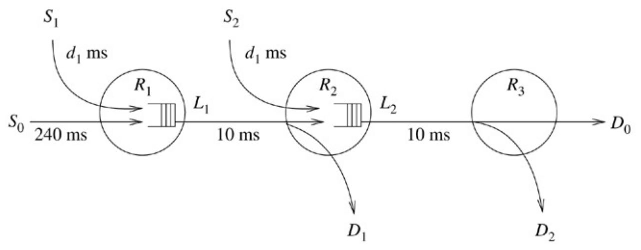


Fig. 7. A network of three sources sharing the links between routers R_1, R_2, and R_3.

Our main conclusion from this section is that our model captures the proportional sharing of the bandwidth between the two connections quite well, although the absolute values of the throughputs (or utilization) tend to be too optimistic.

5. An extended model: Three TCP sources sharing two buffers

The SPN framework allows for an easy extension of the model in various directions. We consider an extension of the model to include multiple sources, as well as an extension to include on/off-behavior of the sources; for conciseness, they are not presented here, but can be found in the technical report [16]. Instead, we focus on a network consisting of three sources and two links. Such a model appears difficult to tackle by other means.

5.1. Scenario and model

The scenario we investigate is depicted in Fig. 7 and consists of two links with link rates L_1 and L_2. Routers R_1 and R_2 are located before these links, with buffers of sizes B_1 and B_2 respectively. Three connections are using the network, each consisting of a source and destination (S_i and $D_i, i = 0, 1, 2$), where connection 0 uses both links, and connection 1 (respectively 2) only uses link 1 (2). Throughout this section it is assumed that the proportional loss scenario applies.

We first describe the SPN that corresponds to the setting above, then define the performance measures and finally present some results.

The SPN for the network is shown in Fig. 8. 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 Fig. 7, is different in that its connection uses both buffers. We elaborate on this now, generalizing the setting in Section 3.1.

We define the joint window process as $\mathbf{W}(t) = (W_0(t), W_1(t), W_2(t))$, where $W_i(t)$ is the state of the window of sender i, taking values in $\{0, 1, \ldots, N_i\}$ as before, N_i being the maximum window size of source i. A typical state of the joint process can then be written as a vector $\mathbf{n} = (n_0, n_1, n_2)$. Similarly, the joint buffer process is $\mathbf{Q}(t) = (Q_1(t), Q_2(t))$ where $Q_j(t)$ is the discrete buffer process for buffer j, taking values in $\{0, 1, \ldots, K_j\}$; a typical state of this process is written as a vector $\mathbf{k} = (k_1, k_2)$.

Defining T_i as the round-trip time of source i with empty buffer(s), it is now clear that the round-trip times of sources 1 and 2 when the joint buffer state is $\mathbf{k}$ can be expressed as

$$T_1(\mathbf{k}) = T_1 + \frac{k_1}{K_1}\frac{B_1}{L_1}, \quad \text{and} \quad T_2(\mathbf{k}) = T_2 + \frac{k_2}{K_2}\frac{B_2}{L_2}, \tag{10}$$

similar to (4), and the analog of (5) is immediate for these sources, e.g.,

$$r_1(\mathbf{k}) = \frac{\mathrm{MSS}_1}{T_1(\mathbf{k})}.$$

Note that this is in fact a function of k_1 alone, independent of k_2. The round-trip time of source 0 then equals

