An Improved Numerical Algorithm for Calculating Steady-State Solutions of Deterministic and Stochastic Petri Net Models*

Christoph Lindemann

Technische Universität Berlin
Institut für Technische Informatik
Fachgebiet Prozeßdatenverarbeitung und Robotik
(Real-Time Systems and Robotics)
Franklinstr. 28/29
W-1000 Berlin 10, F. R. Germany

Abstract

This paper introduces an algorithm for calculating steady-state solutions of DSPN models. The described method employs the randomization technique enhanced by a stable calculation of Poisson probabilities. A complete re-design and re-implementation of the appropriate components implemented in the version 1.4 of the software package GreatSPN has lead to significant savings in both computation time and memory space. These benefits are illustrated by DSPN models taken from the literature. We consider DSPN models for an $E_r/D/1/K$ queueing system and a fault-tolerant clocking system. These examples show that the model solutions are calculated with significantly less computational effort and a better error control by the algorithm described than by the method implemented in the version 1.4 of the software package GreatSPN.

1 Introduction

Several classes of timed transition Petri nets have been proposed in order to define a unified modeling tool for formal description and performance analysis of computer and communication systems. Such Petri net models have been broadly accepted due to the availability of appropriate software packages (e.g. GreatSPN [6], [7], METASAN [15], SPNP [8]) which completely automate their solution process. Marsan, Balbo, and Conte defined Generalized Stochastic Petri Nets which include timed transitions associated with exponentially distributed firing delays and immediate transitions firing without a delay [1]. Marsan and Chiola introduced Deterministic and Stochastic Petri Nets (DSPNs, [2]) as an extension to Generalized Stochastic Petri Nets. DSPN models allow the association of a timed transition either with a deterministic or an exponentially distributed firing delay. Hence, DSPN models are suited to represent system features such as time-outs, propagation delays or processor rebooting times which are naturally associated with constant delays. DSPNs have been applied for modeling an Ethernet bus LAN [3] and a fault-tolerant clocking system [12].

Marsan and Chiola showed that by sampling the stochastic behavior of a DSPN only at appropriately selected instants of time, an embedded Markov chain can be derived which underlies the DSPN [2]. The numerical solution technique for computing steady-state marking probabilities of DSPN models is based on the restriction that there are no concurrently enabled transitions with deterministic delays. If all deterministic transitions of a DSPN are exclusively enabled, the calculation of the state transition matrix of the embedded Markov chain is straightforward. The solution of DSPN models which include deterministic transitions competitively or concurrently enabled with exponential transitions requires the calculation of time-dependent state probabilities. The problem with DSPN models lies in the substantial computational effort required by the numerical solution algorithm of the software tool GreatSPN. This fact has restricted the applicability of DSPN models to systems of small size [3]. A decomposition technique for DSPNs has been proposed for improving the efficiency of the analysis of DSPN models in [4]. In this approach the reduction of the computational effort of the solution is based on the structural analysis of DSPN models. As described in a recent paper [7] in the version 1.5 of the software package GreatSPN the numerical solution module for DSPNs has been removed and a software module for interactive timed simulation has been introduced.

This paper presents a numerical algorithm for a stable and efficient computation of steady-state solutions of

* This work was supported by the Federal Ministry for Research and Technology of Germany (BMFT) and by the German Research Council (DFG) under grants ITR9003 and Ho 1257/2-1, respectively.

DSPN models. The proposed computational method employs the randomization technique [11] for calculating the time-dependent quantities rather than a time discretization and direct matrix exponentiation by truncated series expansion as implemented in the version 1.4 of the software package GreatSPN [6]. To handle stiffness in a DSPN model a stable numerical method for calculating Poisson probabilities is employed [9]. This allows to employ DSPNs for dependability and performability modeling where typically parameters with different orders of magnitude cause stiffness.

