Matrix-Geometric Solution of Infinite Stochastic Petri Nets

Boudewijn R. Haverkort
University of Twente, Department of Computer Science
P.O. Box 217, 7500 AE Enschede, the Netherlands
E-mail: b.r.h.m.haverkort@cs.utwente.nl

Abstract

In this paper we characterize a class of stochastic Petri nets that can be solved using matrix geometric techniques. Advantages of such an approach are that very efficient mathematical technique become available for practical usage, as well as that the problem of large state spaces can be circumvented.

We first characterize the class of stochastic Petri nets of interest by formally defining a number of constraints that have to be fulfilled. We then discuss the matrix geometric solution technique that can be employed and present some boundary conditions on tool support. We illustrate the practical usage of the class of stochastic Petri nets with two examples: a queueing system with delayed service and a model of connection management in ATM networks.

1 Introduction

Stochastic Petri nets (SPNs) have been used extensively for the performance evaluation of computer and communication systems as well as of flexible manufacturing systems. Many variants have been proposed: stochastic Petri nets (SPNs) [26], generalized stochastic Petri nets [1], stochastic activity networks (SANs) [31], stochastic reward nets (SRNs) [8, 9], deterministic and stochastic Petri nets (DSPNs) [2, 25] and Markov-regenerative SPNs [7]. An overview of the mathematical differences between these many different variants has recently been presented in [11]. These many different variants have been proposed for various reasons: (i) to increase the modelling flexibility by introducing convenient new constructs such as variable arc multiplicities, gates and enabling functions; (ii) to increase the modelling power by allowing more general timing characterizations as compared to the standard SPNs; or (iii) to introduce more structured models so as to make the solution process of the underlying stochastic process an easier task.

A general problem in employing SPNs is the rapid growth of the state space (the largeness problem). Solutions that have been proposed are often based on (approximate) truncation techniques [20] or lumping [4, 5, 6, 31], thereby still performing the solution at the state space level, or they are based on product-form results which allow for an efficient convolution or mean-value analysis style of solution [12, 14]. In all these cases, the "trick" lies in trying to circumvent the generation of the overall state space before solution, or to decrease the size of the state space. In that respect, also simulation avoids the generation of the overall state space.

Besides the above, we should also consider the following. In queueing theory, the steady-state analysis of infinitely large systems is often simpler than analyzing the corresponding finite systems. These considerations have lead to the following approach to tackle the problem of very large but finite state spaces. Instead of generating and solving a very large but finite SPN model, we solve infinitely large CTMCs derived from an SPN model. In doing so, we explicitly avoid generating the overall state space; instead we exploit the repetitive structure of the infinite CTMC. Once we have solved the infinitely large model, we can use the results also for finite versions of the model, albeit not always exactly. Indeed, also in many application areas, such as the analysis of ATM multiplexers, infinitely large models, i.e., infinite buffer models, are used to compute bounds or approximations on finite models [33].

For the infinitely sized models to be easy solvable, we require them to exhibit a regular structure. This limits their applicability, however, it is surprising to see how many models do fulfill these requirements. A class of infinitely large Markovian models which exhibits both an efficient solution, and occurs very often in practice, are the so-called quasi birth-death (QBD) models, as described in [27, 28]. A state-wise specification of such models, however, is cumbersome for many practical applications. We therefore define a class of infinite-state SPNs which has an underlying Markov chain of the QBD type. We discuss the mapping of such SPNs to a compact description of the QBD process and discuss an efficient matrix geometric solution technique.

Important other work in this direction has been re-

0-8186-7059-2/95 $04.00

ported by Florin and Natkin. In [16] they present so-called "one-place unbounded SPNs" and prove the existance of an underlying QBD structure. In [15] they elaborate on ergodicity conditions of such SPNs. The SPNs addressed are restricted in the sense that transitions have constant firing rates, arc multiplicities are always 1 (monovaluated nets), and unbounded places may not be the origin of inhibitor arcs. Then, in [17], they discuss the ergodicity of "multiple-place unbounded SPNs". Also such SPNs may have un underlying QBD, however, when the so-called "marking-space dimension" is larger than 1, the block matrices $\mathbf{A}_0$ through $\mathbf{A}_2$ are infinitely large themselves (see Section 3). Finally, in [18], they discuss matrix product-from solution for closed, i.e., bounded, stochastic Petri nets. Concluding, our proposed class of SPNs is larger than the class proposed by Florin and Natkin (see the rest of this paper), however, this comes at the cost of more difficulty in deriving the underlying QBD structure.

Recent work by Berson and Muntz is also of interest to our work [3]. They present an approach to detect block-M|G|1 and block-G|M|1 structures directly from models specified using a state-machine specification language. For 2-dimensional models they are able to decide on the (in)finiteness and on the block structure of the models, directly from the specification. For larger-dimensional models they can only do so by further restricting the specification language.

Two performance evaluation tools have been reported in the literature that employ matrix geometric techniques. With the tool MAGIC [32], CTMCs with a QBD structure can be defined at the state level and solved efficiently. With the tool Xmgm, higher-level, user-oriented, constructs are provided that allow for a more user-friendly model specification of QBD models [21]. In this paper we propose to go a step further, namely to use SPNs for the specification of a class of models that allows for a matrix geometric solution.

Regarding the applicability of our approach, we see good opportunities in the field of broadband communication systems, such as ATM systems, where relatively simple multiplexers or switches are used in combination with intricate arrival processes. Also in the field of queues with breakdowns, our approach seem to be well applicable. In section 5 we will see examples in these directions.

This paper is further organized as follows. The special class of SPNs is defined in Section 2. The underlying QBD process is described in Section 3, together with an efficient solution technique. A proposal for a tool supporting this class of SPNs is outlined in Section 4. Two applications are presented in Section 5. Section 6 concludes the paper.

2 Infinite stochastic Petri nets

We discuss notation and terminology in Section 2.1. The restricting requirements are given in Section 2.2. They are discussed in Section 2.3.

2.1 Preliminaries

We consider a class of SPNs similar to the class of SRNs as defined by Ciardo *et al.* [8, 9].

Without loss of generality we assume that the SPN under study, denoted *SPN*, has a set $P = \{p_0, p_1, \cdots, p_n\}$ of places of which only p_0 may contain an infinitely large number of tokens. A distribution of tokens over the places is called a marking and denoted $\underline{\mu} = (m_0, \underline{m}) = (m_0, m_1, \cdots, m_n)$. The set of all possible markings is denote $\mathcal{S} = I\!N \times \mathcal{M}$, i.e., $m_0 \in I\!N$ and $\underline{m} \in \mathcal{M}$. Clearly, $|I\!N| = \infty$ and $|\mathcal{M}| < \infty$. The set of transitions is denoted T.

