Modeling and analysis of multi-class threshold-based queues with hysteresis using Stochastic Petri Nets

Louis-Marie Le Ny[1] and Bruno Tuffin[2]

[1] IRISA-Université de Rennes 1
Campus universitaire de Beaulieu
35042 Rennes cedex, France
leny@irisa.fr
[2] IRISA-INRIA
Campus universitaire de Beaulieu
35042 Rennes cedex, France
btuffin@irisa.fr

Abstract. This paper deals with multi-class queueing systems where thresholds are included in order to smooth the variations of throughput and delay by modifying the queue behaviour. Hysteresis is also inserted, so that the control mechanism will not switch too much. One motivation for using multiple classes of customers is its capability to model heterogeneous traffics like data, voice and video. Moreover, threshold queues have many applications in the transport protocols of communication networks. The analysis is done using Stochastic Petri Nets and Fluid Stochastic Petri Nets. This powerful paradigm helps to obtain a very simple representation of the systems and the analysis is transparent using an available Petri net package. Numerous numerical illustrations are given in order to validate the use of threshold queues with hysteresis as well as their representation by SPNs and FSPNs and performances of various scheduling schemes are compared in order to minimize a cost function.
keywords: Hysteresis, Threshold queues, Performance analysis, Stochastic Petri Nets, Fluid Stochastic Petri Nets.

1 Introduction

We consider in this paper threshold-based queues with hysteresis and mutiple classes of customers. This kind of queueing system has many applications in the dynamic control of computer systems and telecommunication networks. For instance, congestion control and traffic shaping is a big challenge in DiffServ, the service differentiation which will drive the future Internet. Indeed, different quality of services (in terms of throughput, delay or jitter for instance) are required for different applications such as video, telephony, email, ftp... Our modelling will help to approach these characteristics. Of course, other applications of multi-class threshold queues with the same control principles can be found, like in [2] where it is applied to the signaling system no. 7.

Basically, the queues are controlled by a sequence of forward thresholds and a sequence of backward thresholds such that when the content of the queue exceeds a forward threshold, the services of some classes are speeded up. In the same way, when a backward threshold is reached, the services are slowed down. Due to the existence of different types of traffic, each queue is split into several independent queues, one for each type, and the server is allocated to one of these queues according to a given policy (some policies will be described later). Some delay may happen, for example when some management is necessary before modifying the parameter of the queue services.

Mono-class threshold-based queueing systems have been extensively studied in the literature, without hysteresis first in [17, 21, 23] and with hysteresis next [16, 18, 20, 22]. In the most general setting, the queue is Markovian and a closed-form solution for the steady-state probabilities of a heterogeneous multi-server threshold queue with hysteresis is obtained. The works we are going to describe now are devoted to multi-class queues. In [19], different scheduling policies at an ATM switch are studied; packet loss probabilities and mean waiting times are compared in terms of the traffic load. In [1], threshold policies for a single-server queueing network with two job classes are used in order to devise and implement different policies. Numerical results, obtained with the power series algorithm, show that the model satisfies the requirements of higher-priority traffic while offering acceptable performance for the lower priority one. In [2], Choi *et al.* analyse a multi-class priority queueing system with congestion based on thresholds. Each class is controlled by a triple of thresholds composed of an abatement threshold, an onset threshold and a discard threshold. By then, a priority policy is included in the queue in order to provide a better quality of service to some classes. An algorithm to compute performance measures is given if the packet arrival process is a queue-length dependent Markovian arrival process. In [25], a quantitative study of differentiated services for the Internet (DiffServ) is given; the authors compare the loss and delay behaviors related to two router mechanisms called threshold dropping and priority scheduling. Using an analytical method, they show that a significant improvement in throughput can be achieved, especially when using priority scheduling. Finally, we quote a recent work on multi-class threshold-based queuing system with hysteresis: in [11], the number of servers is controlled by the threshold sequences and a fast and accurate iterative solution for evaluating the system performance under varying thresholds and parameters is given.

Since the above literature shows that an analytic analysis becomes quickly tricky, an alternative way, using Stochastic Petri Nets (SPNs), is attempted in this paper. Indeed, SPNs are proven to be a powerful representation of systems with synchronisation or concurrency and we show that the representation of threshold-based queues is very simple and convenient. When the SPN is Markovian, software packages can provide automated generation and solution of continuous-time Markov chains, otherwise it can be simulated. The general class of SPN we will use is actually Stochastic Reward Nets (SRNs) [4, 6] where each state can be associated to a reward rate and allow some functionalities such as guard functions (to enable or not events), state-dependence of firing times and events and priorities. We will also use a generalization of SRNs, Fluid Stochastic Petri Nets (FSPNs) [7, 15, 27] which is a hybrid extension of SPNs where fluid can model continuous components or approximate discrete states to prevent the explosion of the state-space.

We are going to model our systems by SPNs and their extensions like we have done in the mono-class case in [28]. This work can be related to [13], where a somehow similar multi-class analysis using SPNs has been done but targeted to polling systems.

To analyse our SPN and FSPN models, we are going to use one of the available software packages, SPNP (for *Stochastic Petri Net Package*) [5, 14], a software developed at Duke University. This tool uses analytic-numeric methods to solve Markovian nets and simulate FSPNs and non-Markovian SPNs.

The organization of this paper is the following. Section 2 presents the systems we are going to model and analyse. Section 3 is devoted to a description of SPNs (in a very general setting) and FSPNs and Section 4 presents the description of multi-class threshold queues by means of SPNs and FSPNs. Numerical analysis is provided in Section 5. Finally, we give some conclusions and directions for future work in Section 6.

2 Model definition

We consider here multi-class systems, wich are required when dealing with different QoS levels. We consider a set of L different classes arriving to a one-server queue. The queue capacity is assumed to be finite, with capacity C_l for class-l (but a common capacity can be devised as well). A state of the system is given by the number n_l ($1 \leq l \leq L$) of class-l customers in the system and the class number of the customer in service. In this model, each class is assumed to have a separate buffer. Nevertheless, the queue is not separate in several independent queues (even if it can be) because the service policy depends on the global state of the queue. We can affect for instance a strict priority at each class, but many other policies can be devised. We follow here the general policy of [19] where two-class queues are analysed and where a class-l customer will be served with probability $f_l(i,j)$ when there are i class-1 customers and j class-2 customers. The policies analyzed in [19] are

1. Head of line priority (HOL): $f_1(i,j) = 1$ if $i > 0$.
2. Bernoulli scheme (BS): $f_1(i,j) = p$ and $f_2(i,j) = 1 - p$ (if $i, j > 0$).
3. Queue Length Threshold scheduling (QLT): class-1 customers have strict priority ($f_1(i,j) = 1$) if $j < T$, i.e., if the number of class-2 customers is under a threshold. Above this level, class-2 has strict priority.
4. Queue length threshold on class-2 buffer with Bernoulli scheduling (QB2): A threshold value T is used for the number of queued class-2 customers. We use $f_1(i,j) = 1$ if $i > 0$ and $j < T$ and $f_1(i,j) = p$ if $i > 0$ and $j \geq T$.