The randomization approach has already been employed in the packages SPNP and METASAN for calculating transient state probabilities of the Markov chain underlying a GSPN or a Stochastic Activity Network model. In this paper a computational formula is derived for the average sojourn time during the enabling interval of a deterministic transition. This formula allows to employ the randomization technique for computing steady-state solutions of DSPN models. A complete re-design of the appropriate software components implemented in the version 1.4 of the software package GreatSPN has lead to significant savings in both computation time and memory space. The benefit of the described solution algorithm is illustrated by empirical examples taken from the literature. DSPN models for the $E_r/D/1/K$ queueing system and a fault-tolerant clocking system [12] are considered. For these models we present curves relating the computational effort of the solution as a function of the model size. These results are compared with the corresponding curves obtained by the numerical algorithm of the software package GreatSPN [6]. Since these DSPN models have closed-form solutions, the accuracy achieved by the proposed algorithm and by the numerical method of GreatSPN can be determined. The examples show that steady-state solutions of DSPN models are calculated with significantly less computational effort and a better error control by the approach proposed than by the method implemented in the version 1.4 of the software package GreatSPN. Moreover, the presented algorithm can be employed in the decomposition technique for DSPN models proposed in [4] to calculate transient solutions of DSPN subnets consisting of exponential and immediate transitions.

The remainder of this paper is organized as follows. Section 2 recalls the main concepts of the randomization technique. In section 3 we show how to employ the randomization technique for calculating the state transition probabilities of the embedded Markov chain. Moreover, a formula is derived for efficiently computing the conversion factors required to obtain the steady-state solution of DSPN models. In section 4 two DSPN models are considered as empirical examples to illustrate the benefit of the described solution algorithm. Finally, concluding remarks are given.

2 Description of the Randomization Technique

Several numerical techniques for computing time-dependent state probabilities of a continuous-time Markov chain with finite state space and sparse generator matrix have been evaluated by Reibman and Trivedi [14]. In particular, the randomization technique [11] and two linear multi-step methods for numerically solving the Chapman-Kolmogorov differential equation have been considered. All these methods have implementations which exploit the sparsity of the generator matrix Q. However, only the randomization technique additionally exploits the probabilistic structure of the matrix Q. As a consequence, it has been shown that the randomization approach is more efficient than general solution techniques for differential equations. Employing the randomization approach a scalar q and a matrix A are defined as follows:

$$q = 1.02 \cdot \max_{1 \le k \le N} |q_{kk}|$$

$$A = \frac{1}{q} Q + I \tag{1}$$

The definition of the scalar q is due to Wallace and Rosenberg [17] and ensures that the subordinated discrete-time Markov chain with generator A is aperiodic. Since negative entries of the matrix Q are restricted to its diagonal, all entries of the matrix A are non-negative. Rewriting equation (1) the matrix Q and the matrix exponential e^{Qt} [10] can be expressed as:

$$Q = Aq - Iq \tag{2}$$

$$e^{Qt} = e^{Aqt} \cdot e^{-qt} \tag{3}$$

Since the state probability distribution at time t, denoted by $\pi(t)$, is given by the vector-matrix product of the initial state probability distribution $\pi_{initial}$ with the matrix exponential e^{Qt}, the transient probability vector $\pi(t)$ can be calculated by

$$\begin{aligned} \pi(t) &= \pi_{initial} \cdot e^{Qt} \\ &= \sum_{k=L(qt,\varepsilon)}^{R(qt,\varepsilon)} \pi_{initial} \cdot A^k \left(e^{-qt} \cdot \frac{(qt)^k}{k!}\right) \\ &= \sum_{k=L(qt,\varepsilon)}^{R(qt,\varepsilon)} \Phi(k) \cdot \beta(k) \end{aligned} \tag{4}$$

In formula (4) $\Phi(k)$ denotes the state probability vector of the subordinated discrete time Markov chain at the

instant of time k and $\beta(k)$ denotes the k-th Poisson probability. The left and right truncation points $L = L(qt, \varepsilon)$ and $R = R(qt, \varepsilon)$ of the summation in formula (4) are depending on the pre-defined error tolerance ε for each element of $\pi(t)$ and can be determined by:

$$\sum_{k=L}^{R} \beta(k) \geq 1 - \varepsilon \tag{5}$$

Since formula (5) defines a bound on the total mass of the truncated series, it provides a conservative estimate of the truncation error of formula (4) for reasonable values of ε. The time-dependent quantities of the solution of the DSPN models considered in section 4 can be calculated on a Sun 4/65 Sparc™ station with the described algorithm with an accuracy of 10^{-14}. The existence of an absolute error bound for each element of the probability vector $\pi(t)$ forms a major advantage of this computational method.