We now define *level* $\mathcal{S}(k)$ to be the set of markings such that place p_0 contains k tokens, i.e., $\mathcal{S}(k) = \{\underline{\mu} = (m_0, \underline{m}) \in \mathcal{S} | m_0 = k\}$. The levels $\mathcal{S}(k)$, $k \in I\!N$ constitute a partition of the overall state space: $\mathcal{S} = \bigcup_{k=0}^{\infty} \mathcal{S}(k)$ and $\mathcal{S}(k) \cap \mathcal{S}(l) = \emptyset, k \neq l$. For ease in notation, we also introduce $\mathcal{S}'(k) = \{\underline{m} | (k, \underline{m}) \in \mathcal{S}(k)\}$.

We furthermore define the following two *leads to* relations. We denote $\underline{\mu} \xrightarrow{t} \underline{\mu}'$ if transition t is enabled in $\underline{\mu}$ and, upon firing, leads to marking $\underline{\mu}'$. The firing rate of t is not important. We denote $\underline{\mu} \xrightarrow{t,\lambda} \underline{\mu}'$ if transition $t \in T$ is enabled in $\underline{\mu}$ and, upon firing, with rate λ, leads to marking $\underline{\mu}'$.

2.2 Requirements: formal definition

We now define a number of requirements on the SPN structure and transition firing behaviour. It should be noted that these requirements are sufficient, rather than necessary, for the SPNs to have an underlying QBD-structure.

Requirement 1. Given *SPN*, there exists a $\kappa \in I\!N$ such that for all $k, l \geq \kappa$: $\mathcal{S}'(k) = \mathcal{S}'(l)$. We denote $L = |\mathcal{S}'(\kappa)|$.

Requirement 2. Given *SPN* and κ as defined above, the following requirements should hold for the so-called *repeating portion* of the state space:

1. intra-level equivalence:
   $\forall k, l \geq \kappa$, $t \in T$, $\lambda \in I\!R^+$: if $(k, \underline{m}) \xrightarrow{t,\lambda} (k, \underline{m}')$ then $(l, \underline{m}) \xrightarrow{t,\lambda} (l, \underline{m}')$;
2. inter-level one-step increases only:
   $\forall k \geq \kappa$, $\exists t \in T$: $(k, \underline{m}) \xrightarrow{t} (k+1, \underline{m}')$;

$\forall k \geq \kappa$, $\exists t \in T$, $\lambda \in \mathbb{R}^+$: if $(k, \underline{m}) \xrightarrow{t,\lambda} (k+1, \underline{m}')$ then $(k+1, \underline{m}) \xrightarrow{t,\lambda} (k+2, \underline{m}')$;
$\forall k \geq \kappa$, $\forall i \in \mathbb{N}, i \geq 2$, $\nexists t \in T$: $(k, \underline{m}) \xrightarrow{t} (k+i, \underline{m}')$;

3. inter-level one-step decreases only:
$\forall k > \kappa$, $\exists t \in T$: $(k+1, \underline{m}) \xrightarrow{t} (k, \underline{m}')$;
$\forall k > \kappa$, $\exists t \in T$, $\lambda \in \mathbb{R}^+$: if $(k+2, \underline{m}) \xrightarrow{t,\lambda} (k+1, \underline{m}')$ then $(k+1, \underline{m}) \xrightarrow{t,\lambda} (k, \underline{m}')$;
$\forall k > \kappa$, $\forall i \in \mathbb{N}, i \geq 2$, $\nexists t \in T$: $(k+i, \underline{m}) \xrightarrow{t} (k, \underline{m}')$;

Requirement 3. Given *SPN* and κ as defined above, for the so-called *boundary portion* of the state space the following requirements should hold:

1. no boundary jumping:
$\forall k < \kappa - 1, \forall l > \kappa$, $\nexists t_1 \in T$: $(k, \underline{m}) \xrightarrow{t_1} (l, \underline{m}')$;
$\forall k < \kappa - 1, \forall l > \kappa$, $\nexists t_2 \in T$: $(l, \underline{m}) \xrightarrow{t_2} (k, \underline{m}')$;

2. only boundary crossing:
$\exists t_1, t_2 \in T$: $(\kappa - 1, \underline{m}_1) \xrightarrow{t_1} (\kappa, \underline{m}_1')$, $(\kappa, \underline{m}_2) \xrightarrow{t_2} (\kappa - 1, \underline{m}_2')$;

2.3 Requirements: discussion

Although the requirements to be posed on the SPN models are exact in themselves, it is good to discuss them in a more informal way.

Requirement 1 states that, starting from a certain level κ onwards, all levels are the same as far as the non-infinite places are concerned; they only differ in the number of tokens in place p_0. It is for this reason that the levels $k \geq \kappa$ are often called the repeating portion (levels) of the state space. The levels $k < \kappa$ are denoted the boundary portion (levels) of the state space. In Figure 1 we depict the overall state space and its partitioning in levels. We have tried to visualize the fact that starting from level κ onwards, the levels are similar to each other. Levels 0 through $\kappa - 1$ can be totally different from each other. In between states from levels 0 through $\kappa - 1$ all kinds of transitions may occur. That is why we can also see these boundary levels as one aggregated boundary level (Requirement 1).

Transitions can occur within a level, and between levels. Since the repeating levels are always the same, apart from the level number itself, all internal transitions in one level, must have similar equivalents in other repeating levels. There are no transitions possible between non-neighbouring levels. There have to exist up- and down-going transitions between neighbouring levels. Also, for the repeating levels, their interaction with neighbouring levels is always the same (Requirement 2).

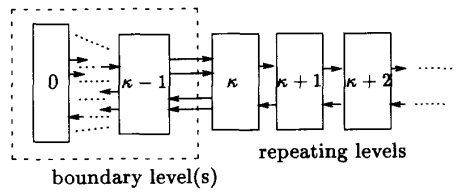

Figure 1: State space partitioning in levels

The transitions between the boundary levels and the repeating levels only take place in levels $\kappa - 1$ and κ, however, they may have any form (Requirement 3).

Regarding the levels as "super states" one easily sees the similarity with the well-known state spaces of birth-death queueing models. It is this similarity that has named a stochastic process on such a "levelized" state space a quasi birth-death model.