5. Queue length threshold on class-1 buffer with Bernoulli scheduling (QB1): the threshold value is now used for class-1 queue. We use $f_1(i,j) = p$ if $0 < i < T$ and $j > 0$ and $f_1(i,j) = 1$ if $i \geq T$ and $j > 0$.

An exact analysis is done in [19] (with two classes) when the model is Markovian. This is very general because policies like head of line priority, Bernoulli scheduling scheme or queue length threshold can be represented using functions f_l as we will see below. This model can easily be generalized to more than two classes.

Indeed, we can define $K-1$ forward and backward multi-dimensional thresholds, defined over

$$\{(n_1, \cdots, n_L) \ : 0 \leq n_l \leq C_l\},$$

having a partial order (even if in the subsequent application the order is actually total) and such that when a threshold is reached, the probability functions $f_l(n_1, \cdots, n_L)$ are modified. We have K probability functions corresponding to the $K-1$ forward thresholds: when the k^{th} forward threshold is reached, probability functions are switched to $f_l^{(k+1)}(\cdot)$ and when the k^{th} backward threshold is reached, probability functions are switched back to $f_l^{(k)}(\cdot)$. It is then a generalization of the work in [19]. Hysteresis comes from the fact that forward and backward thresholds do not have the same values. Then, the number of costful switches (in management operations) is expected to decrease.

The way we introduce hysteresis in the policies of [19] is the following (with $K = 2$):

1. HOL: incorporating hysteresis to HOL scheduling would lead to behave like when incorporating hysteresis to QB1 or QB2, so we do not consider it here.
2. BS: the two thresholds s_1 (backward) and S_1 (forward) depend on the number of class-1 customers in the queue. We take $f_1^{(1)}(i,j) = p_1$ and $f_1^{(2)}(i,j) = p_2$ with $p_2 > p_1$ so that if class-1 queue has a lot of customers, we increase the probability that this class is served.
3. QLT: thresholds depend here only on class-2 buffer content. Two thresholds s_2 (backward) and S_2 (forward) are introduced such that $f_1^{(1)}(i,j) = 1$ and $f_1^{(2)}(i,j) = 0$. It means that until S_2 is reached class-1 has strict priority. Then strict class-2 priority is used until s_2 is hit.
4. QB2: thresholds depend here only on class-2 buffer content. We use $f_1^{(1)}(i,j) = 1$ (if $i > 0$) and $f_1^{(2)}(i,j) = p$ (if $i > 0$). We then have two thresholds instead of one in [19] and the probability function used (when between those thresholds) depends on the last threshold hit.
5. QB1: thresholds depend here only on class-1 buffer content. We use $f_1^{(1)}(i,j) = p$ if $i, j > 0$ and $f_1^{(2)}(i,j) = 1$ if $j > 0$ and the scheme works like QB2.

Inter-arrivals and services will first be assumed to be exponential. Even if this exponential assumption is valid for some applications such as telephone networks, many application areas need more general distribution assumptions. In the ATM and Internet networks for instance, services should be assumed to be determistic; moreover, arrivals often present self-similar or heavy-tailed behavior (see for instance [8]). We will study these non-Markovian threshold-queues in the following. As another generalization, there can be some delay between the time that the threshold is reached and the time that the parameters are changed. This can happen for instance when the queue monitors its content at fixed or random intervals of time. These cases will also be investigated numerically.

3 Stochastic Petri Nets, Stochastic Reward Nets and Fluid Stochastic Petri Nets

Petri nets is a formal graph particularly well suited for representing the flow of information and control in systems with concurrency and synchronization characteristics. We describe here an extension of SPNs and SRNs which we will abusively call this model SPN. We then define FSPNs.

The class of stochastic Petri Nets we are considering is given by a tuple

$$(P, T, D, D^0, g, m^0, >, d, \omega, r, F)$$

where

- P is the set of places (represented by circles). Each place may contain *tokens*. The *marking* of the SPN is then defined by the number of tokens in each place. Let $\mathcal{S}$ be the set of markings
- T is the set of transitions (represented by bars).
- $D : (((P \times T) \cup (T \times P)) \times \mathcal{S}) \to \mathbb{N}$ is the set of input arcs (from a place to a transition) and output arcs (from a transition to a place). Each arc is given by its (marking-dependent) multiplicity.
- $D^0 : (P \times T \times \mathcal{S}) \to \mathbb{N}$ is the set of inhibitor arcs, from a place to a transition (represented by an arc terminated by a small circle), with its associate multiplicity.
- $g : (T \times \mathcal{S}) \to \{0, 1\}$ is the (marking-dependent) guard function for each transition. It is a generalization of inhibitor arcs, nevertheless we still also consider inhibitor arcs for their graphical usefulness and for sake of generality.
- $m^0 \in \mathbb{N}^{|P|}$ is the initial marking.
- $>: T \to \mathbb{N}$ is the explicit priority to transitions.
- $d : T \to \{\text{distributions}\}$ defines the firing time distribution of each transition (note that it includes immediate transitions).
- $\omega : (T \times \mathcal{S}) \to \mathbb{R}^+$ is a weight function used to choose the one which will fire if several have the same firing time.
- $r : T \to \{\text{policies}\}$ is the static resampling policy for each transition when it becomes enabled again after being disabled by the firing of a concurrent transition. Three policies are possible: PRI (preemptive repeat identical), PRD (preemptive repeat different) and PRS (preemptive resume). The default value is PRD

- $F : (T \times T) \rightarrow$ {policies} defines the affecting resampling policy for each transition when another transition fires but the considered transition remains enabled. PRI, PRS and PRD are also possible and the default value is PRS.