According to formula (4) the computation of the transient probability vector $\pi(t)$ of the continuous-time Markov chain is reduced to the computation of the transient probability vector $\Phi(k)$ of the subordinated discrete-time Markov chain. The function $\Phi(k)$ forms the (forward) Chapman-Kolmogorov equation for the discrete-time Markov chain assuming the system is initially in state $\pi_{initial}$. It can be efficiently computed by recursive vector-matrix multiplications [11].

$$\Phi(0) = \pi_{initial}$$
$$\Phi(k+1) = \Phi(k) \cdot A \tag{6}$$

In case the product $qt < 25$ the Poisson probabilities $\beta(k)$ are calculated recursively as:

$$\beta(0) = e^{-qt}$$
$$\beta(k+1) = \beta(k) \cdot \frac{qt}{k+1} \tag{7}$$

If the product $qt > 25$ a special-purpose algorithm for calculating Poisson probabilities proposed by Fox and Glynn [9] is employed. The summation of formula (7) starts at an appropriately determined left truncation truncation point $L > 0$ and the left tail of the Poisson distribution is replaced by a normal approximation [14]. This computational method considers the smallest and largest representable machine number and ensures that numerical overflow and underflow do not occur.

Time-dependent state probabilities of a Markov chain can also be determined by direct series expansion of the matrix exponential e^{Qt} [10].

$$\pi(t) = \pi_{initial} \cdot e^{Qt} = \sum_{k=0}^{R(t,\varepsilon)} \pi_{initial} \; Q^k \cdot \frac{t^k}{k!} \tag{8}$$

This computation of the truncated series expansion may lead to severe cancellation errors because of the negative diagonal elements of the generator matrix Q. To overcome this problem the DSPN solution algorithm provided by the version 1.4 of the software package GreatSPN [6] employs a refined computational technique of equation (8). The time interval [0, t] is divided into several subintervals depending on the rates of exponential transitions. To keep the computational effort within a reasonable range the length of the subintervals are bounded by constants incorporated in the source code of GreatSPN. As a consequence, the computational cost of this multi-step method depends on the minimum allowable step length. Moreover, this *adaptive matrix exponentiation* algorithm can take into account the numerical nature of the model parameters only in a limited way.

Since all terms of the matrix A are non-negative and less than or equal to 1, the computation of the series expansion of e^{Aqt} is not as badly affected by round-off errors as the series expansion of e^{Qt}. Given an error tolerance ε the computational complexity of the randomization method is given by $O(\eta qt)$ [14] where η denotes the number of nonzero entries in the generator matrix Q. In the next section we show how to employ this technique for calculating the time-dependent quantities of the solution process of DSPN models.

3 Efficient Calculation of Steady-state Solutions for DSPN Models

Marsan and Chiola introduced Deterministic and Stochastic Petri Nets (DSPNs, [2]) as an extension to Generalized Stochastic Petri Nets. DSPN models allow the association of a timed transition either with a deterministic or an exponentially distributed firing delay. It has been shown that a DSPN can be mapped on a semi-Markov process because a discrete-time Markov chain can be defined by sampling the stochastic behavior of the DSPN only at appropriately selected instants of time [2]. If a deterministic transition is enabled, the continuous-time stochastic behavior of the DSPN is sampled at the instant of its firing. Otherwise the stochastic behavior of the DSPN is sampled at the instant of firing of an exponential transition.