The stated requirements are sufficient, however, not always necessary. One might imagine SPNs which have a slightly different structure that still allow for a matrix geometric solution.

3 Matrix geometric solution

We start with a state space partitioning in Section 3.1. Then, in Section 3.2, we derive a matrix geometric solution for the steady-state probabilities, and in Section 3.3 we discuss the algorithmic aspects. The derivation of reward-based measures is discussed in Section 3.4. Finally, we discuss how results for infinite models can be used to derive (approximate) results for finite models in Section 3.5.

3.1 State space partitioning

Referring to Figure 1, it is easy to see that the generator matrix $\mathbf{Q}$ of the QBD has the following form:

$$\begin{pmatrix}
\mathbf{B}_{0,0} & \cdots & \mathbf{B}_{0,\kappa-1} & 0 & 0 & \cdots \\
\mathbf{B}_{1,0} & \cdots & \mathbf{B}_{1,\kappa-1} & 0 & 0 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\mathbf{B}_{\kappa-1,0} & \cdots & \mathbf{B}_{\kappa-1,\kappa-1} & \mathbf{B}_{\kappa-1,\kappa} & 0 & \cdots \\
0 & \cdots & \mathbf{B}_{\kappa,\kappa-1} & \mathbf{B}_{\kappa,\kappa} & \mathbf{B}_{\kappa,\kappa+1} & 0 \\
0 & \cdots & 0 & \mathbf{B}_{\kappa+1,\kappa} & \mathbf{B}_{\kappa+1,\kappa+1} & \mathbf{B}_{\kappa+1,\kappa+2} \\
 & 0 & \cdots & 0 & 0 & \mathbf{B}_{\kappa+2,\kappa+1} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{pmatrix}.$$

Now, by the requirements posed on the intra- and inter-level transitions, we have

$$\begin{cases} \mathbf{A}_0 = \mathbf{B}_{k,k+1}, & k = \kappa, \kappa+1 \cdots, \\ \mathbf{A}_1 = \mathbf{B}_{k,k}, & k = \kappa+1, \kappa+2 \cdots, \\ \mathbf{A}_2 = \mathbf{B}_{k,k-1}, & k = \kappa+1, \kappa+2 \cdots. \end{cases}$$

Using this notation, we may write $\mathbf{Q}$ as follows:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{B}_{0,0} & \cdots & \mathbf{B}_{0,\kappa-1} & 0 & \cdots & \cdots \\ \vdots & \cdots & \vdots & 0 & \cdots & \cdots \\ \mathbf{B}_{\kappa-1,0} & \cdots & \mathbf{B}_{\kappa-1,\kappa-1} & \mathbf{B}_{\kappa-1,\kappa} & 0 & \cdots \\ 0 & 0 & \mathbf{B}_{\kappa,\kappa-1} & \mathbf{B}_{\kappa,\kappa} & \mathbf{A}_0 & \cdots \\ 0 & 0 & 0 & \mathbf{A}_2 & \mathbf{A}_1 & \cdots \\ 0 & 0 & 0 & 0 & \mathbf{A}_2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

3.2 Steady-state probabilities

When the CTMC would have been finite, we would have calculated the steady-state probability row vector $\underline{\pi}$ by numerically solving the global balance equations (GBEs):

$$\underline{\pi}\mathbf{Q} = \underline{0}, \text{ and } \underline{\pi}\underline{1}^{\mathrm{T}} = 1, \tag{1}$$

where the right part is a normalisation, i.e., $\underline{1}^{\mathrm{T}} = (1, \cdots, 1)^{\mathrm{T}}$. In the infinite case, we start from the same equations. First, we partition $\underline{\pi}$ according the states belonging to the various levels, i.e., $\underline{\pi} = (\underline{z}_0, \underline{z}_1, \cdots, \underline{z}_{\kappa-1}, \underline{z}_\kappa, \underline{z}_{\kappa+1}, \cdots)$. Substituting this in (1), we obtain

$$\begin{array}{ll} \text{(a)} : i = 0, \cdots, \kappa-2 : & \sum_{j=0}^{\kappa-1} \underline{z}_j \mathbf{B}_{j,i} = \underline{0}, \\ \text{(b)} : i = \kappa-1 : & \sum_{j=0}^{\kappa} \underline{z}_j \mathbf{B}_{j,i} = \underline{0}, \\ \text{(c)} : i = \kappa : & \sum_{j=\kappa-1}^{\kappa+1} \underline{z}_j \mathbf{B}_{j,i} = \underline{0}, \\ \text{(d)} : i = \kappa+1, \cdots : & \sum_{j=0}^{2} \underline{z}_{i+j-1} \mathbf{A}_j = \underline{0}, \\ \text{(e)} : \text{normalisation} : & \sum_{i=0}^{\infty} \underline{z}_i \cdot \underline{1}^{\mathrm{T}} = 1. \end{array} \tag{2}$$

We now try to exploit the regular structure in the state space in the solution process. In particular, looking at (2.d), it seems that for the state probabilities $\underline{z}_i$, $i = \kappa, \kappa+1, \cdots$, only the neighbouring levels are of importance. We know this situation from the birth-death process for the M|M|1 queue. For that case, it is well-known that $\pi_{i+1} = \rho\pi_i$, with π_i the probability of having i customers in the system and $\rho = \lambda/\mu$ the traffic intensity. In a similar way, we can *assume* that

$$\underline{z}_{\kappa+1} = \underline{z}_\kappa \mathbf{R}, \ \underline{z}_{\kappa+2} = \underline{z}_{\kappa+1}\mathbf{R} = \underline{z}_\kappa \mathbf{R}^2, \cdots, \tag{3}$$

or, equivalently,

$$\underline{z}_{\kappa+i} = \underline{z}_\kappa \mathbf{R}^i, i \in I\!N, \tag{4}$$

where $\mathbf{R}$ is an $L \times L$ matrix relating the steady-state probability vector at level $\kappa+i$ to the steady-state probability vector at level $\kappa+i-1$ ($i = 1, 2, \cdots$). To validate the above assumption, we have to substitute it in Equation 2(d). For $i \in I\!N$, we find

$$\begin{aligned} &\underline{z}_{\kappa+i}\mathbf{A}_0 + \underline{z}_{\kappa+i+1}\mathbf{A}_1 + \underline{z}_{\kappa+i+2}\mathbf{A}_2 = \mathbf{0} \Leftrightarrow \\ &\underline{z}_\kappa \mathbf{R}^i \mathbf{A}_0 + \underline{z}_\kappa \mathbf{R}^{i+1}\mathbf{A}_1 + \underline{z}_\kappa \mathbf{R}^{i+2}\mathbf{A}_2) = \mathbf{0} \Leftrightarrow \\ &\underline{z}_\kappa \mathbf{R}^i(\mathbf{A}_0 + \mathbf{R}\mathbf{A}_1 + \mathbf{R}^2\mathbf{A}_2) = \mathbf{0}. \end{aligned} \tag{5}$$