The dynamics of SPNs is the following. A transition is said to be *enabled* if each of its input places contains as many tokens as the multiplicity of the corresponding arc and if the next two properties are verified. First the *inhibitor arcs* (if any), must verify that each inhibitor place must contain a number of tokens less than the multiplicity of the corresponding arc. The second required property is that the value of the guard function of the transition (if any) is 1. Among all enabled transitions, the one with the smallest firing time is said to *fire*. The firing times of transitions follow random distributions which can be general (i.e., not only exponential). If several transitions have the same firing time, the one which will actually fire is sampled with a probability equal to its individual weight divided by the sum of weights of all enabled transitions. When it fires, the transition removes a number of tokens from each of its input places equal to the multiplicity of the corresponding arc and it deposits in each of its output place a number of tokens equal to the multiplicity of the corresponding arc, leading then to a new marking.

An FSPN [7, 15, 27] is an extension of an SPN where some places can be fluid and where some fluid can flow through transitions, following given rules. Fluid places are represented by two concentric circles and fluid arcs by directed thick arcs. The dynamics of an FSPN is then governed by discrete events (firing of transitions modifying the discrete and fluid marking) and continuous flows in fluid places between these firings as described in the following. When the transitions related to fluid arcs (input or output) are enabled, fluid flows along the arc with the constraint that the fluid level remains between 0 and a prespecified bound. For computational simplicity, the fluid flow rate is at most linear: for each fluid place p, the fluid level $x_p(t)$ changes according to the equation [7]

$$\frac{dx_p(t)}{dt} = a(m)x_p(t) + b(m)$$

where m is the current discrete marking and a and b are some functions of the discrete marking. Moreover, when a transition fires, some fluid can be immediately put in or removed from fluid places. The arcs related to these marking-dependent fluid impulses of the input and output connecting transitions and fluid places are represented by thin directed arcs like the arcs connecting transitions and discrete places. Of course, the initial marking x^0 in fluid places, needs also to be specified.

4 Representation by means of SPNs and FSPNs

4.1 Markovian queue

Consider a two-class system, where, for instance, class-1 is for demanding service (such as video or real-time traffic) and class-2 for less stringent service (like e-mail,...). It seems better to give a higher service probability (or a strict priority) to the class-1 customers, especially when there are many class-1 customers waiting in the buffer. Figure 1 represents such a system by a Markovian SPN with rate functions and guard functions described in Tables 1 and 2. A generalization to more classes is straightforward. The number of tokens place $Pcont$ gives the control integer k such that probability functions $f_l^{(k)}(\cdot)$ are used (which, in Petri Net formalism, are actually the weight functions). Of course, $f_1^{(k)}(n_1, n_2) + f_2^{(k)}(n_1, n_2) = 1$ $(k = 1, 2)$ $\forall n_1, n_2$. Transitions $Tinc$ and $Tdec$ fire only when forward and backward thresholds are hit. In this example, thresholds depend only on class 1 queue but it can easily be made more general (by modifying the guard functions of transitions $Tinc$ and $Tdec$ in Table 2). Transitions

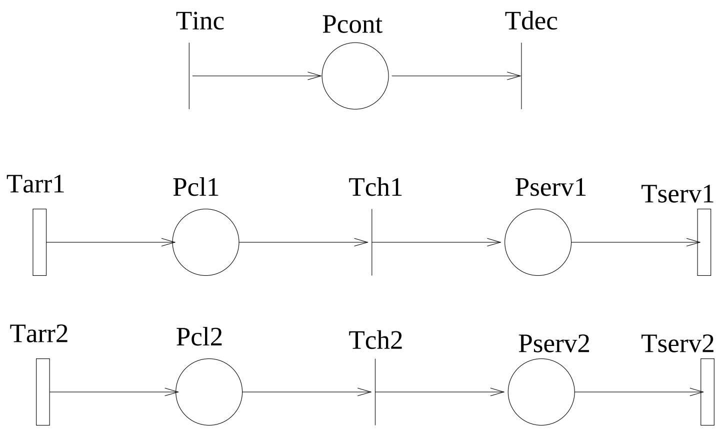

Fig. 1. Representation of a two-class threshold-queue with hysteresis by an SPN.

Transition	Rate
$Tarr1$	λ_1
$Tarr2$	λ_2
$Tserv1$	μ
$Tserv2$	μ

Table 1. rate functions of the SPN of Figure 1

$Tarr1$ and $Tarr2$ are for arrivals of class-1 and class-2 customers respectively (of course their rates can be made dependent of an external stochastic process like On/Off sources). The guards on these transitions are such that there is a separate buffer for each class (of respective capacity $C1$ and $C2$). This can be easily

Transition	guard
$Tinc$	$\#Pcl1 + \#Pserv1 > S_{\#Pcont}$ && $\#Pcont < K$
$Tdec$	$\#Pcl1 + \#Pserv1 \leq s_{\#Pcont-1}$ && $\#Pcont > 0$
$Tarr1$	$\#Pcl1 < C1$
$Tarr2$	$\#Pcl2 < C2$
$Tch1$	$\#Pserv1 == 0$ && $\#Pserv2 == 0$
$Tch2$	$\#Pserv1 == 0$ && $\#Pserv2 == 0$

Table 2. Guard functions of the SPN of Figure 1

changed to a shared buffer by setting the guard function of $Tarr1$ and $Tarr2$ to $\#Pcl1 + \#Pcl2 < C$. Places $Pcl1$ and $Pcl2$ are for class-1 and class-2 customers waiting to be served. Places $Pserv1$ and $Pserv2$ give the class of the customer in service. Only one token can be in $Pserv1$ and $Pserv2$ (exclusively), this being controlled by using the guard on transitions $Tch1$ and $Tch2$ giving the class to be served. When the customer is served (transitions $Tserv1$ and $Tserv2$), the new customer class to be served is chosen using the probabilities/weights $f_l^{(k)}(n_1, n_2)$ as defined in Table 3.

Transition	probability (or weight)
$Tch1$	$f_1^{(\#Pcont)}(\#Pcl1 + \#Pserv1, \#Pcl2 + \#Pserv2)$
$Tch2$	$f_2^{(\#Pcont)}(\#Pcl1 + \#Pserv1, \#Pcl2 + \#Pserv2)$

Table 3. Weight functions of the SPN of Figure 1

4.2 General queue

In the non-Markovian case, marking-dependent exponential distributions of transitions $Tarr1$, $Tarr2$, $Tserv1$ and $Tserv2$ are replaced by marking-dependent general distributions. We introduce two places $Pentrance1$ and $Pentrance2$ in order to model arrivals (admitted or not) and to be able to compute loss probabilities (because PASTA property does not apply anymore). The model is described in Figure 2. With respect to the model of the previous subsection, rates ar replaced by general distributions, and we introduce guard functions for arrivals at the queue in Table 4, meaning that the new token/customer/packet is admitted if the queue is not full. These guard functions can be modified in order to apply admission control.

