Performance Evaluation of Threshold-Based ATM Cell Scheduling Policies under Markov-Modulated Poisson Traffic using Stochastic Petri Nets

Boudewijn R. Haverkort¹, Hessel P. Idzenga², and Byung G. Kim³

¹University of Twente, P.O. Box 217, 7500 AE Enschede, the Netherlands

²AT&T-NS-NL, Forward Looking Work, P.O. Box 18, 1270 AA Huizen, the Netherlands

³University of Massachusetts, Lowell, MA 01854, USA

Abstract

This paper addresses the modelling and evaluation of cell scheduling policies in ATM multiplexers using stochastic Petri nets. In particular, we compare four cell scheduling policies: a FIFO policy with and without non-preemptive priorities, a threshold priority policy as recently proposed by Lee and Sengupta, and an extension thereof.

The aim of this paper is twofold. First, we want to stress the suitability of stochastic Petri nets as a versatile tool for modelling and evaluating ATM-related performance issues. Secondly, we like to point out that under fairly realistic traffic conditions, i.e., under a combination of Poisson nonreal-time (data) traffic and MMPP real-time (video) traffic, our proposed extension of the threshold priority scheduling policy not only reduces the average delay for real-time traffic at the cost of only a slight increase of the average delay of nonreal-time traffic, but also reduces the delay jitter in the real-time traffic.

Keyword Codes: C.2.3; C.4; I.6.3

Keywords: Computer-Communication Networks, Network Operations; Performance of Systems; Simulation and Modelling, Applications

1 Introduction

ATM switches and multiplexers have been subjected to performance evaluations of many kinds. Analytical performance evaluation studies have mainly focussed on individual ATM multiplexers under symmetric traffic conditions. “Closed-form” analysis techniques that are often employed are matrix analytic and geometric techniques [7, 8, 17], and generating functions [14]. When the systems to be analyzed become more complex, or the traffic conditions become more complex or asymmetric, however, these analytical techniques fall short. In most of these cases, simulation studies are then performed. However, also simulation suffers from some drawbacks, most notably its relatively high cost, especially

in obtaining accurate estimates for small quantities such as blocking probabilities, and the error-prone process of coding simulation programs.

In between the closed-form analytical and the simulation approach, i.e., “in between” with respect to both the modelling capabilities and the evaluation costs, lies the numerical approach based on stochastic Petri nets (SPNs) [1, 3, 6, 16]. SPNs allow for a very flexible construction and solution of, possibly large, continuous-time Markov models of ATM switches. In Appendix A we present a brief overview of the capabilities of SPNs. What is important to understand here, is that SPNs represent a formally defined framework for the construction of large continuous-time Markov chains (CTMCs). These models can be solved numerically, using powerful current-day software packages and workstations. Advantage of the SPN approach over simulation is that rare-events are much less of a problem. Also, by its formally well-established semantics, SPN models are easily constructed and less error-prone than C-coded simulation programs. By the fact that a numerical solution is employed, more generality in the models can be achieved than with the closed-form analytical approaches, as will be illustrated in this paper.

We are only aware of one other SPN-based performance study of ATM switches, i.e., the paper by Kant and Sanders in which they analyse the knockout switch under non-uniform and bursty traffic [11]. The conclusions from their analyses are that the cell loss probabilities under the non-uniform and bursty traffic conditions are much higher than under more mild, i.e., Poisson, traffic conditions. In this respect, the SPN-based approach revealed more detailed characteristics than earlier performed analytical performance studies.

In this paper we will describe a number of workload models as SPNs. Similarly, we will describe ATM cell scheduling policies at multiplexers using SPNs. Then, we will combine these model parts so as to form realistic performance models of ATM multiplexers under realistic traffic conditions. We parameterise our models with measurement results reported in the literature. In our analyses, we will focus on mean response times, the variance in the queue length, and blocking probabilities, for both real-time and nonreal-time traffic.

This paper is further organised as follows. In Section 2 we present, using SPNs, a number of workload models that have been proven successful in the past to describe the expected traffic for ATM switches. Similarly, in Section 3, we present SPN models of cell scheduling policies in ATM multiplexers. In Section 4 we combine these model parts and discuss a number of realistic performance evaluation studies of ATM cell scheduling policies. Section 5 concludes the paper. Appendix A contains a brief summary of SPNs.

2 Workload models

We discuss the traditional Poisson process and its corresponding SPN in Section 2.1, and the Markov modulated Poisson process and its corresponding SPN in Section 2.2; these two sections address the arrival pattern of cells to the multiplexers. In Section 2.3 we discuss the cell length distribution.