For these three multiplicative terms to yield zero, at least one of them must equal zero. If $\underline{z}_\kappa$ would be zero, all higher level probabilities would be zero as well. This can not be the case. A similar reasoning is valid for $\mathbf{R}$ so that we have to conclude that assumption (3) is valid, if $\mathbf{R}$ satisfies the matrix polynomial

$$\mathbf{A}_0 + \mathbf{R}\mathbf{A}_1 + \mathbf{R}^2\mathbf{A}_2 = \mathbf{0}. \tag{6}$$

In case $i = \kappa$, (2.c) can be rewritten to incorporate assumption (3), because $\underline{z}_{\kappa+1}$ can be rewritten in terms of $\underline{z}_\kappa$ and $\mathbf{B}_{\kappa+1,\kappa} = \mathbf{A}_2$:

$$\sum_{j=\kappa-1}^{\kappa+1} \underline{z}_j \mathbf{B}_{j,\kappa} = \underline{z}_{\kappa-1}\mathbf{B}_{\kappa-1,\kappa} + \underline{z}_\kappa(\mathbf{B}_{\kappa,\kappa} + \mathbf{R}\mathbf{A}_2) = \underline{0}. \tag{7}$$

With this substitution, (2.a–c) comprises a system of $\kappa+1$ linear vector equations with as many unknown vectors. However, as these vectors are still dependent, the normalisation (2.e) has to be integrated in it, to yield a unique solution. This normalisation can be rewritten as follows:

$$\begin{aligned} \sum_{i=0}^{\infty} \underline{z}_i \underline{1}^{\mathrm{T}} &= \sum_{i=0}^{\kappa-1} \underline{z}_i \underline{1}^{\mathrm{T}} + \sum_{i=\kappa}^{\infty} (\underline{z}_\kappa \mathbf{R}^i)\underline{1}^{\mathrm{T}} \\ &= \sum_{i=0}^{\kappa-1} \underline{z}_i \underline{1}^{\mathrm{T}} + \underline{z}_\kappa \mathbf{R}^\kappa (\mathbf{I} - \mathbf{R})^{-1} \underline{1}^{\mathrm{T}} = 1. \end{aligned} \tag{8}$$

3.3 Algorithm

To summarize, the following steps should be undertaken in the solution process:

S0. From the *SPN*, derive κ, $\mathbf{A}_i$ ($i = 0, 1, 2$) and $\mathbf{B}_{ij}$ ($i, j = 0, \cdots, \kappa$).

S1. Compute $\mathbf{R}$ from the matrix polynomial (6);

S2. Solve the linear system (2.a–c), using (7) and (8);

S3. Compute $\underline{z}_{\kappa+i}$ as $\underline{z}_\kappa \mathbf{R}^i$.

Step **S0** typically requires software tool support, which is dealt with separately in Section 4.

Step **S1** is normally done via an iterative approach. Starting with $\mathbf{R}(0) = -\mathbf{A}_0\mathbf{A}_1^{-1}$, we obtain successive approximations of $\mathbf{R}$ as follows:

$$\mathbf{R}(k+1) = -(\mathbf{A}_0 + \mathbf{R}^2(k)\mathbf{A}_2)\mathbf{A}_1^{-1}. \tag{9}$$

The iteration is stopped whenever $||\mathbf{R}(k)-\mathbf{R}(k+1)|| < \epsilon$. It can be shown that the sequence $\{\mathbf{R}(k),\ k = 0, 1, \cdots\}$ is entry-wise nondecreasing, and that it converges monotonically to the matrix $\mathbf{R}$ [27]. Substantial speed-up can be gained when using the new algorithm of Latouche and Ramaswami, as illustrated in [33].

If series of performance analyses need to be performed (parametric analysis), speed-up can be gained if these analyses are ordered in increasing order of system utilisation. The **R**-matrix of a previous analysis can then be used as initial value for the next analysis. Although we have not (yet) automated this, experience has shown that the required number of steps in the iteration decreases, albeit not dramatically.

Step **S2** is normally done via Gaussian elimination or Gauss-Seidel iteration (or SOR), depending on the size of the system. Our experience is that the boundary part of the model is usually of such a size that Gaussian elimination is well feasible [24, 33].

Step **S3** is only performed when required (see also Section 3.4).

3.4 Reward-based measures

Once the steady-state probabilities are known, reward-based measures can be computed easily. Let $r : I\!N \times \mathcal{M} \rightarrow I\!R$ denote a real-valued reward function defined on the state space of the model. The steady-state expected reward rate is then computed as

$$E[r] = \sum_{i=0}^{\infty} \sum_{\underline{m} \in \mathcal{S}'(i)} z_{i,\underline{m}} r(i, \underline{m}).$$

If the rewards are only dependent on the level number and not on the inter-level state, i.e., if $r(i, \underline{m}) = r(i, \underline{m}')$, for all $\underline{m}, \underline{m}' \in \mathcal{S}'(i)$ given fixed $i \in I\!N$, then we write $r(i) = r(i, \underline{m})$ and consequently

$$E[r] = \sum_{i=0}^{\infty} r(i)(\underline{z}_i \underline{1}^{\mathrm{T}}).$$

If $r(i) = i$, further reductions in complexity can be reached, using results for the geometric series, thus avoiding explicit calculation of the state probabilities and the infinite summation.

3.5 Finite models

CTMCs with infinite state space are often used to obtain bounds or approximations for models with a large but finite state space. Two steps need to be undertaken to accomplish this:

Infinite model solution. The infinite-size model is solved, for instance with matrix geometric techniques, yielding $\underline{\pi} = (\underline{z}_0, \underline{z}_1, \cdots)$;

Renormalisation. A new probability vector $\underline{u} = (\underline{u}_0, \underline{u}_1, \cdots, \underline{u}_F)$ is defined, with $\underline{u}_i = \Phi_F \underline{z}_i$ $(i = 0, \cdots, F)$, and $\Phi_F = (\sum_{i=0}^{F} \underline{z}_i \underline{1}^{\mathrm{T}})^{-1}$. In doing so, $\underline{u}$ is a probability vector on the levels 0 through F.