4.3 Queue with management delay

Delay can be introduced when the system monitors the state of the queue only at fixed or random instants of time. Thus, the policy is modified only when

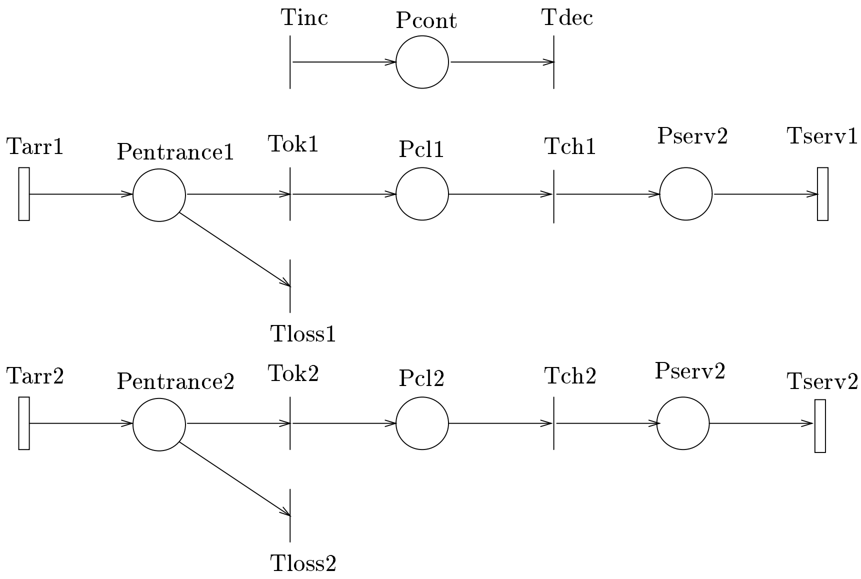

Fig. 2. Queue with general distributions

Transition	guard
$Tloss1$	$\#Pcl1 == C$
$Tok1$	$\#Pcl1 < C$
$Tloss2$	$\#Pcl2 == C$
$Tok2$	$\#Pcl2 < C$

Table 4. Supplementary Guard functions of the SPN of Figure 2

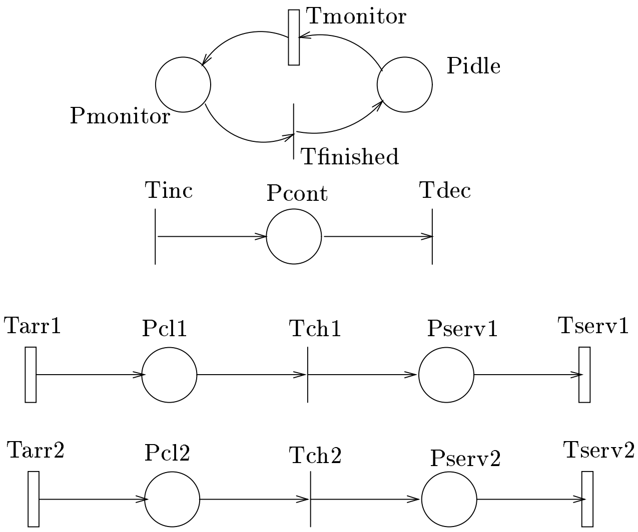

Fig. 3. A queue with delay

the monitoring is done. Figure 3 represents such a queue. The system behaves like in the non-delayed case but supplementary conditions are introduced on the guard functions of transitions $Tinc$ and $Tdec$ which represent the switches in the service policy. These two transitions are enabled only if the monitoring has just been done, i.e., if there is a token in place *Pmonitor*. Then, immediate transitions $Tinc$ and $Tdec$ can fire (if the other conditions on the guard functions are fulfilled). Next, transition $Tfinished$ fires so that the system waits for another period of time to monitor the queue. Among possible concurrent transitions, a higher priority is given to transitions $Tinc$ and $Tdec$ with respect to $Tfinished$ to ensure that the service modification is applied before place *Pmonitor* becomes idle again. The new guard functions are given in Table 5. Other parameters are defined like in the Markovian non-delayed case, but we can also generalize the case where inter-arrivals and/or services do not follow an exponential distribution.

Transition	guard
$Tinc$	$\#Pcl1 + \#Pserv1 > S_{\#Pcont}$ && $\#Pcont < K$ && #Pmonitor==1
$Tdec$	$\#Pcl1 + \#Pserv1 \leq s_{\#Pcont-1}$ && $\#Pcont > 0$ && $\#Pmonitor == 1$
$Tarr1$	$\#Pcl1 < C1$
$Tarr2$	$\#Pcl2 < C2$
$Tch1$	$\#Pserv1 == 0$ && $\#Pserv2 == 0$
$Tch2$	$\#Pserv1 == 0$ && $\#Pserv2 == 0$

Table 5. Guard functions of the SPN of Figure 3

4.4 Fluid queue

We consider here a fluid queue driven by *On-Off* sources, but other driving processes can be used as well. Discrete arrivals and services are aggregated in a continuous flow. This is a good approximation of ATM traffic for instance. The FSPN modelling this system is described in Figure 4 with guard functions and fluid rate functions described respectively in Tables 6 and 7. The arrival fluid rates are made dependent of the number of *On* sources for each class. If the buffer is full (i.e., $\#Psyst1 + \#Psyst2 = C$), these rates are also bounded by the current service rate (i.e., the remaining flow is lost). The service fluid rates are such that the proportion p and $1-p$ of served class-1 and class-2 fluid depend on the hysteretic control variable like in the discrete case. Here, if a place is empty, the fluid arrival rate is bounded by the service fluid rate. Also, fluid rates are optimized in such a way that if the buffer is empty for a given class, the other class is served instead. We have made the thresholds dependent of the class-1