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

and the analog of (5) now becomes

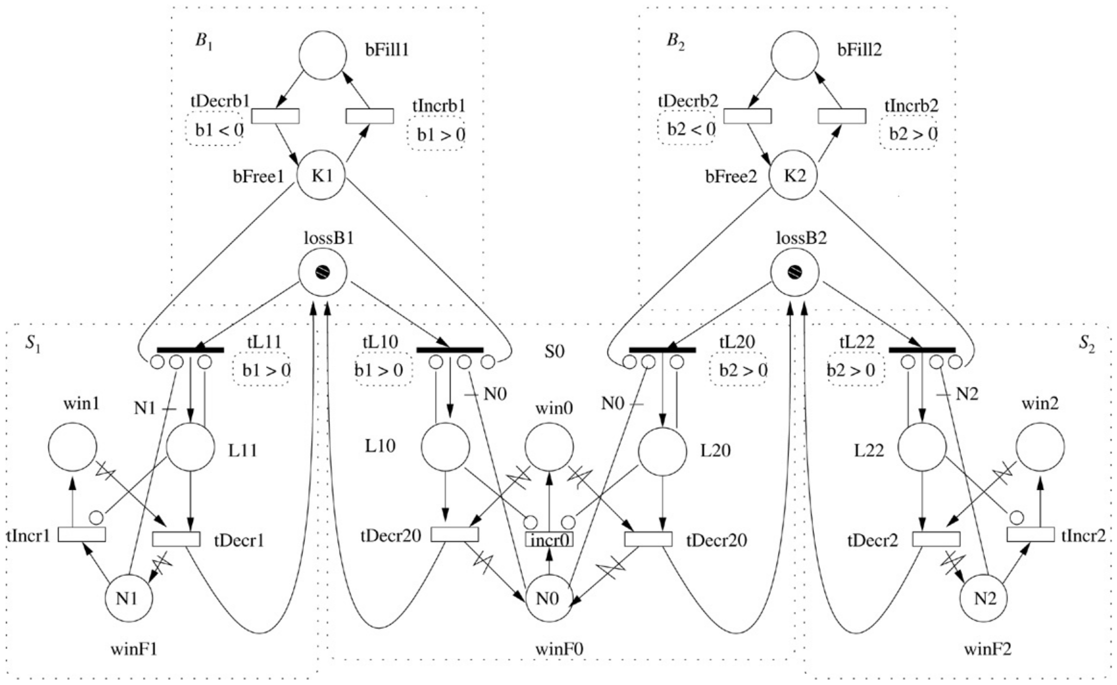


Fig. 8. TCP source 0 uses buffers 1 and 2, while source 1 (2) uses buffer 1 (2). Note that in Fig. 1 we used rather descriptive names for the transitions. In the present case we turned to shorter, but less descriptive, names for presentational reasons.

$$r_0(\mathbf{k}) = \frac{\text{MSS}_0}{T_0(\mathbf{k})},$$

which does depend on both k_1 and k_2. The analogue of (6) for the first buffer at times when $\mathbf{W}(t) = \mathbf{n}$ and $\mathbf{Q}(t) = \mathbf{k}$ is then given by

$$r_0(\mathbf{k})n_0 + r_1(k_1)n_1 - L_1,$$

leading to

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

for the transition rate at which increments and decrements of the first buffer level process occur, cf. (7).

To obtain a similar expression for β_2 we should account for the fact that the first buffer shapes the output process of source 0. We approximate the output rate, δ_0 say, of source 0 at the first buffer similarly to (8):

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

Now we can define β_2 as

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

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 is lost. However, we contend that this undesirable side effect has small impact, as follows. First, for this event to happen, congested periods of both buffers have to overlap. Second, once the congested periods overlap, source 0 should receive the loss tokens of both buffers. As source 0 will typically send at a lower rate than sources 1 and 2, both conditions will not often be satisfied simultaneously.

5.2. Performance measures

We are now ready to define the performance measures of interest, starting with the current throughputs at times t when $\mathbf{W}(t) = \mathbf{n}$ and $\mathbf{Q}(t) = \mathbf{k}$ as follows,

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

where δ_0 is as before and Δ_i is the current rate of source i, i.e.,

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

With this we find the expected throughputs as

$$\gamma_i = \mathbb{E}\{\gamma_i(\mathbf{W}, \mathbf{Q})\}, \quad i = 0, 1, 2, \tag{11}$$

where $\mathbf{W}$ and $\mathbf{Q}$ are the steady-state vectors describing the joint window state and joint buffer state respectively. The utilization of the first and second link can now also be found as

$$u_1 = \frac{\gamma_0 + \gamma_1}{L_1}, \quad \text{and} \quad u_2 = \frac{\gamma_0 + \gamma_2}{L_2}.$$

5.3. Performance results

To present the numerical results we compute the γ_i and u_j for this model. In the scenario that we consider, both buffers are identical, i.e., $L_2 = L_1 = 25.7$ and $d_{B_2} = d_{B_1} = 16$ ms, i.e., the small buffer case 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. Since $T_0 = 520$ ms and $T_1 = T_2 = 2d_1 + 20$ ms, this means that $s = T_1/T_0 = T_2/T_0$ varies between 0.2 and 1.

The left panel of Fig. 9 shows the ratios of the throughputs γ_1/γ_0 and γ_2/γ_0 as functions of s. 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 γ_2 is just slightly larger than γ_1 shows, in accordance with the above, that the rate of the thinned connection 0 is a bit smaller than its initial rate.