Figure 1: A Poisson process as an SPN model

2.1 Poisson processes

A data source is typically modelled as a Poisson process, in which the times between two successive cell arrivals are assumed to be independent. In an SPN, a Poisson process is modelled as depicted in Figure 1. As long as there are tokens in the outside **world**, i.e., as long as transition **arrive** is enabled, tokens can arrive via transition **arrive**, after which they end up in place **buffer** which models the buffer of the switch studied. Notice that the initial number of tokens in place **world** must be limited so as to make sure that the underlying CTMC has a finite state space.

2.2 Markov modulated Poisson processes

Due to the small size of an ATM cell, sources typically produce streams of cells when active [7, 9, 19]. The conventional modelling by a Poisson input process is therefore considered to be no longer valid. Instead, an “on/off” model is used to capture the fact that sources alternate between an active (on) period during which cells are periodically emitted and a silence (off) period in which no cells are produced. In case there is only one source, this yields an interrupted Poisson process (IPP). When N similar IPPs are multiplexed, one obtains a Markov modulated Poisson process (MMPP) with $N + 1$ states, where the state number indicates the number of active sources. When N grows very large, such an MMPP will again converge to a simple Poisson process.

In Figure 2 we depict the SPN representation of an MMPP. Place **active** denotes the number of active sources; place **passive** denotes the number of passive sources. The transitions **begin** and **end** model the beginning and the ending of an active period of a single source. Their rates are linearly dependent on the number of tokens in places **passive** and **active** respectively, i.e., the more sources are passive, the higher the rate is that one becomes active, and *vice versa*.

Next to the above so-called modulating part, we again have a Poisson process of which the rate (transition **arrive**) is linearly dependent on, i.e., modulated by, the number of tokens in place **active**. If the arrival rate of a single active source is denoted λ , the actual rate of transition **arrive** is $\lambda \times \#(\text{active})$, where $\#(P)$ denotes the number of tokens in place P .

For our analyses concerning video, we use a special type of MMPP as proposed by Saito [19]: a 3-state cyclic CTMC is used for every source. One cycle corresponds to the length of a single frame, i.e., to 1/30 sec. The average delay in each state of the cycle is chosen to be 1/90 sec. To match the peak and mean arrival rates as given below, during the third phase cells arrive at the peak rate, while during the first two phases no cells

Figure 2: A Markov modulated Poisson process as an SPN model

Figure 3: Saito's video traffic model as an SPN model

arrive. In Figure 3 we present the corresponding SPN where the initial number of tokens in place `silent-1` denotes the number of sources. The arrival rate of the cells, according to a Poisson process, is linearly proportional to the number of tokens in place `at-peak`.

For modelling voice traffic, the following parameters are often used [19]: talkspurts and silence periods last 352 and 650 msec, respectively. The peak rate typically is 64 kbps. For video traffic the peak rate is assumed to be 44.7 Mbps and the average bit rate is 16.8 Mbps.

2.3 Cell length

Apart from the arrival pattern of cells, also the cell service times form a part of the workload model. As cells are of fixed length, their service requirement is of a deterministic nature. However, due to the fact that the SPNs are solved via an underlying CTMC, it is most easy to assume that the service times are exponentially distributed. In Appendix A we indicate various approaches that have been taken to include or approximate deterministic (cell) service times in SPN models.

In this paper we will follow the method of stages, i.e., we will use the Erlang- k distri-

bution to model cell service times (with either $k = 2$ or $k = 3$). It is well-known, that when k is not too small, this approach yields adequate results. This is especially true when the cell arrival process is becoming more bursty, i.e., when MMPPs are employed, due to the following fact.

When bursty traffic sources are employed, the probability that either very many or very few cells are queued, increases in comparison with more smooth cell sources (such as Poisson sources); whenever the arrival process is in a burst, very many cells will be present, whereas during idle periods of the source almost no cells will be queued. This observation implies that most cells passing the multiplexer, will do so in periods of relatively long queues and will therefore experience many cells queued in front. The waiting time a particular cell perceives, therefore can be seen as the sum of many cell service times, i.e., of all the cells queued in front. The distribution of this sum of service times will decrease as the number of queued cells increases (the individual variations will average out), i.e., when the sources become more bursty. This implies that, from a statistical point of view, the variance of the individual cell service times becomes less important as the cell arrival pattern becomes more bursty; only the mean values remain important. This is another reason why the Erlang- k approximation is not so bad after all. For more background on this topic, we refer to some work Neuts and others did on this topic [8, 17, 18].

3 Cell scheduling policies

In this section, four cell scheduling policies are presented. A simple FIFO policy with and without non-preemptive priorities is presented in Section 3.1 and a threshold priority policy, as recently proposed in [14], is discussed in Section 3.2. An extension of this threshold priority policy is proposed in Section 3.3.