In case a deterministic transition is competitively or concurrently enabled with some exponential transitions time-dependent state probabilities have to be calculated in order to determine the state transition probabilities $P(S_i \rightarrow S_j)$ of the embedded Markov chain. These state transition probabilities are given by [2]:

$$P(S_i \rightarrow S_j) = u_i \cdot e^{Q\tau_i} \cdot \Delta_i \cdot u_j^T \quad (9)$$

In formula (9) Δ_i denotes the state transition matrix representing the feasible changes from marking M_i to other tangible markings of the DSPN following a path of immediate transition firings which are enabled after firing the deterministic transition. Nonzero entries of Δ_i are either "1" or given by the weights associated with conflicting immediate transitions. Q denotes the generator matrix of the Markov chain defined by the exponential transitions competitively and concurrently enabled with a particular deterministic transition. The state transition probabilities $P(S_i \rightarrow S_j)$ from a state S_i enabling a deterministic transition with delay τ_i are derived by the transient state probability vector $\pi(\tau_i)$ with initial state vector u_i. Practically, the entire i-th row of the state transition matrix P(i) is determined in one step. The recursion (6) starts with $\Phi(0) = u_i$ Thus, with equation (7) the row vector P(i) is calculated by:

$$P(i) = \pi(\tau_i) \cdot \Delta_i = \left(\sum_{k=L}^{R} \Phi(k) \cdot \beta(k) \right) \cdot \Delta_i \quad (10)$$

Moreover, in case the deterministic transition is concurrently enabled with some exponential transitions conversion factors have to be determined in order to convert the steady-state probability of the discrete-time Markov chain to the steady-state probability of the continuous-time DSPN. These conversion factors can be interpreted as the average sojourn time in the state S_i during the enabling interval τ_i of the deterministic transition. They are given by [2]:

$$C(i,j) = \frac{1}{\tau_i} \int_0^{\tau_i} u_i \cdot e^{Qt} \cdot u_j^T dt \quad (11)$$

The following shows how the conversion factors C(i,j) of formula (11) can also be calculated using the randomization technique. Practically, an entire row C(i) is computed. Recalling formula (11) and substituting the matrix exponential according to (3) yields:

$$\begin{aligned} C(i) &= \frac{1}{\tau_i} \int_0^{\tau_i} u_i \cdot e^{Qt} dt \\ &= \frac{1}{\tau_i} \int_0^{\tau_i} u_i \cdot \left(e^{Aqt} \cdot e^{-qt} \right) dt \end{aligned} \quad (12)$$

Replacing the matrix exponential by its truncated series expansion leads to

$$C(i) = \frac{1}{\tau_i} \int_0^{\tau_i} u_i \cdot \left(\sum_{k=0}^{R} A^k \cdot \frac{(qt)^k}{k!} \right) \cdot e^{-qt} dt \quad (13)$$

After exchanging the integral and the summation and appropriately reordering the terms we have

$$C(i) = \frac{1}{\tau_i} \sum_{k=0}^{R} u_i \cdot A^k \left(q^k \int_0^{\tau_i} \frac{t^k}{k!} e^{-qt} dt \right) \quad (14)$$

It can be shown by integration by parts that

$$\int_0^{\tau_i} \frac{t^k}{k!} e^{-qt} dt = \frac{1}{q^{k+1}} - e^{-q\tau_i} \sum_{n=0}^{k} \frac{\tau_i^n q^{n-k-1}}{n!} \quad (15)$$

Employing (15) in formula (14) leads to

$$C(i) = \frac{1}{q\tau_i} \sum_{k=0}^{R} u_i \cdot A^k \left(1 - e^{-q\tau_i} \sum_{n=0}^{k} \frac{(q\tau_i)^n}{n!} \right) \quad (16)$$

Replacing the vector-matrix product by $\Phi(k)$ according to (4) and substituting the Poisson probabilities by its left-truncated recurrence relation (5) the i-th row of conversion matrix C(i) can be efficiently calculated by:

$$C(i) = \frac{1}{q\tau_i} \sum_{k=0}^{R} \Phi(k) \cdot \left(1 - \sum_{n=L}^{k} \beta(n) \right) \quad (17)$$

The calculation of one row of the conversion matrix C(i) according to (17) requires only little extra computational effort because no additional vector-matrix multiplication is needed. At the k-th step the vector-matrix product $\Phi(k)$ and the truncated Poisson probability mass function $\sum_{n=L}^{k} \beta(n)$ have already been computed in (4) and (5). Thus, one iteration of formula (17) requires $O(\eta)$ operations. The total cost for calculating C(i) with the error tolerance ε by formula (17) is $O(q\tau_i\eta)$ floating point operations. As a consequence, the overall computation cost for calculating the time-dependent quantities associated with a particular state S_i enabling a deterministic transition with delay τ_i requires $O(q\tau_i\eta)$ floating point operations, too.