In the right panel of Fig. 9, we show the total utilization u_1 of the first link as a function of s. The graph is quite similar to that of the single buffer case, see the left panel of Fig. 5. We also show how the utilization is shared by the connections 0 and 1. Clearly connection 0 suffers from the fact that it also has to compete with connection 2 for using the second link. As the difference between γ_1 and γ_2 is small, we did not plot the corresponding graphs for the second link.

When $T_0 = T_1 = T_2$ we can make a comparison of our value of the throughput ratio γ_1/γ_0 to some theoretical fairness results for networks: the so-called minimum-potential-delay fairness scheme as defined in [24] and which, according to the authors of [22], is the most appropriate for TCP networks.

When we apply the theoretical analysis to our network as shown in Fig. 7 we obtain that $\gamma_1/\gamma_0 = \sqrt{2}$. Our SPN model, on the other hand, gives for $s = 1$ that $\gamma_1/\gamma_0 = 1.50$, which is quite near to $\sqrt{2}$. In Fig. 9 we plotted as a

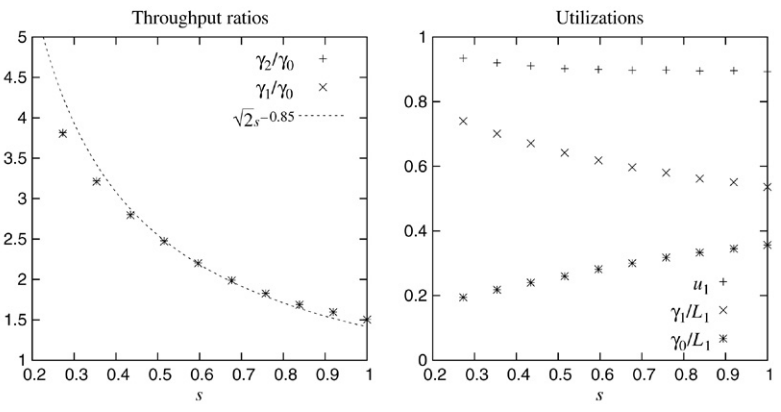


Fig. 9. The throughput ratios for connections 1 and 2 w.r.t. connection 0 (left panel) and the utilization of the first link (right panel) as functions of $s = T_1/T_0 = T_2/T_0$.

reference the function $\sqrt{2}s^{-0.85}$. This shows considerable agreement to the numerical results. Hence it seems that the function $cs^{-\alpha}$ is a good approximation for the throughput fraction when comparing two connections. The value of α is then dictated by the loss model, while the pre-factor is determined by the topology and can be computed according to the methods in [24]. We defer an investigation of this interesting conjecture to future research.

6. Summary and conclusions

We have used stochastic Petri nets to specify, in a versatile way, stochastic models of TCP New-Reno or Sack (more specifically, AIMD) sources that share one or two buffers.

Our modelling approach is new, in that we use SPNs to specify discretised fluid models. This is different from so-called fluid SPNs (FSPNs), in which indeed "fluid places" and true fluid tokens are used [29,20]. For such FSPNs the numerical algorithms require the solution of partial differential equation systems, which are solved using a discretisation approach. Alternatively, a special scheme toward discrete-event simulation has to be used [12]. One could argue that in our approach, we also take care of a discretisation, however, we do so already at the modelling level, so that we can resort to the much simpler numerical evaluation of the steady-state probabilities of an ordinary CTMC. Furthermore, our approach does not suffer from the modelling restrictions of FSPNs.

Our new methodology is flexible, extendable, and enables us to obtain qualitative insight into the impact of various source and network parameters on transient and long-term performance 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. [5,18], and therefore not reported here.

In our first model (two sources, one buffer) we implement two popular assumptions about the loss process at the buffer, proportional loss and synchronized loss. We validate our stochastic models for both cases by comparing the resulting fairness and utilization on the one hand to simulations with NS2, and on the other hand to the theory developed in [2], [3] and [21]. Our models provide results that are consistent with these theoretical results, and sometimes even improve these, in the sense that better resemblance is found to the simulation results. Moreover, our approach shows that the loss process is somewhere in between the proportional and synchronized loss model.