It should be noted that the models presented in this section only give the cell scheduling part of the performance models. The models still have to be combined with the workload models given in Section 2, form the whole performance model that is suitable for evaluation.

In all the scheduling models that follow, the cell service times are depicted as single (exponential) transitions. This is only done to keep the figures concise; in the actual evaluation studies we did, the cell service times were assumed to have an Erlang- k distributed length (with k either being 2 or 3, depending on the state space size capabilities of our workstation).

The proposed cell scheduling policies should be regarded as possible implementations to deal with prioritized traffic, as indicated in ITU-T Recommendation I.371 [10]. Based on the 1-bit Cell Loss Priority (CLP) in every ATM cell header, a cell is classified as being either of high or low priority. The idea is to maintain end-to-end quality of service requirements for both traffic classes, however, if congestion appears, with highest priority to the high-priority cells.

Depending on the specific implementation of the policies, e.g., with or without truly separated buffers, or with or without combined buffers with flexible boundary, the scheduling policies can be classified as being time- or space-based, or even a combination

thereof. To preserve in-sequence integrity, all cells of a single connection should be given the same value for the CLP bit.

3.1 First-in, first-out

In the first-in, first-out (FIFO) policy, a single buffer is used in which cells from both real-time and nonreal-time sources are stored without service discriminations.

To make an SPN model that really takes into account the exact arrival pattern of packets with their types, would require a very large model. Instead, we approximate the FIFO policy by a so-called random policy. In a random scheduling policy with two cell classes, not the exact arrival ordering of the buffered cells from the two classes is taken into account, but only the number of buffered cells, at any moment in time. The decision from which class to serve the next cell, is then probabilistically taken, based on the relative occurrence of cells, rather than on the fact which cell really is queued in front.

To make this more concrete, consider the case when there are n_{rt} real-time packets and n_{nrt} nonreal-time cells queued at the moment a new cell to be transmitted has to be selected. The next cell to be transmitted in the random policy, is then with probability $p_{rt} = n_{rt}/(n_{rt} + n_{nrt})$ a real-time cell, and with probability $p_{nrt} = n_{nrt}/(n_{rt} + n_{nrt}) = 1 - p_{rt}$ a nonreal-time cell. This choice can be explained by the fact that, given n_{rt} and n_{nrt} , of all possible FIFO cell queueing orderings, a fraction p_{rt} has a real-time packet queued up front and a fraction p_{nrt} has a nonreal-time packet queued up front.

Using the random policy as opposed to the FIFO policy reduces the size of the state space tremendously. Practical experience also has turned out that this form of state lumping gives good results in general. The minor differences that might occur are not very important here, since we use the FIFO (or random) policy only for comparison with the TPP and ETPP policies.

Modelling this policy with SPNs, yields the model as given in Figure 4. There are two buffers: **buff-nrt** for nonreal-time traffic and **buff-rt** for real-time traffic. When the server is available, i.e., when there is a token in place **server**, the next cell to be transmitted is decided upon by the two immediate transitions **rt** and **nrt** whose weights are linearly proportional to the number of tokens in the connected buffer places. Since there is initially only 1 token in place **server** there can only be one transmission active at any time. After service completion, via either of the two transmission transitions **Tx-rt** or **Tx-nrt**, the cells leave the switch and the server becomes available for the next cell transmission.

As stated above, it should be noted that the incoming tokens, in the places **buff-rt** and **buff-nrt**, originate in the arrival process SPN models, as presented in Section 2; the departing tokens (cells), indicated with **realtime-** and **nonreal-time** departures, flow back to the corresponding places named **world** in the workload models.

If we add an inhibitor arc from place **buff-rt** to transition **nrt** we prohibit the start of a transmission of a nonreal-time cell whenever there are real-time cells buffered, i.e., we give absolute non-preemptive priority to real-time traffic. We refer to this policy as FIFO-PR.

Figure 4: The FIFO policy as an SPN model

3.2 Threshold priority policy

In the threshold priority policy (TPP) [14], two buffers are used for individual traffic classes. A predetermined threshold L is associated with the real-time buffer. When the queue length in the real-time buffer is less than or equal to L , the server alternates between the two buffers transmitting one cell from each buffer (as long as a queue is not empty). On the other hand, when the queue length in the real-time buffer exceeds L , the server continues transmission only from the real-time buffer until its queue length is reduced to L . The value of the threshold L gives the degree of preferential treatment of the real-time traffic. When $L = 0$, real-time traffic is given an absolute priority like in the FIFO-PR policy. In case $L = \infty$, both traffic classes are served alternately when not empty. By selecting L between these two extremes, one may provide an adequate quality of service to both real-time and nonreal-time traffic.