Under many circumstances, $\underline{u}$ provides a good approximation for the exact steady-state proabilities of a finite model on levels 0 through F (see [34]). When certain quasi-reversibility properties hold, the renormalization is even exact [20, 23].

Finally, it should be noted that also for a class of *finite* QBD models, matrix geometric solutions apply directly [28, Chapter 4], [19]. These, of course, could also be applied when appropriate.

4 Tool support

A prerequisite to make any performance evaluation technique really usable is to have software tool support for it available. Such tools should (i) allow users to specify their models at a high-level of abstraction; (ii) automatically derive the, often more detailed, underlying mathematical model; (iii) solve the mathematical model in the most appropriate way; and (iv) present the solution results in terms of the original high-level model.

For the class of SPN model at hand, this implies that users should be able to specify, at SPN level, a model and that the tool finds out whether the requirements are met or not. If so, the matrix geometric solution technique can be employed. If the requirements are not met, various ways can be chosen. The tool can just indicate so, and stop the analysis, possibly also indicating where "it went wrong". A different approach might be to analyze the model using either a simulation approach, or a numerical approach with a truncated version of the model on a very large but still finite state space.

One practical problem that comes along here is that the requirements 1–3 may be easy to write down, they are not easily verified in general. As an example of this, consider a marking-dependent transition between two neighbouring levels in the repeating portion of the state space. If the transition has different rates between different neighbouring levels, Requirement 2.1 is not fulfilled. However, it very much depends on the way the marking dependent transition rate is specified, whether this can be checked in finite time or not.

When inhibitor arcs and enabling functions are allowed, in its most general setting, the verification problem might even be undecidable (see also [3]). Therefore, the given requirements should be interpreted as being sufficient to allow for the efficient solution. However, we are free to pose extra requirements, which ease the

task of verifying whether a certain SPN passes the test or not, albeit possibly at the cost of less modelling flexibility. As an example of this, not allowing for marking dependent transition rates, will ease the task of verification.

Another problem is the determination of κ. Although this is often easy for a human being, doing this for instance "by inspection" of the upper left part of the matrix $\mathbf{Q}$, for a computer program this is less easy to do. In particular, the state space generation order plays an important role here; it should be done level-by-level. One way to solve this, might be to have the tool user indicate the value of κ. An indicated value that is too large does not spoil the solution process, however, it becomes slightly inefficient. An indicated value of κ that is too small will not yield correct results.

The aim of this paper is to present a class of infintely large SPNs which allow for an efficient solution. In a later paper we will report about the tooling effort we undertake. For the time being, we have used our tool for matrix geometric methods, i.e., Xmgm [21], for the solution of the underlying infinite CTMCs, i.e., for steps **S1** through **S3**. Step **S0** has been performed manually.

5 Applications

In this section we discuss two applications. The first application, a single server queueing system with delayed service, is presented in Section 5.1. Then, in Section 5.2, we discuss an application from the area of B-ISDN.

5.1 Queueing with delayed service

Considers a single server queueing system (see Figure 2) at which customers arrive as a Poisson process with rate λ via transition `arr` and are served with rate μ via transition `serve`.

Before service, arriving customers are stored in a `buffer`. Service is not immediately granted to an arriving customer, even not is the server is idle at that time (a token in place `sleep`). Only after there are at least T (for threshold) customers queued, the server awakes and starts its duties. This is enforced by an enabling predicate associated with the immediate transition `wake-up`: `#buffer` $\geq T$. The server subsequently remains awake until the buffer becomes empty, after which it resumes sleeping.

For a threshold $T = 3$, the corresponding CTMC is given in Figure 3. From the models, it can easily be seen that they fulfill Requirements 1–3 with $\kappa = T = 3$.

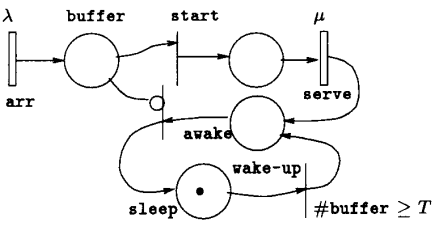

Figure 2: SPN of a single server queueing system with delayed service

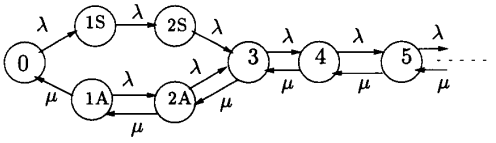

Figure 3: CTMC of the single server queueing system with delayed service; $T = 3$

The generator matrix then has the following form:

$$\mathbf{Q} = \left(\begin{array}{c|c|c|c|c|c|c|c} -\Sigma & \lambda & 0 & 0 & 0 & & & \cdots \\ 0 & -\Sigma & 0 & \lambda & 0 & & & \cdots \\ \mu & 0 & -\Sigma & 0 & \lambda & & & \cdots \\ \hline 0 & 0 & 0 & -\Sigma & 0 & \lambda & & \cdots \\ 0 & 0 & \mu & 0 & -\Sigma & \lambda & & \cdots \\ \hline 0 & 0 & 0 & 0 & \mu & -\Sigma & \lambda & \cdots \\ 0 & 0 & 0 & 0 & 0 & \mu & -\Sigma & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \mu & \cdots \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array}\right),$$

where Σ is chosen such that the row sums equal 0. From this matrix we observe that the $L \times L$ $\mathbf{A}$-matrices have the following form ($L = 1$): $\mathbf{A}_0 = (\lambda)$, $\mathbf{A}_1 = (-(\lambda + \mu))$ and $\mathbf{A}_2 = (\mu)$. From these matrices we derive $\mu\mathbf{R}^2 - (\lambda + \mu)\mathbf{R} + \lambda = 0$ which has as only valid solution $\mathbf{R} = (\lambda/\mu)$. From this, it once again becomes clear that $\mathbf{R}$ takes over the role of ρ in simpler queueing analysis, such as in the M|M|1 queue.