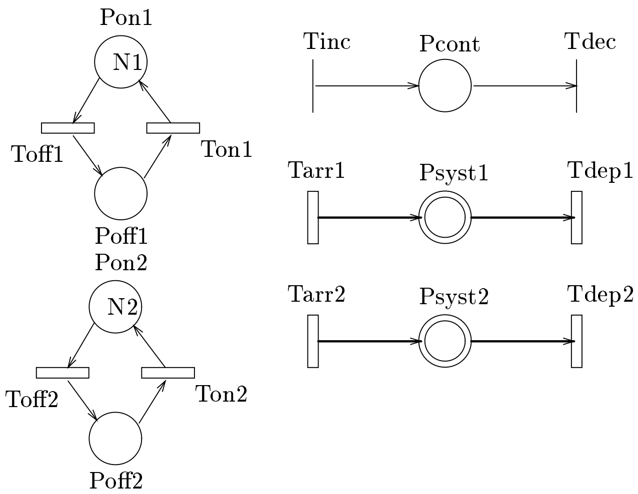

Fig. 4. Two-class fluid queue

Transition	guard
Tinc	$\#Psyst1 > S_{\#Pcont}$ && $\#Pcont < K$ && $r(Tarr1) > r(Tdep1)$
Tdec	$\#Psyst1 \leq s_{\#Pcont-1}$ && $\#Pcont > 0$ && $r(Tarr1) < r(Tdep1)$

Table 6. Guard functions of the SPN of Figure 4

fluid level, but we can make it dependent of the class-2 fluid level or of a linear combination of both as well.

Transition	Fluid rate $r(\text{Transition})$
$Tarr1$	$\lambda 1 \#Pon1\ 1_{[\#Psyst1+\#Psyst2<C]} + \min(\lambda 1 \#Pon1, r(\text{Tdep1}))1_{[\#Psyst1+\#Psyst2=C]}$
$Tarr2$	$\lambda 2 \#Pon2\ 1_{[\#Psyst1+\#Psyst2<C]} + \min(\lambda 2 \#Pon2, r(\text{Tdep1}))1_{[\#Psyst1+\#Psyst2=C]}$
$Tdep1$	$\mu p(\#Pcont)$ if $\#Psyst1, \#Psyst2 > 0$ $\min(\lambda_1 \#Pon1, \mu p(\#Pcont))$ if $\#Psyst1 = 0,\ \#Psyst2 > 0$ $\mu - \min(\lambda_2 \#Pon2, \mu(1 - p(\#Pcont)))$ if $\#Psyst1 > 0,\ \#Psyst2 = 0$ $\min(\lambda_1 \#Pon1, \mu - \min(\lambda_2 \#Pon2, \mu(1 - p(\#Pcont))))$ if $\#Psyst1, \#Psyst2 = 0$
$Tdep2$	$\mu(1 - p(\#Pcont))$ if $\#Psyst2, \#Psyst1 > 0$ $\min(\lambda_2 \#Pon2, \mu(1 - p(\#Pcont)))$ if $\#Psyst2 = 0,\ \#Psyst1 > 0$ $\mu - \min(\lambda_1 \#Pon1, \mu p(\#Pcont))$ if $\#Psyst2 > 0,\ \#Psyst1 = 0$ $\min(\lambda_2 \#Pon2, \mu - \min(\lambda_1 \#Pon1, \mu p(\#Pcont)))$ if $\#Psyst2, \#Psyst1 = 0$

Table 7. Fluid rate functions of the SPN of Figure 4

5 Analysis

5.1 Solution methods

The solution methods we are going to use are those implemented in SPNP [5, 14], a package developed at Duke University, and are the following.

Markovian SPNs of the previous sections are solved analytically-numerically. The reachability graph is first generated from the SPN and then converted to a Markov chain (or more exaclty to a Markov reward model) which is solved numerically. For steady-state evaluation, we will choose Steady-State SOR (Successive Overrelaxation [26]), the fastest method, but Steady-State Gauss-Seidel [26], and Steady-State Power method [30] are also avaible in SPNP and are used when SSOR does not converge. For transient-state solution of the CTMC, standard uniformization or uniformization using the Fox and Glynn method for computing the Poisson probabilities can be used.

On the other hand, non-Markovian SPNs and FSPNs are solved by simulation (because it is currently the only method implemented in SPNP). Nevertheless, there exists some numerical analysis algorithms solving FSPNs when there are one (see for instance [15]) or at most two fluid places in the system [31]. Steady-state simulation will be performed using simulation with batches or regenerative simulation [9] (in this last case, our SPN needs to be in the class of MRSPNs [3]). Transient simulation can be performed by crude independent replications. However, there are other simulation methods available in SPNP: importance splitting techniques (Restart and splitting) [29], importance sampling [9, 10, 24], regenerative simulation with importance sampling [12, 24], all suitable to estimate rare events.

5.2 Numerical results

We consider a two-class queue like in Figure 1. We assume that class-1 traffic is for real-time applications and is more important than class-2 traffic. The queue capacities are $C_1 = C_2 = 64$; the thresholds are $s_1 = 24$ and $S_1 = 48$ in the hysteresis case and $S = (S_1 = s_1 =)36$ if the policy does not require hysteresis. The cost function we use here is divided in two parts, the network cost and the user cost. The user cost function, representing what is felt by the user application, is

$$C_{user} = c_1 P(\text{Class-1 loss}) + c_1' P(\text{Class-2 loss}) + c_2 P(\text{queue 1 empty}) \\ + c_3 g_1(m_1) + c_3' g_2(m_2)$$

where function $g_1(m_1)$ for class-1 depending on the number m_1 of class-1 customers in the queue is hyperbolic:

$$g_1(m) = \begin{cases} \frac{64}{3m+16} & \text{if } m \leq 16, \\ \frac{64}{3(64-m)+16} & \text{if } m \geq 48, \\ 1 & elsewhere. \end{cases}$$

By then, the queue is penalized if it is close to be full or close to starvation. Function $g_2(m_2)$ for class-2 has an hyperbolic arc only when the class-2 buffer is close to be full (if it is empty, it does not matter). The network cost function, representing the additional costs in order to improve the QoS without any perception by the user, is

$$C_{net} = c_4 \mu P(\text{server used}) + c_5 \alpha_u + c_6 \alpha_d$$

where α_u (resp. α_d) is the mean number of forward switches (resp. backward switches) per unit time when changing of policy (i.e., of values f_k). Note that $\alpha_u = \alpha_d$ in steady state. For our examples, we arbitrarily choose the weights to be $c_1 = 40$, $c_1' = 4$, $c_2 = 15$, $c_3 = 0.8$, $c_4 = 0.05$, $c_5 = 0.4$, $c_6 = 0.4$ and $c_3' = 0.2$. Moreover, the service rate is $\mu = 10.0$ and the total arrival rate is λ so that $\lambda_1 = \lambda r$ and $\lambda_2 = \lambda(1-r)$ where r is the fraction of traffic allocated to class-1. As we are interested in steady-state analysis, the initial marking specification is not necessary.