In Figure 5 we depict the corresponding SPN model. On the left side, we see the arrival streams coming into the buffers for the two traffic classes. The arriving tokens, via the transitions `arrive_rt` and `arrive_nrt`, originate in the arrival process SPN models, as presented in Section 2; the departing cells, indicated with `realtime-` and `nonreal-time` departures, flow back to the corresponding places named `world` in the workload models.

The server is represented by the single token that alternates between places `try_rt` and `try_nrt`. After a cell of one class is served (via either transition `serve_rt` or `serve_nrt`) the server polls the other class. When nothing is buffered for a particular traffic class, the server also polls the other class, via the transitions `empty_rt` and `empty_nrt`. However, depending on whether there are more or less than L cells buffered in place `buff_rt`, it can be decided that the server remains serving the real-time traffic class. This is enforced by the immediate transitions `rt-rt`, `rt-nrt`, `nrt-nrt`, `nrt-rt1` and `nrt-rt2`. Apart from the normal enabling conditions for these transitions, i.e., at least a token in every input place and no tokens in places that are connected via an inhibitor arc to the transition, these transitions have so-called *enabling functions* associated with them. Whenever all normal condition for being enabled are fulfilled, the associated enabling function is validated and must yield `true` for the transition to be enabled. In fact, all transitions have an associated enabling function. However, when not specified explicitly, this function always yields the value `true`. The enabling functions for the five immediate transitions are given in Table 1; they are taken such that the TPP is exactly enforced.

Figure 5: The TPP as an SPN model

Table 1: Enabling functions for the immediate transitions in the TPP model

transition	enabling condition
rt-rt	$(\#buff_rt > L) \text{ or } ((\#buff_rt > 0) \text{ and } (\#buff_nrt = 0))$
rt-nrt	$(\#buff_rt > 0) \text{ and } (\#buff_rt \leq L)$
nrt-nrt	$(\#buff_rt = 0) \text{ and } (\#buff_nrt > 0)$
nrt-rt1	$(\#buff_rt > L)$
nrt-rt2	$(\#buff_rt > 0)$

Table 2: Enabling functions for the immediate transitions in the ETPP model

transition	enabling condition
rt-rt	(#sttrt= 1) or ((#buff_rt> 0) and (#buff_nrt= 0))
rt-nrt	(#buff_nrt> 0) and (#normal= 1)
nrt-nrt	(#buff_rt= 0) and (#buff_nrt> 0)
nrt-rt1	(#buff_nrt= 0) or (#sttrt=1)
nrt-rt2	(#buff_rt> 0)
from-normal	(#buff_rt> L)

3.3 Exhaustive threshold priority policy

When the TPP is used in combination with a Poisson arrival stream, it can lead to acceptable performance for both traffic classes, as shown in [14]. However, when used in combination with MMPP real-time traffic, it is highly likely that by the bursty nature of the sources, once the real-time buffer exceeds L , the buffer will fill rapidly. Instead of again polling the nonreal-time buffer again when the buffer occupancy is smaller than L , one could give priority to the real-time traffic for the duration of the burst. This can be done by serving the real-time buffer until it is empty. This policy, denoted the exhaustive threshold priority policy (ETPP), introduces a hysteresis in the thresholding strategy, and thereby reduces the rapid succession of switching to and from giving priority to the real-time traffic. Although this will penalize the nonreal-time traffic, we expect less variance in the queue lengths, and therefore in the delays of the real-time traffic.

In the corresponding SPN, extra functionality is required to distinguish between the two different possibilities that can occur when the buffer occupancy is smaller than L , i.e., either the server alternates between the two queues, or it remains at the real-time queue as this queue needs to be emptied because the buffer occupancy has been larger than L but has not been zero since then.

In Figure 6 the overall SPN is shown. Added are the two places `normal` and `sttrt` (service to real-time traffic). Place `normal` initially contains a single token, indicating that the operation is normal, i.e., as before in the TPP. The immediate transition `from-normal` has an enabling function that evaluates to `true` whenever place `buff_rt` is occupied by more than L tokens. When it becomes enabled, it fires and puts a token in `sttrt`. This indicates that now all the service capacity is given to the real-time traffic, until the real-time buffer is empty again. Transition `to-normal` becomes only enabled when `buff_rt` becomes empty. Consequently, after firing, normal operation is resumed. In Table 2 we present the slightly adapted enabling functions for the ETPP.

4 A variety of performance analyses

In this section we compare the FIFO(-PR) policy with the TPP under Poisson traffic assumptions in Section 4.1. We then analyse the TPP under a more realistic workload in Section 4.2. This analysis also suggests the improvement of TPP towards ETPP.