4 Application Examples

The benefit of employing the computational algorithm described in this paper is illustrated by empirical examples. The experiments have been performed on a Sun 4/65 Sparc™ station with 16 MByte main memory running the operating system SunOS4.1. We observed that employing the programming language C rather than Pascal already yields some reduction of the computation time required by the solution algorithm of GreatSPN. To obtain a fair comparison between the numerical method proposed and the numerical algorithm provided by GreatSPN this part of the source code of GreatSPN has been compiled from Pascal to C. Moreover, the component of GreatSPN which solves the

balance equations of the embedded Markov chain has been replaced by a sparse implementation of the direct Gaussian elimination algorithm [16]. For the performance tests the CPU time has been measured by employing appropriate systems calls. The analytical solutions of the considered DSPN models are calculated by employing the software package Mathematica™. Both solution algorithms perform each numerical calculation in double precision arithmetic. The minimum allowable step length for the multi-step method implemented in GreatSPN has been set to 0.3. In the Figures presented below the performance indices of the method provided by GreatSPN are shown by the dashed curves and attached with the label "ME". The corresponding performance indices of the proposed method are shown by the dotted curves and attached with the label "RA".

Even if most problems involving modeling of computer or communication systems require to consider multiple units, the first set of experiments considers a single-server queue. Figure 1 depicts a DSPN model for a single-server queueing system with Erlangian input, constant service, and limited waiting room. According to Kendall's notation this queueing system is referred to as $E_r/D/1/K$. The submodel representing the Erlangian arrivals is taken from [5]. The exponential transitions T2 and T3 are associated with a firing rate of $r\lambda$. The output arc from transition t1 to place P2 and the input arc from place P3 to transition T3 have the multiplicity r-1. Tokens contained in place P4 represent customers waiting in the queue or currently being served. The constant service requirement is modeled by the deterministic transition T4 with a firing delay of τ. The DSPN model has rK+1 tangible markings which can be classified as follows. One marking enables the deterministic transition exclusively, r markings enable only exponential transitions, and r(K-1) markings enable one exponential transition concurrently with the deterministic transition. An analytical solution for the steady-state distribution of the number of customers can be derived from the steady-state solution of the M/D/1/K queueing system [10]. In the presented tests an arrival rate of $\lambda = 9$ and a mean service time of $\tau = 0.1$ is considered.

A DSPN model of a fault-tolerant clocking system is shown in Figure 2. The number of clocking modules is represented by the marking parameter m depicted in place P1 of Figure 2. Unless otherwise stated the model

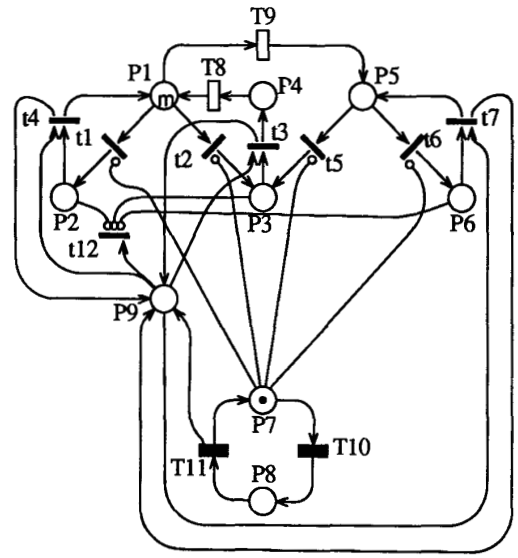

Figure 2. DSPN model of a fault-tolerant clocking system from [12]

parameters of this DSPN are chosen as follows: $\alpha = 0.14$, $\beta = 0.017$, $T = 20$, $\delta = 4$, $\lambda = 10^{-6}$, and $\mu = 10^{-3}$. A detailed description of the behavior of this