Consider first a Markovian model. The (exact) results are displayed by means of 3D-surfaces; in this way, one can easily see the lowest cost and hence choose the best policy with respect to the parameters p and r. Only the schemes showing the best performance are displayed in order to keep the surfaces clear. Recall that p denotes the probability for serving a class-1 customer first and r is the proportion of class-1 arriving customers. For comparison sake, in the case of hysteresis, we take $p_1 = p$ and $p_2 = p + 0.2$; therefore class-1 is more likely to be served if its buffer has passed the forward threshold. We are going to observe the behavior of the schemes when the total traffic increases, i.e., for different values of λ.

In Figure 5 the Bernoulli scheduling with and without hysteresis are investigated under light traffic ($\rho = 0.3$) for which they are giving the best results. The costs are computed for $p = 0.3$, $p = 0.5$ and $p = 0.7$ whereas the value of r ranges from 0.1 to 0.9 with a step of 0.1. The surfaces are obtained using the cubic spline method available in the MATLAB package. Figure 5 shows that the best policy is BS (Bernoulli scheduling without hysteresis) under some conditions on the values of r and p:

- the ratio r of class-1 customers is high (beyond 0.8).
- r is between 0.4 and 0.8, and p is close to 0.5.
- $r < 0.4$ and $p > 0.2$.

In the other cases, the policy BShy (Bernoulli scheduling with hysteresis) performs better. If one wants to pull out a simple and approximative rule, we can suggest to use BS if $p < r$ and BShy otherwise. Note that higher costs for small values of r (and λ) are due to queue starvation which happens often, due to the light traffic.

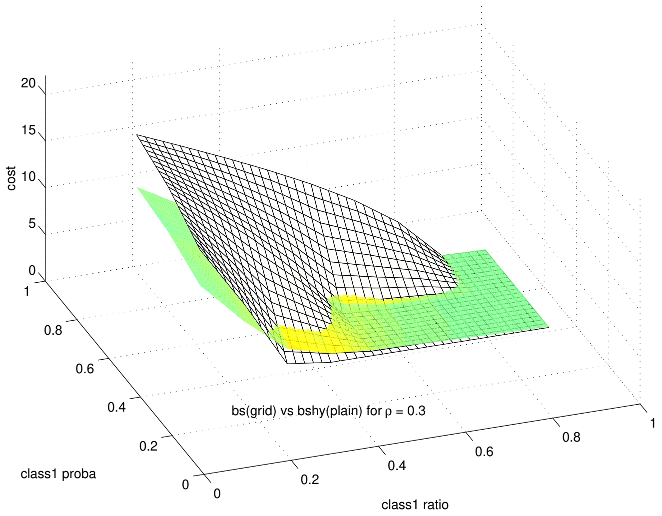

Fig. 5. Cost comparisons between policies Bernoulli Scheduling (BS) and BS with Hysteresis (BShy) when $\rho = 0.3$

Taking $\rho = 0.5$ (see Figure 6) the same two policies give the best results. The rule (using BS if $p < r$ and BShy otherwise) is still valid since the frontier is clearer. In heavier traffic (Figure 7), for $\rho = 0.8$, the best policies are Bernoulli Scheduling with hysteresis (BShy) or QB1 with hysteresis (QB1hy) depending on the values of p and r. If p is low (less than 0.3), for any value of r, the less expensive policy is QB1hy, whereas if p is higher the cost values are almost the same and it is thus possible to keep the QB1hy policy. Finally, in Figure 8, we consider a congestion case, i.e., $\rho = 1.1$. Again, we almost have that, if $p < r$, the best policy is BS with hysteresis whereas if $p > r$, QB1 with hysteresis performs better. All the above results are summarized in Table 8. Of course, this rule is valid only for our cost function and numerical values, but it shows how a rule

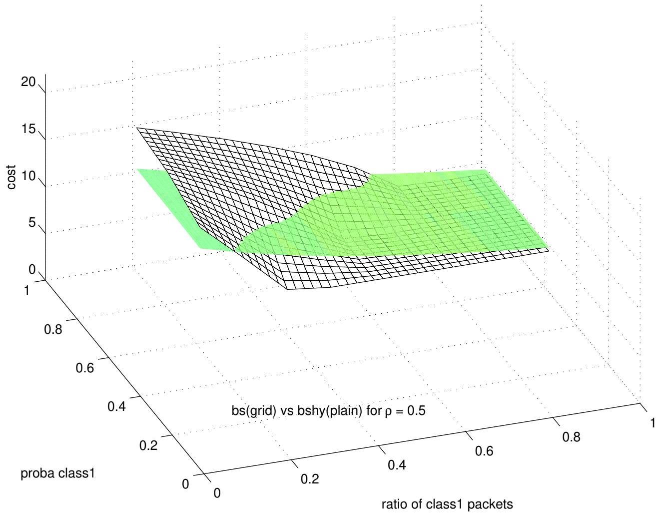

Fig. 6. Cost comparisons between policies Bernoulli Scheduling (BS) and BS with Hysteresis (BShy) for $\rho = 0.5$

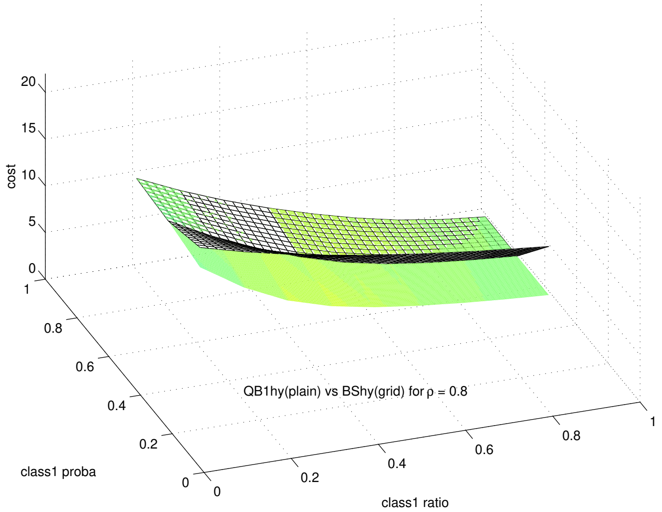

Fig. 7. Cost comparisons between policies Bernoulli Scheduling with Hysteresis (BShy) and QB1 with Hysteresis (QB1hy) when $\rho = 0.8$

ρ	0.3	0.5	0.8	1.1
the best policy	BS ($p < r$) BShy ($p > r$)	BS ($p < r$) BShy ($p > r$)	QB1hy	QB1hy ($p > r$) BShy ($p < r$)

Table 8. Optimal policies with respect to r, p, and ρ for the Markovian queue.