For our second model (three sources, two buffers) we are also able to show satisfying and insightful performance results. In particular when the round-trip times of all connections are equal and the buffers are small, the computed fairness is in line with the theoretical results in [24]. 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, our approach based on SPNs is one of the few theoretical approaches that enables such quantitative analysis. In [24] 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 noteworthy advantages over implementing the generator by hand, as done in [18]. It is easy to include complex behavior of the application layer, or modify aspects of TCP in the model. For instance, in a recent master thesis project, we studied various active queue management algorithms in the context of our models [13].

Acknowledgments

To obtain the performance results in this paper we used the software package SPNP. The authors thank Kishor S. Trivedi (Duke University) for making this package available. The authors are grateful to Chadi Barakat for sharing the NS2 script he used to obtain the results of [2]. We also thank Pasi Lassila for some help with the simulations. Finally, we thank the four reviewers for their constructive and insightful comments, that certainly helped in improving the readability of this paper.

References

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, Modelling with Generalized Stochastic Petri Nets, John Wiley & Sons, 1995.
[2] E. Altman, C. Barakat, E. Laborde, P. Brown, D. Collange, Fairness analysis of TCP/IP, in: Proc. of IEEE Conference on Decision and Control, 2000.
[3] E. Altman, T. Jimenez, R. Núñez-Queija, Analysis of two competing TCP/IP connections, Performance Evaluation 49 (2002) 43–55.
[4] K. Avratchenkov, U. Ayesta, E. Altman, P. Nain, C. Barakat, The effect of router buffer size on the TCP performance, in: LONIIS workshop on Telecommunication Networks and Teletraffic Theory, 2002.
[5] P. Brown, Resource sharing of TCP connections with different roundtrip times, in: Proc. of IEEE INFOCOM, 2000, pp. 1734–1741.
[6] M. Calder, V. Vyshemirsky, D. Gilbert, R. Orton, Analysis of signalling pathways using continuous time Markov chains, in: Transactions on Computational Systems Biology VI, vol. 4220, Springer-Verlag, 2006, pp. 44–47. http://www.dcs.gla.ac.uk/~muffy/papers/prismpaper.pdf.
[7] C. Casetti, M. Meo, A new approach to model the stationary behavior of TCP connections, in: Proc. of IEEE INFOCOM, 2000, pp. 367–375.
[8] C. Casetti, M. Meo, An analytical framework for the performance evaluation of TCP Reno connections, Computer Networks 37 (5) (2001) 669–682.
[9] C. Casetti, M. Meo, Modeling the stationary behavior of TCP Reno connections, in: QOS-IP, 2001, pp. 141–156.
[10] D.H. Chiu, R. Jain, Analysis of the increase and decrease algorithms of congestion avoidance in computer networks, Computer Networks and ISDN Systems 17 (1989) 1–14.
[11] G. Ciardo, G. Muppalla, K. Trivedi, SPNP: Stochastic Petri Net Package, in: 3rd Int. Workshop on Petri Nets and Performance Models, PNPM'89, IEEE Comp. Soc. Press, 1989, pp. 142–151.
[12] G. Ciardo, D.M. Nicol, K.S. Trivedi, Discrete-event simulation of fluid stochastic Petri nets, IEEE Transactions on Software Engineering 25 (2) (1999) 207–217.
[13] S. Dijkstra, Modeling active queue management algorithms using stochastic Petri nets, Master's Thesis, University of Twente, 2004.
[14] S. Floyd, T. Henderson, RFC 2582: The NewReno modification to TCP's Fast Recovery algorithm, Technical Report, IETF, 1999.
[15] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Transactions on Networking 1 (4) (1993) 397–413.
[16] N.D. van Foreest, B.R. Haverkort, M.R.H. Mandjes, W.R.W. Scheinhardt, Versatile Markovian models for networks with asymmetric TCP sources, Memorandum 1734, Department of Applied Mathematics, Enschede, The Netherlands, 2004. http://www.math.utwente.nl/publications/.
[17] N.D. van Foreest, M.R.H. Mandjes, W.R.W. Scheinhardt, Analysis of a feedback fluid model for heterogeneous TCP sources, Stochastic Models 19 (3) (2003) 299–324.
[18] N.D. van Foreest, M.R.H. Mandjes, W.R.W. Scheinhardt, A versatile model for asymmetric TCP sources, in: Proc. of ITC 18, 2003, pp. 631–640.
[19] R.J. Gibbens, S.K. Sargood, C. Van Eijl, F.P. Kelly, H. Azmoodeh, R.N. Macfadyen, 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, vol. 13, 2000.
[20] G. Horton, D. Nicol, V.G. Kulkarni, K.S. Trivedi, Fluid stochastic Petri nets: Theory, applications, and solution techniques, European Journal of Operational Research 105 (1) (1998) 184–201.
[21] T.V. Lakshman, 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.
[22] K.W. Lee, T.E. Kim, V. Bharghavan, A comparison of end-to-end congestion control algorithms: the case of AIMD and AIPD, in: Proc. of Globecom, 2001.
[23] Y. Liu, F. Lo Presti, V. Misra, D. Towsley, Y. Gu, Fluid models and solutions for large-scale IP networks, ACM SIGMETRICS Performance Evaluation Review 31 (4) (2003) 91–101.
[24] L. Massoulié, J.W. Roberts, Bandwidth sharing: Objectives and algorithms, in: Proc. of IEEE INFOCOM, 1999, pp. 1395–1403.
[25] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, RFC 2018: TCP selective acknowledgment options, Technical Report, IETF, 1996.
[26] V. Misra, W. Gong, 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) (2000) 151–160.