Figure 6: The ETPP as an SPN model

Table 3: The addressed combinations of switch and workload models

Section	FIFO	FIFO-PR	TPP	ETPP
PP	4.1	4.1	4.1	
MMPP			4.2–3	4.3

Finally, in Section 4.3, ETPP is compared with TPP under realistic traffic. The addressed combinations of switch and workload models are summarized in Table 3.

4.1 FIFO versus TPP under Poisson traffic

We first compare FIFO with TPP under Poisson traffic, thereby using the numerical parameters as assumed by Lee and Sengupta [14]: the server speed is 3642 cells/sec (1.544 Mbps with cell of 53 bytes), the data load is always 40% or 1456 cells/sec (Poisson process) and the voice load is increased from 1 to 13 calls where every call brings in an extra 150 cells/sec. To approximate the deterministic cell service times we have used Erlang-3 cell service times.

Notice that although we have specified the models as SPNs, we solve them using well-known closed-form M|G|1 results for FIFO and non-preemptive priority scheduling [12]. First, we address the M|G|1-FIFO case. The overall cell arrival rate $\lambda(k) = \lambda_d + \lambda_v(k) = 1456 + 150k$ cell/sec, with k the number of active voice calls. Consequently, $\rho(k) = \rho_d + \rho_v(k) = \lambda_d E[S] + \lambda_v E[S] = \lambda(k) E[S]$, with $E[S] = 1/3642$ the average cell

service time. For the second moment we find $E[S^2] = 12/3642^2$. We calculate the average response time $E[R(k)]$ for both voice and data cells, given k voice calls, as

$$E[R(k)] = E[S] + \frac{\lambda(k)E[S^2]}{2(1 - \rho(k))}, \quad k = 1, \dots, 13 \text{ voice calls.} \quad (1)$$

In the case of FIFO-PR, we find that the average response time for voice cells $E[R_v(k)]$, given k voice calls, equals

$$E[R_v(k)] = E[S] + \frac{E[T]}{1 - \rho_v(k)}, \quad k = 1, \dots, 13 \text{ voice calls,} \quad (2)$$

where $\rho_v(k) = 150kE[S]$ is the utilization due to voice calls, and where $E[T] = \frac{1}{2}(\lambda_d + \lambda_v(k))E[S^2]$ is the expected residual service time of a cell as perceived by an arriving cell. Similarly, we find that the average response time for data cells $E[R_d(k)]$, given k voice calls, equals

$$E[R_d(k)] = E[S] + \frac{E[T]}{(1 - \rho_v(k))(1 - \rho(k))}, \quad k = 1, \dots, 13 \text{ voice calls.} \quad (3)$$

In Figure 7 we show the average response time $E[R]$ (in msec) as a function of the number of supported voice calls for the FIFO policy, with and without the non-preemptive priority for real-time traffic. As can be observed, using the pure FIFO policy, no distinction is made between real-time and nonreal-time traffic. However, if we grant non-preemptive priority to real-time traffic, a large performance increase can be observed, however, at the cost of a performance decrease for data traffic.

The analysis above, using closed-form M|G|1 results, might lead to the question what the use of an SPN-based approach is anyway. Indeed, for this particular combination of scheduling policy and arrival process a cheap analytical solution is available, however, only for the mean response times and infinite buffer size. If we would be interested in a finite-buffer model, in more general arrival processes like MMPPs, in more intricate scheduling policies, or in more detailed performance measures like buffer occupancy probabilities, the above closed-form solution does not help us much. In these cases, however, an SPN-based approach still is appropriate.

The SPN approach is therefore used to analyse the TPP in combination with Poisson traffic. In Figure 8 we depict the average response times for both real-time and nonreal-time traffic and various thresholds L , for increasing number of voice sources. We observe that, in case $L = 0$, the curves come very close to those for the FIFO-PR policy in Figure 7. For larger values of L , the curves approach the (middle) FIFO curve in Figure 7. Clearly, by changing L , we have obtained a means for trading real-time performance against non-real-time performance. Also notice the similarity of these two figures with the corresponding figures in [14] (Figures 2 and 3).

4.2 TPP under MMPP traffic

In this section we present the analysis of the TPP under MMPP traffic. Such a combination of workload and system models can not be handled anymore with the approach presented in [14].

Figure 7: $E[R]$ (in msec) for the FIFO and the FIFO-PR policy under Poisson traffic for increasing number of real-time (voice) sources

Figure 8: $E[R]$ (in msec) for the TPP under Poisson traffic for increasing number of real-time (voice) sources

Figure 9: $E[R]$ (in μsec) for the TPP under MMPP traffic for increasing number of real-time (video) sources

we use the following parameters. The server speed is 600 Mbps, or 1415094 cells/sec. The nonreal-time traffic, modelled as a Poisson process, amounts for 40% of the load. The real-time traffic is, for every source (increased from 1 to 11), described using the model of Saito and the parameters given in Section 2.2. The cell service times have an Erlang-2 distribution (using an Erlang-3 distribution increased the state space size with a factor $1.5^2 = 2.25$ which could not be dealt with in reasonable time).