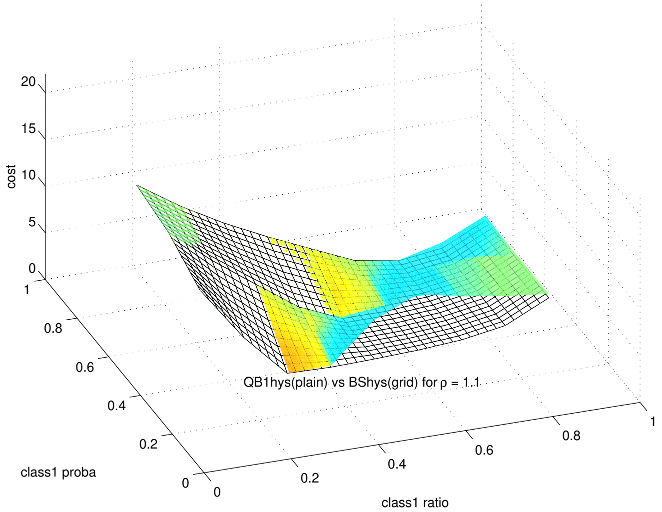

Fig. 8. Cost comparisons between policies Bernoulli Scheduling with Hysteresis (B-Shys) and QB1 with Hysteresis (QB1hys) when $\rho = 1.1$

can be displayed using our modeling in order to choose the best service policy with respect to traffic parameters.

We now study non-Markovian queues. The parameters used are deterministic services with value 5.0, inter-arrivals following Pareto distributions (with shape parameter 0.05/2.1 and smallest value 0), so that the total traffic intensity is $\rho = 0.5$. We give in Table 9 numerical results showing that the previous Markovian rule does not apply here as QB1hy performs the best. It means that another rule has to be devised by running the model when the parameters varies.

Table 10 shows, using QB1hy scheduling of the previous non-Markovian queue that delay can slightly modify the performance of the system (compared with the same policy in Table 9). We are using here an inter-monitoring time interval following an exponential distribution with rate 1.0, the other parameters being exactly the same than in the above general queue.

Consider now the system as a fluid queue. Table 11 displays the results obtained for QB1hy scheduling when $\rho = 0.5$ and $p = 0.5$. The parameters are here $C_1 = C_2 = 2.0$, $p = 0.5$, the service fluid rate is $\mu = 4.4$; there are 10 On-Off sources for each class, the fluid arrival rate being $\lambda = 0.4$ per source for both classes. The thresholds are $S_1 = 1.5$, $s_1 = 0.5$. Note that the expression of costs have to be modified in order to fit the fluid modelling. The loss probability is computed as the ratio of the mean feeding fluid rate $\lambda E(\#Pon)$ minus the mean of $r(Tarr)$ on $\lambda E(\#Pon)$ (and the empty probability is computed using the same principle). The hyperbolic cost is the one which has to undergo numerous adaptations. We precomputed the different possibilities of integrals of $g_1(\cdot)$ and $g_2(\cdot)$ between two instants of time and added them to the cost at each step of

Scheme	user cost	network cost	total cost
HOL	[40.670,42.477]	[0.19742,0.20194]	[40.870,42.677]
BS, $p = 0.3$	[54.210,56.897]	[0.18022,0.18436]	[54.392,57.080]
BS, $p = 0.7$	[47.816,50.008]	[0.18356,0.18763]	[48.001,50.194]
QLT	[67.670,70.987]	[0.19487,0.19939]	[67.867,71.185]
QB1, $p = 0.7$	[41.514,43.351]	[0.21275,0.21791]	[41.729,43.567]
QB2, $p = 0.7$	[54.197,56.562]	[0.22117,0.22590]	[54.420,56.786]
BShy, $p = 0.3$	[56.479,59.065]	[0.20213,0.20617]	[56.683,59.270]
BShy, $p = 0.7$	[48.879,50.961]	[0.21360,0.21758]	[49.094,51.177]
QLThy	[64.239,67.430]	[0.19020,0.19461]	[64.431,67.623]
QB1hy, $p = 0.7$	[38.449,40.093]	[0.19392,0.19824]	[38.645,40.289]
QB2hy, $p = 0.7$	[44.439,46.513]	[0.18991,0.19436]	[44.631,46.706]

Table 9. Results obtained for the general queue when $\rho = 0.5$ and $r = 0.6$

Scheme	user cost	network cost	total cost
QB1hy $p = 0.7$	[40.904,42.733]	[0.18983,0.19422]	[41.096,42.926]

Table 10. Results obtained for the general queue with delay when $\rho = 0.5$ and $r = 0.6$ using QB1hy scheduling

the discrete event simulation. The hyperbolic curves are on intervals $[0, 0.4]$ and $[1.6, 2]$.

6 Conclusion

Modeling and analysis of threshold-based queues using SPNs and FSPNs is very convenient and SPNP tool (for instance) allows to do it quickly. This can aid in the study and configuration of the queue to conform to design specifications. Multi-class queues are important due to their applications in the next Internet generation which must include service differentiation. We have modeled those queues in several situations (Markovian, non-Markovian, with delay, fluid) and for several service policies between classes. By varying parameters, the model can specify the best schedulling policy. Moreover, if the network knows the traffic intensity at different times of day, it can apply different policies in order to improve the performance.

As directions for future work, we can enhance:

- Test and devise protocols where the nodes of the network notify to the sources when and how to speed-up or slow down their transmission rates.
- Using Colored SPNs and devise Colored Fluid Stochastic Petri Nets in order to separate the flows of different sources and to analyse the behaviour of each traffic stream. Moreover, color will be a more natural way to model multi-class systems.

Scheme	user cost	network cost	total cost
QB1hy $p = 0.5$	[5.2649,5.6235]	[5.5849e-01,5.9072e-01]	[5.8353,6.2023]

Table 11. Results obtained for the fluid queue using QB1hy scheduling

Acknowledgement

The authors would like to thank the anonymous reviewers for their suggestions and comments.