Denoting $\underline{z}_i = z_i$ ($i = 0$ or $i = 3, 4, \cdots$) and $\underline{z}_i = (z_{i,S}, z_{i,A})$ ($i = 2, 3$), the boundary equations become (from these seven equations, any one of the first six can be omitted):

$$\begin{cases} -\lambda z_0 + \mu z_{1A} = 0, \\ \lambda z_0 - \lambda z_{1S} = 0, \\ -(\lambda + \mu) z_{1A} + \mu z_{2A} = 0, \\ \lambda z_{1S} - \lambda z_{2S} = 0, \\ \lambda z_{1A} - (\lambda + \mu) z_{2A} + \mu z_3 = 0, \\ \lambda z_{2S} + \lambda z_{2A} - (\lambda + \mu) z_3 + \mu z_4 = 0, \\ z_0 + z_{1S} + z_{1A} + z_{2S} + z_{2A} + z_3(1 - \rho)^{-1} = 1. \end{cases}$$

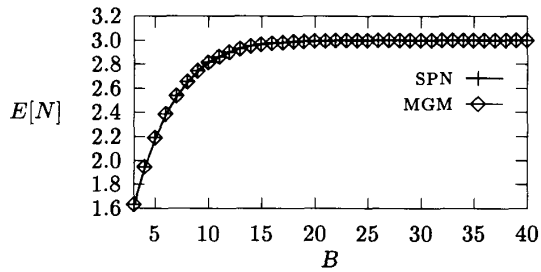

Figure 4: The average number of customers in the system, $E[N]$, as a function of the number of buffer places

Concluding, to solve this SPN model using matrix gemetric techniques, for a threshold T, we require a linear system of size $2T$ to be solved, as well as one quadratic equation which, since $L = 1$, can be solved in fixed time.

In contrast, when using a numerical approach based on the global balance equations of a finite SPN model, i.e., a model with a limit of B buffer places, we require a system of linear equations of size $T + B$ to be solved. Especially when B becomes large, the infinite-buffer model with matrix geometric solution is more efficient. Also when the renormalization procedure as outlined in Section 3.5 is used (see below), the matrix gemetric approach becomes more efficient, as for increasing buffer size B, only a renormalization has to be performed, whereas the solution based on the global balance equations has to be totally redone.

As a numerical example, consider the case where $\lambda = 2$, $\mu = 3$, $T = 3$ and, consequently, $\rho = 2/3$. The matrix geometric solution with infinite buffer, i.e., $B = \infty$, results in the following boundary steady-state probabilities:

$$\begin{cases} z_0 = 0.11111, & z_{1A} = 0.07407, & z_{1S} = 0.11111, \\ z_{2A} = 0.12346, & z_{2S} = 0.11111, & z_3 = 0.15638. \end{cases}$$

Using these probabilities, and $z_i = z_3\rho^{i-3}$, $i = 4, 5, \cdots$, we obtain for the average number of customers in the system $E[N] = 3.00$.

In case we have a finite-buffer model, we can also compute $E[N]$ directly from the global balance equations. In Figure 4 we show the average number of customers in such a system as a function of the number of buffer places, as calculated from an SPN model, using the GBEs explicitly (labelled 'SPN'). We also show the renormalized results derived from the infinite-buffer model, according to the procedure proposed in Section 3.5 (labelled 'MGM'). Clearly, the results match perfectly. This is due to the quasi-reversibility property that holds in this case [20, 23]. A similar conclusion can be drawn for the computation of the buffer-overflow probabilities.

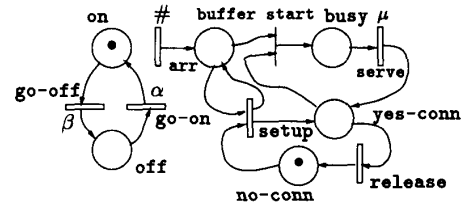

Figure 5: SPN model of OCDR mechanism

5.2 Connectionless traffic in B-ISDN

An ATM/B-ISDN-based communication infrastructure offers a connection-oriented service. Via two of the ATM adaptation layers 3/4 and 5, also connectionless services can be provided [30]. Packets arriving at the AAL service boundary to make use of such a service, suffer a possible delay from the connection establishment at the ATM service boundary, i.e., a connection set-up delay, unless there already exists a connection when the packet arrives. Once the connection has been established, all buffered packets can be transmitted and the connection can be released. This can be done immediately, or with some delay. The former has as disadvantage that a connection is being maintained when it is not needed, however, it has as advantage that some packets might profit from the fact that there is still a connection when they arrive. Whenever packets arrive in bursts, there is a trade-off between the release delay, the costs and of maintaining an unused connection and the perceived performance (average delay). The above way of implementing connectionless services, has been proposed by Heijenk *et al.* [22] under the name 'on-demand connection with delayed release' (OCDR).

An SPN model for such a system is given in Figure 5. Packets arrive via transition `arr` and are placed in the `buffer`. The rate of transition `arr` is modulated by an independent on/off-model. A token in place `on` or `off` respectively models the fact that the source is in a burst or not. When in a burst, packets are generated according to a Poisson process with rate λ packets/second. When not in a burst, no packets are generated. The transitions `go-on` and `go-off`, with rates α and β respectively, model the time durations the source remains in the off and on state. The service rate is μ Mbps and the average packet length is denoted l.

If the server is busy, there will be a token in place `busy` and arriving packets have to wait on their turn.

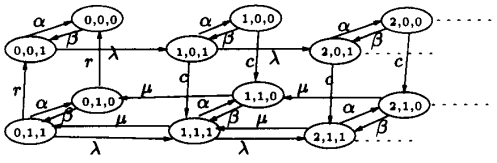

Figure 6: CTMC of the OCDR mechanism

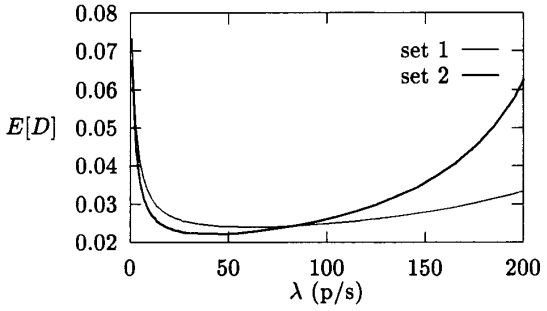

Figure 7: The expected delay $E[D]$ (in seconds) as a function of the arrival rate λ in a burst

If the server is idle, but there is no connection available, signified by a token in place `no-conn`, a connection will be established, causing a negative exponential delay with average length $1/c$ (transition `set-up`). Once there is a connection, normal packet transmissions can take place. Once the buffer is empty, the connection is released with a negative exponential delay with average length $1/r$ (transition `release`).