In Figure 9 we depict the average cell response time (delay) for increasing number of video sources. For $L = 0$, i.e., the absolute priority limiting case, the mechanism still works fine. For $L = 4$, the operation already becomes less pronounced. Moreover, for $L = 6$, the aimed-at strategy does not seem to work anymore. Indeed, under high load, the average delay for the real-time traffic is higher than for the nonreal-time traffic! This is exactly the opposite of what we were aiming at. The TPP does not seem to be able to cope well with the bursty character of the MMPP traffic.

In the following section we will discuss the improvement made with the ETPP with respect to the real-time traffic performance.

4.3 TPP versus ETPP under MMPP traffic

In Figures 10 through 13 we present some performance results of the ETPP and compare it with the TPP, when the traffic is, at least for the realtime part, of MMPP type.

In Figure 10 we observe that, for a fixed threshold $L = 6$, the ETPP does indeed decrease the expected response time for real-time cells at the cost of a small increase in the expected response time for nonreal-time cells. Indeed, the earlier distinguished better performance (smaller average response time) for the nonreal-time traffic has vanished.

Figure 10: The average response time $E[R]$ (in μsec), $L = 6$, for increasing number of real-time (video) sources

Moreover, in Figure 11 it can be observed that also the variance of the real-time buffer occupancy decreases by the proposed modification of the scheduling policy. This is a very nice property, as this also reduces the delay jitter in the real-time traffic. This property only comes at the cost of a small increase in variance of the nonreal-time buffer occupancy. A less pronounced but similar effect has been observed in case $L = 4$ (not shown here).

In Figure 12 we depict the buffer-full probability β for a fixed buffer size of 15 cells, a threshold $L = 6$, and an increasing number of supported real-time sources (note the logarithmic β -scale). The earlier mentioned video model of Saito is used. Cell service times are assumed to be Erlang-2 distributed. As to be expected, β increases with increasing load. We also observe a small improvement when comparing ETPP with TPP.

Finally, in Figure 13 we depict the buffer-full probability β (again, note the logarithmic β -scale) for a fixed threshold $L = 6$, 8 active real-time sources and the number of buffers increasing from 10 through 20. The same workload assumptions as above apply. Over the whole range of buffer sizes, the ETPP provides a fixed decrease of β by almost one order of magnitude.

Finally, notice that with our SPN-based, numerical solution approach, the computation of blocking probabilities as small as 10^{-12} is not a problem at all.

5 Concluding remarks

In this paper we have shown how to use SPN models for the analysis of cell scheduling policies in ATM multiplexers. The aim of the paper has been twofold. First of all, it aims to show the versatility of the SPN approach. From the discussed evaluations, it has

Figure 11: The variance $\text{var}[N_q]$ (in cell^2) of the number of queued real-time and nonreal-time packets, $L = 6$, for increasing number of real-time (video) sources

Figure 12: The blocking probability β for increasing number of real-time (video) sources and $L = 6$

Figure 13: The blocking probability β for increasing real-time buffer size, $L = 6$ and 8 real-time (video) sources

become clear that SPNs are a convenient way to describe and evaluate the performance of complex cell scheduling strategies. The ability to not only calculate average performance measures but also variances and very small blocking probabilities from the same model has been advantageous.

The second aim of the paper has been to extend a recently proposed threshold priority scheduling policing that provides a better quality of service for (real-time) video traffic, also when realistic traffic assumptions are done; previous studies only assumed Poisson traffic. The SPN analysis of this exhaustive threshold priority policy has indeed revealed that not only lower response times are obtained but also that there is less variance in the video buffer occupancy. The latter fact implies that there will be less jitter in the delay, which is normally regarded as a better quality of service.

Although the SPN-approach yields good results, it can be quite costly. The ETPP model under MMPP traffic has an underlying CTMC with as much as 3153 states and 10258 nonzero entries in its generator when only one source is modelled, up to 81978 states (432948 nonzero entries) in case of 11 sources. Although performed completely automatically, the former model requires about 20 seconds to be solved whereas the latter requires about 90 minutes to be solved on a SUN Sparc 10 (2 processors) and 32 MB main memory. This indicates the need for ongoing research in the field of numerical solution and state space reduction techniques for SPN models.

Future work will include a wider variety of analyses for the models presented here, as well as the development of models for other workloads and cell scheduling policies. Also, other SPN modelling and evaluation tools will be used, e.g., for the inclusion of deterministically timed transitions.