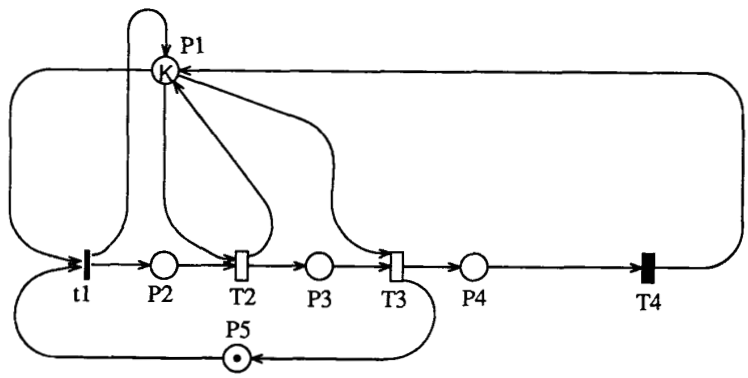

Figure 1. DSPN model of the $E_r/D/1/K$ queue

DSPN is given in [12]. A revised analytical solution of this DSPN in case of m = 1 is presented in an appendix.

Figure 3 plots the amount of CPU time required for calculating the steady-state solution of the $E_r/D/1/K$ model for both computational methods. The number of phases is set to 10 and the number of buffers varies from 10 to 100. The solution is calculated with an error tolerance of $\varepsilon = 10^{-12}$. As shown in Figure 3 the employment of the described solution algorithm requires substantially less computational effort than the solution algorithm implemented in GreatSPN. Note, that the

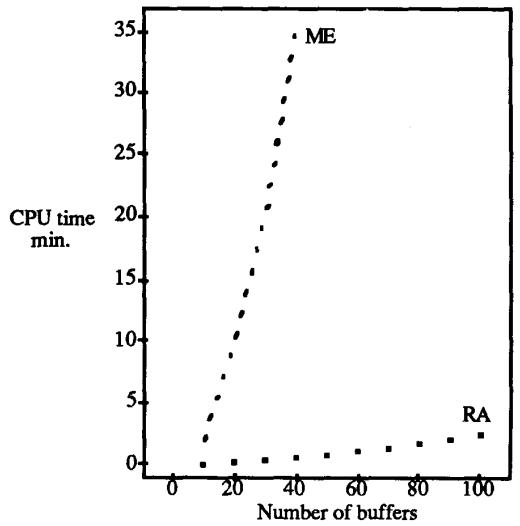

Figure 3. $E_r/D/1/K$ model - CPU time versus number of buffers

algorithm of GreatSPN requires about 35 minutes of CPU time for 30 buffers whereas the proposed algorithm needs about 2 minutes of CPU time for 100 buffers. Moreover, the algorithm of GreatSPN requires substantially more main memory space which additionally limits its applicability.

To consider a machine-independent measure for the computational cost Table 1 shows the number of vector-matrix-multiplication required by both solution algorithms. A linear growth is observed for both methods. But for each additional phase the algorithm of GreatSPN requires 16300 additional multiplications whereas the refined randomization method requires only 3700 additional multiplications. Hence, for this DSPN model the linear growth of the computational cost of the refined randomization method is nearly 5 times smaller than the linear growth of the method implemented in GreatSPN. Due to the large number of markings in which at least one exponential transition is concurrently enabled with the deterministic transition T4 this DSPN clearly shows the benefit of the described algorithm.

In the next set of experiments the number of buffers is kept fixed to K = 50 and the number of exponential phases of the Erlangian arrival stream is varied from 1 to 10. The solution is calculated with an error tolerance of $\varepsilon = 10^{-12}$. Figure 4 presents curves relating the number of phases to the amount of CPU time required by the both numerical method.