The corrsponding CTMC is given in Figure 6. In this model, the states space $\mathcal{S} = \{(i,j,k)|i \in I\!N, j,k = 0,1\}$. Parameter i denotes the number of packets in the system, j denotes whether there is a connection ($j = 1$) or not ($j = 0$) and k denotes whether the arrival process is in a burst ($k = 1$) or not ($k = 0$). As can be seen from the CTMC, but as can also be verified using Requirements 1–3, this model has a structure that allows for a matrix geometric solution. Every level $\mathcal{S}(l)$ consists of the $L = 4$ states with l packets present, i.e., $\mathcal{S}(l) = \{(l,0,0),(l,0,1),(l,1,0),(l,1,1)\}$.

Clearly, $\kappa = 1$ and the boundary equations are given by the global balance equations for the first two levels, i.e., level 0 and 1, plus the normalisation equation, in total yielding a system of 8 linear equations. The $L \times L$ matrix $\mathbf{R}$ with $L = 4$ now has to be solved numerically. If a finite buffer model were used, e.g., with B buffer

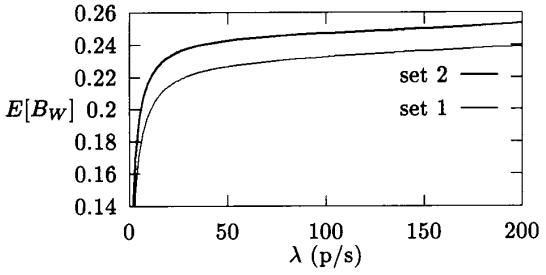

Figure 8: The expected bandwidth $E[B_W]$ (in Mbps) as a function of the arrival rate λ in a burst

places, an underlying finite CTMC can be generated with $4(B+1)$ states. As measures of interest we could address: (i) the average node delay $E[D]$ (in seconds); (ii) the average reserved bandwidth $E[B_W]$ (in Mbps); and (ii) the expected number of connection establishments per second $E[C]$ (in per-second). All these quantities can be expressed in closed-form using $\mathbf{R}$ and the boundary vector $\underline{z}_0$ (for details, see [22]).

Under the assumption that communication capacity can be claimed in various amounts, the service rate μ can be chosen freely. Given a certain workload, a higher requested transmission speed μ will yield smaller connection times, however, at higher costs per time unit. The parameter μ, therefore, together with the connection release rate r are interesting quantities to control the system performance and cost.

Let us now turn to some numerical results. We assume that $\alpha = 1.0$, $\beta = 0.04$, $c = 10.0$, and $l = 10$ kbit. We address the following two combinations of transmission and release rates: $(\mu,r)_1 = (336.0, 1.0)$ and $(\mu,r)_2 = (236.0, 0.5)$. In the first case, the transmission speed is relatively high, but connections are rapidly released after usage. In the second case, a lower transmission speed is used, but a connection is maintained longer. Therefore, arriving packets have a smaller probability to perceive an extra connection setup delay.

In Figure 7 we depict the expected delay $E[D]$ (in seconds) as a function of the arrival rate λ in a burst. Although for $\lambda \approx 85$ the average delay values coincide ($E[D] \approx 24.5$), for changing λ's this is certainly not the case. For the first parameter set, the average delay is less sensitive to changes in λ, especially towards higher values. For smaller values of λ, the average delay is smaller for parameter set 2. Suprisingly, the less sensitive solution, requires a smaller average bandwidth as well, as illustrated in Figure 8. The number of connection establishments, however, is higher, as illustrated

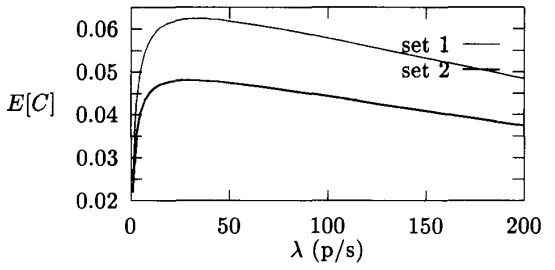

Figure 9: The expected conection setup rate $E[C]$ (in s^{-1}) as a function of the arrival rate λ in a burst

by Figure 9. Since the latter can be associated with costs in a B-ISDN context, the price to be paid for the less sensitive delay behaviour and for the smaller bandwidth consumption is paid here. Also observe, that for higher traffic, the number of connection establishments decreases, i.e., a connection that is once established, is used for a long time since the probability of having a connection and no packets present decreases with larger λ.

Finally, we report on the efficiency of the matrix geometric solution technique. The number of required steps in iteration (9) to calculate $\mathbf{R}$ with accuracy $\epsilon = 10^{-6}$ increases almost linearly with the traffic intensity λ, from a few iterations when λ is very small, to 250 when λ approaches 200. The required computation time to determine $\mathbf{R}$ then ranges from 0.05 through 1.00 seconds. These values are obtained without use of sparse matrices and without the earlier mentioned speed-up that could be obtained when doing multiple analyses.

6 Summary and conclusions

In this paper we have defined a class of infinite SPNs for which an efficient solution technique based on matrix geometric methods is feasible. The requirements to be posed on any SPN to exhibit such a solution are explicitly stated as sufficent conditions. The solution of the steady-state probabilities as well as of reward-based measures, thereby exploiting the QBD structure, is also discussed.

With two examples we have illustrated our approach. From these examples, and from the more general applicability of the theory, we think the newly defined class of SPNs is of importance for many application areas. In order to become practically useful, tool support is necessary. Some requirements on and suggestions for tools have also been made.

As future research areas we envisage the design and implementation of a tool supporting the presented model class and solution technique. Especially the issue of decidability and the refinement of the requirements will be an important research theme. Also, the refinements should preferably become verifyable at the SPN-level, rather than at the level of the underlying CTMC. Finally, attention should be paid to ergodicity conditions and their *a priori* validation.

Acknowledgements

The author would like to thank the reviewers for their constructive comments on the paper. G.J. Heijenk and W. van Dieten are thanked for the cooperation on the OCDR-example.

References

[1] M. Ajmone Marsan, G. Conte, G. Balbo, "A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems", *ACM TOCS* **2**(2), pp.93–122, 1984.

[2] M. Ajmone Marsan, G. Chiola, "On Petri Nets with Deterministic and Exponentailly Distributed Firing Times", *LNCS* **266**, Editor: G. Rozenberg, pp.132–145, 1987.

[3] S.S. Berson, R. Munts, "Detecting Block GI|M|1 and Block M|G|1 Matrices from Model Specifications", Technical Report, UCLA, 1994.