References

- [1] M. Ajmone Marsan, G. Conte, G. Balbo, "A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems", *ACM Transactions on Computer Systems* **2**(2), pp.93–122, 1984.
- [2] G. Ciardo, J. Muppala, K.S. Trivedi, "SPNP: Stochastic Petri Net Package", *Proceedings of the Third International Workshop on Petri Nets and Performance Models*, IEEE Computer Society Press, pp.142–151, 1989.
- [3] G. Ciardo, J. K. Muppala, and K. S. Trivedi, "On the Solution of GSPN Reward Models", *Performance Evaluation* **12**(4), pp. 237–254, 1991.
- [4] G. Ciardo, R. German, C. Lindemann, "A Characterization of the Stochastic Process Underlying a Stochastic Petri Net", *IEEE Transactions on Software Engineering* **20**(7), pp.506–515, 1994.
- [5] G. Chiola, "A Graphical Petri Net Tool for Performance Analysis", in: *Modelling Techniques and Performance Evaluation*, Editors: S. Fdida, G. Pujolle, North-Holland, pp.323–333, 1987.
- [6] J.A.Couvillion, R. Freire, R. Johnson, W.D. Obal II, A. Qureshi, M. Rai, W.H. Sanders, J.E. Tvedt, "Performability Modelling with UltraSAN", *IEEE Software*, pp.69–80, September 1991.
- [7] V. Frost, B. Melamed, "Traffic Modelling for Telecommunications Networks", *IEEE Communications Magazine* **32**(3), pp.70–81, 1994.
- [8] B.R. Haverkort, A.P.A. van Moorsel, D.-J. Speelman, "Xmgm: A Performance Analysis Tool Based on Matrix Geometric Methods", in: *Proceedings of the Second International Workshop on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, IEEE Computer Society Press, pp.152–157, 1994.
- [9] H. Heffes, D.M. Lucantoni, "A Markov Modulated Characterisation of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performance", *IEEE Journal on Selected Areas in Communications* **4**(6), pp.856–868, 1986.
- [10] ITU-T Recommendation I.371, "Integrated Services Digital Network (ISDN), Overall Network Aspects and Functions—Traffic Control and Congestion Control in B-ISDN", International Telecommunication Union, March 1993.
- [11] L.A. Kant, W.H. Sanders, "Performance Evaluation of the Knockout Switch under Bursty and Nonuniform Traffic Conditions", *PMRL Technical Report 93-14*, University of Arizona, 1993.
- [12] L. Kleinrock, *Queueing Systems: Volume 1 and 2*, John Wiley & Sons, 1975/1976.
- [13] U. Krieger, B. Müller-Clostermann, M. Sczittnick, "Modelling and Analysis of Communication Systems Based on Computational Methods for Markov Chains", *IEEE Journal on Selected Areas in Communications* **8**(9), pp.1630–1648, 1990.

- [14] D.-S. Lee, B. Sengupta, "Queueing Analysis of a Threshold Based Priority Scheme for ATM Networks", *IEEE/ACM Transactions on Networking* **1**(6), pp.709–717, 1993.
- [15] C. Lindemann, R. German, "DSPNexpress: A Software Package for Efficiently Solving Deterministic and Stochastic Petri Nets", in: *Computer Performance Evaluation 1992: Modelling Techniques and Tools*, Editors: R. Pooley, J. Hillston, Edinburgh University Press, 1993.
- [16] C. Lindemann, "An Improved Numerical Algorithm for Calculating Steady-State Solutions of Deterministic and Stochastic Petri Net Models", *Performance Evaluation* **18**, pp.79–95, 1993.
- [17] M.F. Neuts, *Matrix Geometric Solutions in Stochastic Models—An Algorithmic Approach*, The Johns Hopkins University Press, 1981.
- [18] M.F. Neuts, "The Caudal Characteristic Curve of Queues", *Advances in Applied Probability* **18**, pp.221–254, 1986.
- [19] H. Saito, *Teletraffic Technologies in ATM Networks*, Artech House, Boston, 1994.

A Stochastic Petri nets

Below, we briefly introduce SPNs in Section A.1. We discuss tool support in Section A.2 and comment on the usage of different transition-firing distributions in Section A.3.

A.1 Introduction

Stochastic Petri nets have been defined as extensions of Petri nets by associating time with transitions. Many variants have SPNs have been proposed [1, 3, 6, 16]; we adhere to the variant proposed by Ciardo *et al.* [2, 3]. The description that follows is necessarily concise; please refer to the original SPN papers for more information.