In Figure 5 the computational effort required for solving this DSPN model is related to the achieved accuracy of the solution. The number of buffers is kept

K	Tangible markings of the DSPN	Multiplications of matrix exponentiation method	Multiplications of randomization method
10	101	14577	3367
20	201	30877	7067
30	301	47177	10767
40	401	63477	14467
50	501	79777	18167
60	601	-	21867
70	701	-	25567
80	801	-	29267
90	901	-	32967
100	1001	-	36667

Table 1. Comparison of the Vector-Matrix-Multiplications required by the two computational methods

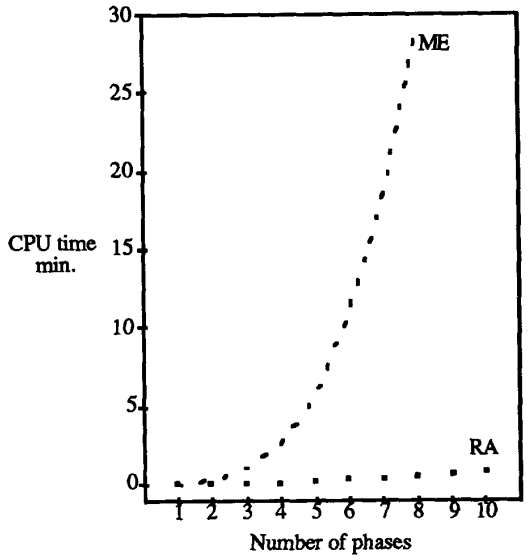

Figure 4. E_r/D/1/K model - CPU time versus number of phases

fixed to K = 50 and the number of phases is set to r = 5. Again, the refined randomization method requires substantially less computational effort than the adaptive matrix exponentiation method provided by GreatSPN.

Moreover, an experiment has been performed relating the achieved numerical accuracy as a function of the service time τ. The number of buffers is kept fixed to K = 50, and the number of phases is 5. For the range $0.1 \leq \tau \leq 2$ both solution algorithms achieve the accuracy of 10^{-11}. But the algorithm provided by GreatSPN requires significantly more CPU time to calculate the solution than the proposed numerical algorithm. We observed that the computational cost of the adaptive matrix exponentiation method of GreatSPN is very sensitive to changes in the model parameters. Due to the left-truncation of the computational formulas introduced in section 3 the computational cost of the refined randomization algorithm is rather insensitive to changes in the model parameters.

Figure 6 shows the computational effort required for solving the DSPN model of the fault-tolerant clocking system. The number of clocks vary from m = 1 to 20. The solution has been calculated by both methods with an error tolerance of $\varepsilon = 10^{-12}$. In case of 20 clocks the DSPN has 2002 tangible markings. In this case the calculation of the state transition rates of embedded Markov chain and the conversion factors requires about 30 minutes of CPU time by the refined randomization method The algorithm of GreatSPN needs already 26 minutes in case of 14 clocks for which the DSPN of Figure 2 contains 800 tangible markings.

Figure 7 show the computational effort required for solving this DSPN model with different accuracy of the solution. For this experiment the number of clocks is kept fixed to m = 10. We observe that for increasing accuracy the computational effort of the refined randomi-