[4] P. Buchholz, "Hierarchical Markovian Models: Symmetries and Reduction", in: *Computer Performance Evaluation '92: Modelling Techniques and Tools*, Editors: R. Pooley, J. Hillston, pp.305–319, 1992.

[5] P. Buchholz, "Aggregation and Reduction techniques for Hierarchical GCSPNs", *Proc. of the 5th Int'l Workshop on Petri Nets and Performance Models*, pp.216–225, 1993.

[6] G. Chiola, C. Dutheillet, G. Franceschines, S. Haddad, "Stochastic Well-Formed Colored nets and Symmetric Modelling Applications", *IEEE TSE* **42**(11), pp.1343–1360, 1993.

[7] H. Choi, V.G. Kulkarni, K.S. Trivedi, "Markov-Regenerative Stochastic Petri Nets", in: *Performance '93*, Editors: G. Iazeolla, S.S. Lavenberg, North-Holland, 1993.

[8] G. Ciardo, J. Muppala, K.S. Trivedi, "SPNP: Stochastic Perti Net Package", *Proc. of the 3rd Int'l Workshop on Petri Nets and Performance Models*, pp.142–151, 1989.

[9] G. Ciardo, J. K. Muppala, and K. S. Trivedi, "On the Solution of GSPN Reward Models," *Performance Evaluation* **12**(4), pp.237-254, 1991.

[10] G. Ciardo, K.S. Trivedi, "A Decomposition Approach for Stochastic Reward Net Models", *Performance Evaluation* **18**(1), pp.37–59, 1993.

[11] G. Ciardo, R. German, C. Lindemann, "A Characterization of the Stochastic Process Underlying a Stochastic Petri Net", *IEEE TSE* **20**(7), pp.506–515, 1994.

[12] A.J. Coyle, W. Henderson, C.E.M. Pearce, P.G. Taylor, "Mean-Value Analysis for Batch-Movement Queueing Networks with Product-Form Equilibrium Distributions", *Proc. of the Australia-Japan Workshop on Stochastic Models in Engineering, Technology & Management*, Gold Coast, Australia, July 14–16, 1993, pp.96–106.

[13] W.J. van Dieten, *Performance Evaluation of a Connectionless Protocol over ATM using Matrix Geometric Methods*, University of Twente, Department of Computer Science, 1994.

[14] S. Donatelli, M. Sereno, "On The Product-Form Solution for Stochastic Petri Nets", *Application and Theory of Petri Nets 1992*, Editor: K. Jensen, pp.154–172, Springer Verlag, 1992.

[15] G. Florin, S. Natkin, "On Open Synchronized Queueing Networks", *Proc. of the International Workshop on Timed Petri Nets*, pp.226–223, 1985.

[16] G. Florin, S. Natkin, "One Place Unbounded Stochastic Petri Nets: Ergodicity Criteria and Steady-State Solution", *Journal of Systems and Software* **1**(2), pp.103–115, 1986.

[17] G. Florin, S. Natkin, "A Necessary and Sufficient Saturation Condition for Open Synchronized Queueing Networks", *Proc. of the 2nd Int'l Workshop on Petri Nets and Performance Models*, pp.4–13, 1987.

[18] G. Florin, S. Natkin, "Generalizations of Queueing Network Product-Form Solutions to Stochastic Petri Nets", *IEEE TSE* **17**(2), pp.99–107, 1991.

[19] L. Gün, A.M. Makowski, "Matrix Geometric Solutions for Finite Capacity Queues with Phase-Type Distributions", in: *Performance '87*, Editors: P.J. Courtois, G. Latouche, North-Holland, pp. 269–282, 1988.

[20] B.R. Haverkort, "Approximate Performability and Dependability Modelling using Generalized Stochastic Petri Nets", *Performance Evaluation* **18**(1), pp.61–78, 1993.

[21] B.R. Haverkort, A.P.A. van Moorsel, D.-J. Speelman, "Xmgm: A Performance Analysis Tool Based on Matrix Geometric Methods", in: *Proc. of the 2nd Int'l Workshop on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, pp.152–157, 1994.

[22] G.J. Heijenk, *Connectionless Communivcations using the Asynchronous Transfer Mode*, Ph.D. thesis, University of Twente, 1995.

[23] F.P. Kelly, *Reversibility and Stochastic Networks*, John Wiley & Sons, 1979.

[24] U. Krieger, B. Müller-Clostermann, M. Sczittnick, "Modelling and Analysis of Communication Systems Based on Computational Methods for Markov Chains", *IEEE JSAC* **8**(9), pp.1630-1648, 1990.

[25] C. Lindemann, "An Improved Numerical Algorithm for Calculating Steady-State Solutions of Deterministic and Stochastic Petri Nets", *Performance Evaluation* **18**(1), pp.79–95, 1993.

[26] M.K. Molloy, "Performance Analysis using Stochastic Petri Nets", *IEEE TC* **31**(9), pp.913–917, 1982.

[27] R. Nelson, "Matrix Geometric Solutions in Markov Models: A Mathematical Tutorial", *IBM Research Report* RC 16777, 1991.

[28] M.F. Neuts, *Matrix Geometric Solutions in Stochastic Models: An Algorithmic Approach*, Johns Hopkins University Press, Baltimore, 1981.

[29] M.F. Neuts, *Structured Stochastic Matrices of* M|G|1 *Type and Their Applications*, Marcel Dekker Inc., New York, 1989.

[30] R.O. Onvural, *Asynchronous Transfer Mode Networks: Performance Issues*, Artech House, 1994.

[31] W.H. Sanders, J.F. Meyer, "Reduced Base Model Construction for Stochastic Activity Networks", *IEEE JSAC* **9**(1), pp.25–36, 1991.

[32] M.F. Squillante, "MAGIC: A Computer Performance Modelling Tool Based on Matrix-Geometric Techniques", in: *Computer Performance Evaluation: Modelling Techniques and Tools*, Editors: G. Balbo, G. Serazzi, North-Holland, pp.411–425, 1992.

[33] W.J. Stewart, "On the Use of Numerical Methods for ATM Models", in: *Modelling and Performance Evaluation of ATM Technology*, IFIP Transactions C-15, Editors: H. Perros, G. Pujolle, Y. Takahashi, North-Holland, pp.375-396, 1993.

[34] H.C. Tijms, "Heuristics for Finite-Buffer Queues", Research Report 1991-29, Free University, Department of Econometrics, Amsterdam, 1991.