An SPN consists of a set of places (circles) P , a set of transitions (bars) T , and arcs connecting transitions to places and *vice versa*, i.e., $A \subset (P \times T) \cup (T \times P)$. Places may contain 0 or more tokens. The distribution of tokens over the places constitute the state of the SPN. All possible states constitutes the state space. Tokens may flow from place to place by the firing of transitions. A transition may fire when it is enabled, i.e., when all its so-called input places, contain at least a token. The input places of a transition are those places that are connected to the transition with an arc pointing to the transition. Upon firing, a transition consumes a token from all of its input places, and produces a token in all of its output places, i.e., in those places that can be reached from the transition by outgoing arcs.

When more than 1 transition is enabled at any time, this conflict is solved probabilistically, i.e., one is chosen to fire first, either because its (stochastic) firing time is smaller, or based on priorities or discrete probabilities associated with the transitions. To further ease the description of various system aspects in SPN-terms, inhibitor arcs, from places to transitions, have been proposed that inhibit the firing of a transition as long there are

tokens in the place. Also, enabling function have been introduced that form extra conditions for a transition to be enabled: whenever the “normal” rules state that a transition is enabled, the corresponding enabling function should evaluate to true for the transition to be finally enabled, otherwise the transition is still not enabled. Enabling functions can generally have any form, so that complex conditions can easily be incorporated (as illustrated in the ETPP model).

The firing of a transition either takes no time at all (immediate or instantaneous transitions), or takes an exponentially distributed time. By this choice, an SPN can be mapped onto a finite continuous-time Markov chain (CTMC). Such a finite CTMC is uniquely described by the infinitesimal generator matrix $\mathbf{Q}$ and the initial probability distribution vector $\pi^{(0)}$. Under usually valid ergodicity conditions, the steady-state probability distribution π of such a CTMC can be obtained by solving the linear system of equations

$$\pi \mathbf{Q} = \mathbf{0}, \text{ under the condition } \sum_i \pi_i = 1. \quad (4)$$

This linear system can be solved with various numerical techniques, such as Gauss-Seidel iteration or Successive Over Relaxation. By the sparseness of $\mathbf{Q}$, systems with hundreds of thousands of equations, i.e., states, can currently be handled. From π , many other interesting performance measures can be derived [13].

A.2 Tool support

For various classes of SPN models, software tools have been developed that help in the construction of the SPN models, that do the automatic derivation of the underlying CTMC, i.e., that derive $\mathbf{Q}$ and $\pi^{(0)}$, and that perform the subsequent numerical analysis, i.e., that compute π . More often than not, graphical user-interfaces are provided. The most notable state-of-the-art software tools are, in random order, SPNP [2, 3], GreatSPN [5], DSPNexpress [15, 16] and UltraSAN [6]. Models with hundreds of thousands of states can easily be handled using state-of-the-art numerical techniques [13] which are also integrated in the mentioned tools. For the evaluations presented in this paper, we have used SPN models as supported by the package SPNP [2, 3].

A.3 Timing distributions

In “standard” SPNs, the involved timing distributions are of exponential type, i.e., an enabled transitions takes an exponentially distributed time to fire. In practice, not all timings obey the exponential distribution, most notably not the deterministic cell switching times. There are a number of ways to circumvent the exponentiality assumption. These all lead to different underlying stochastic models. An overview of the implications for the stochastic model, on the basis of model assumptions made at SPN level, has recently been given by Ciardo *et al.* [4]. We here focus on two approaches only, as these are currently the most practically feasible: the method of stages, and the use of so-called DSPNs.

When using the so-called method of stages, all so-called phase-type distributions are allowed. Phase-type distributions are all distributions that can be represented as the

distribution of the time to absorption in an absorbing CTMC [17, Chapter 2]. The class of phase-type distributions is known to be “dense”, i.e., all distribution functions with rational Laplace transform can be approximated, with any degree of accuracy, with a phase-type distribution.

A particular case of phase-type distribution is the Erlang- k distribution, of which we know that the squared coefficient of variation equals $1/k$. Taking k not too small, in general, reasonable approximations to deterministic distributions are the result. The Erlang- k distribution is known to be the simplest phase-type distribution with a squared coefficient of variation as small as $1/k$, i.e., we can not find a smarter choice to reduce the squared coefficient of variation than using Erlang- k distributions.

A disadvantage of this method of state sets is that it enlarges the state space of the underlying CTMC.

The second approach, which does not have the latter disadvantage, is to use so-called deterministic and stochastic Petri nets (DSPNs) [16]. Under structural restrictions, these models allow for the inclusion of deterministically timed transitions next to exponentially timed transitions. Such models are then solved using an embedding approach which requires the solution of the transient behaviour of a part of the CTMC. The structural restrictions, however, are rather strong and preclude usage in the models we have presented.