References

1. K.D. Ansell, P.S. Glazebrook and I. Mitrani. Threshold policies for a single-server queuing network. *Probability in the Engineering and Informational Sciences*, 15:15–33, 2001.
2. B.D. Choi, S.H. Choi, B. Kim, and D.K. Sung. Analysis of priority queueing system based on thresholds and its application to signaling system no. 7 with congestion control. *Computer Networks*, 32:149–170, 2000.
3. H. Choi, V.G. Kulkarni, and K.S. Trivedi. Markov Regenerative Stochastic Petri Nets. *Performance Evaluation*, 20(1-3):337–357, 1993.
4. G. Ciardo, A. Blakemore, P.F. Chimento, J.K. Muppala, and K.S. Trivedi. Automated Generation and Analysis of Markov Reward Models using Stochatic Reward Nets. In Carl Meyer and Robert Plemmons, editors, *Linear Algebra, Markov Chains and Queuing Models*, volume 48 of *IMA Volumes in Mathematics and its Applications*, pages 145–191. Springer-Verlag, Heidelberg, 1993.
5. G. Ciardo, J.K. Muppala, and K.S. Trivedi. SPNP: Stochastic Petri net Package. In *Proc. Third International Workshop on Petri Nets and Performance Models, PNPM'89*, pages 142–151, 1989.
6. G. Ciardo, J.K. Muppala, and K.S. Trivedi. Analyzing Concurrent and Fault-tolerant Software Using Stochatic Reward Nets. *Journal of Parallel and Distributed Computing*, 15:255–269, 1992.
7. G. Ciardo, D.M. Nicol, and K.S. Trivedi. Discrete-Event Simulation of Fluid Stochastic Petri-Nets. *IEEE Transactions on Software Engineering*, 25(2):207–217, 1999.
8. M.E. Crovella. Performance Characteristics of the World Wide Web. In G. Haring et al., editor, *Performance Evaluation*, volume 1769 of *Lecture Notes in Computer Science*, pages 219–232. Springer-Verlag, 2000.
9. G.S. Fishman. *Monte Carlo: Concepts, Algorithms and Applications*. Springer-Verlag, 1997.
10. P. W. Glynn and D. L. Iglehart. Importance Sampling for Stochastic Simulations. *Management Science*, 35(11):1367–1392, November 1989.
11. L. Golubchik and J.C.S. Lui. A fast and accurate iterative solution of a multi-class threshold-based queuing system with hysteresis. In *Proc. of the ACM SIGMETRICS Conference*, pages 196–206, Santa Clara, June 2000.
12. A. Goyal, P. Shahabuddin, P. Heidelberger, V.F. Nicola, and P.W. Glynn. A Unified Framework for Simulating Markovian Models of Highly Dependable Systems. *IEEE Transactions on Computers*, 41(1):36–51, January 1992.

13. B.R. Haverkort. *Performance of Computer Communication Systems*. John Wiley and Sons, 1998.
14. C. Hirel, B. Tuffin, and K.S. Trivedi. SPNP Version 6.0. In B.R. Haverkort, H.C. Bohnenkamp, and C.U. Smith, editors, *Computer performance evaluation: Modelling tools and techniques; 11th International Conference; TOOLS 2000, Schaumburg, Il., USA*, volume 1786 of *Lecture Notes in Computer Science*, pages 354–357. Springer Verlag, 2000.
15. G. Horton, V. Kulkarni, D. Nicol, and K.S. Trivedi. Fluid Stochastic Petri nets: Theory, Application and Solution. *European Journal of Operational Research*, 105:184–201, 1998.
16. O.C. Ibe and J. Keilson. Multi-server threshold queues with hysteresis. *Performance Evaluation*, 21:185–213, 1995.
17. R.L. Larsen and A.K. Agrawala. Control of a Heterogeneous Two-Server Exponential Queueing System. *IEEE Transactions on Software Engineering*, SE-9(4):522–526, 1983.
18. L-M. Le Ny and B. Tuffin. A simple analysis of heterogeneous multi-server threshold queues with hysteresis. Technical Report 1333, IRISA, 2000.
19. J.Y. Lee and Y.H. Kim. Performance analysis of a hybrid priority control scheme for input and output queueing ATM switches. In *Proceedings of IEEE INFOCOM 98*, pages 1470–1477, March 1998.
20. S.Q. Li. Overload Control in a Finite Message Storage Buffer. *IEEE Transactions on Communications*, 37(12), December 1989.
21. W. Lin and P.R. Kumar. Optimal Control of a Queueing System with Two Heterogeneous Servers. *IEEE Transactions on Automatic Control*, AC-29(8):696–703, 1984.
22. J.C.S. Lui and L. Golubchik. Stochastic complement analysis of multi-server threshold queues with hysteresis. *Performance Evaluation*, 35:185–213, 1999.
23. J.A. Morrison. Two-server queue with one server idle below a threshold. *Queueing Systems: Theory and Applications*, 7:325–336, 1990.
24. V. F. Nicola, M. K. Nakayama, P. Heidelberger, and A. Goyal. Fast Simulation of Highly Dependable Systems with General Failure and Repair Processes. *IEEE Transactions on Computers*, 42(12):1440–1452, December 1993.
25. D. Sahu, S. Towsley and J. Kurose. A Quantitative Study of Differentiated Services for the Internet. *Journal of Communications and Networks*, 2:127–137, 2000.
26. W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.
27. K. S. Trivedi and V. G. Kulkarni. FSPNs: Fluid Stochastic Petri Nets. In *14th International Conference on Applications and Theory of Petri Nets*, pages 24–31, 1993.
28. B. Tuffin and L-M. Le Ny. Modeling and analysis of threshold queues with hysteresis using stochastic Petri nets: the monoclass case. In *Proceedings of Petri Nets and Performance Models*, pages 175–184. IEEE CS Press, 2001.
29. B. Tuffin and K.S. Trivedi. Implementation of importance splitting techniques in stochastic Petri net package. In B.R. Haverkort, H.C. Bohnenkamp, and C.U. Smith, editors, *Computer performance evaluation: Modelling tools and techniques; 11th International Conference; TOOLS 2000, Schaumburg, Il., USA*, volume 1786 of *Lecture Notes in Computer Science*, pages 216–229. Springer Verlag, 2000.
30. W. B. van den Hout. *The Power-Series Algorithm: A Numerical Approach to Markov Processes*. Tilburg University, 1996.
31. K. Wolter. *Performance and Dependability Modelling with Second Order Fluid Stochastic Petri Nets*. PhD thesis, Technische Universitat Berlin, 1999.