[27] L.L. Peterson, B.S. Davie, Computer Networks, second ed., Morgan Kaufman Publ., 2000.
[28] Network Simulator, Available at: http://www.isi.edu/nsnam/ns/.
[29] K.S. Trivedi, V.G. Kulkarni, FSPNs: Fluid stochastic Petri nets, in: Application and Theory of Petri Nets, 1993, pp. 24–31.

Nicky van Foreest (1967) studied Theoretical Physics at Utrecht University, The Netherlands. He worked for KPN Research and the Bell Labs department of Lucent Technologies in Enschede. In 2000 he started a Ph.D. project under the supervision of M.R.H. Mandjes and W.R.W. Scheinhardt on stochastic fluid queues at the University of Twente. Currently, he is an assistant professor in production planning and control at the Faculty of Management and Organization, University of Groningen, The Netherlands.

Boudewijn Haverkort (1964) obtained an engineering and a Ph.D. degree in computer science, both from the University of Twente, in 1986 and 1991 respectively. Since 2003, he is chairholder for Design and Analysis of Communication Systems at the University of Twente, The Netherlands. Prior to that, he was, among others, professor for performance evaluation and distributed systems at the RWTH Aachen, Germany, for seven years, lecturer in computer science at the University of Twente in The Netherlands, for five years and visiting researcher in the Teletraffic Research Centre at the University of Adelaide. His research interests emcompass the design and performance and dependability evaluation of computer-communication systems, model checking, parallel and distributed computing, and fault-tolerant computer systems. He has published over 75 papers in international journals and conference proceedings, edited several books and conference proceedings and wrote a monograph on model-based performance evaluation of computer and communication systems. Since 2005, he serves on the editorial board of *Performance Evaluation*.

Michel Mandjes (1970) received M.Sc. (in both mathematics and econometrics) and Ph.D. degrees from the Free University, Amsterdam, The Netherlands. After finishing his Ph.D., he worked as a member of the technical staff at KPN Research (1996–1998; Leidschendam, The Netherlands) and Bell Laboratories/Lucent Technologies (1999–2001; Murray Hill NJ, USA). Then he became part-time full professor of Stochastic Operations Research at the University of Twente, The Netherlands (2000–2004), and senior scientist/department head at the Centre for Mathematics and Computer Science (CWI) in Amsterdam (2000–2006). He is now full professor for Applied Probability at the University of Amsterdam, a chair that he already held part-time from 2004 on. He is also affiliated (as advisor) with EURANDOM, Eindhoven, The Netherlands. He is editor of *Stochastic Models* and *Queueing Systems*. His research interests include large deviations analysis of multiplexing systems, queueing theory, Gaussian traffic models, traffic management and control in IP networks, and pricing in multi-service networks.

Werner Scheinhardt (1969) received both his M.Sc. and Ph.D. degrees in Applied Mathematics from the University of Twente, The Netherlands, in 1994 and 1998 respectively. After holding a postdoctoral position at Eindhoven University of Technology, The Netherlands, he returned to the University of Twente in 2000 as an assistant professor. He also holds a part-time position at the Centre for Mathematics and Computer Science (CWI) in Amsterdam. His research interests are in the field of stochastic processes, with applications to the performance analysis of computer and communications networks.