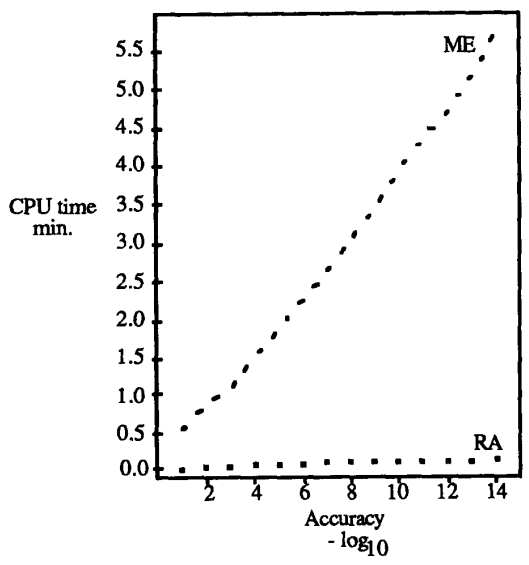

Figure 5. E_r/D/1/K model - CPU time vs. accuracy

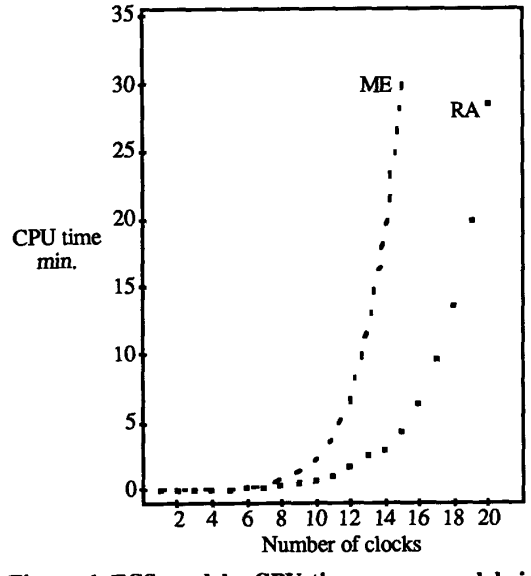

Figure 6. FCS model - CPU time versus model size

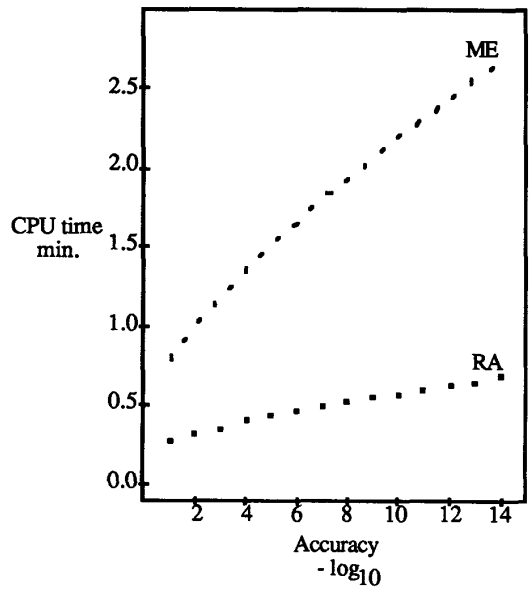

Figure 7. FCS model - CPU time vs. accuracy

zation method grows significantly slower than the effort of the adaptive matrix exponentiation method provided by GreatSPN.

As in case of the $E_r/D/1/K$ model experiments have been performed to relate the achieved numerical accuracy and the required computational effort to the parameter setting of the model. A set of experiments considers the achieved numerical accuracy as a function of the firing delay T associated with the deterministic transition T10 which represents length of the non-detection interval. The number of clocks is kept fixed to m = 10 and an error tolerance of $\varepsilon = 10^{-12}$ is considered. For the range $1 \le T \le 100$ the left truncated randomization method and the adaptive matrix exponentiation method of GreatSPN achieve the accuracy of 10^{-11}. But again the computational cost of the method provided by GreatSPN grows rapidly with increasing firing delay whereas the effort required by the proposed method remains nearly constant.

Conclusions

This paper presents an improved numerical algorithm for calculating the time-dependent quantities required by the solution process of DSPN models. Computational formulas for computing the state transition probabilities of the embedded Markov chain of a DSPN and the corresponding conversion factors have been derived in (10) and (17), respectively. The main features of the described algorithm are its efficient implementation and its excellent numerical stability due to the employment of existing special-purpose methods. As a consequence the described numerical algorithm is rather insensitive to changes in the model parameters. Moreover, it has a better error control than the multi-step method provided by GreatSPN. This does not become evident for DSPN models of small size in which all firing delays of timed transitions are in the same order of magnitude. But DSPN models of moderate size in which firing delays differ in several orders of magnitude (e.g. an $E_{10}/D/1/100$ queue with $\lambda = 9$ and $\tau = 10$) cannot be solved in practise with the adaptive matrix exponentiation algorithm implemented in the version 1.4 of the software package GreatSPN. Due to the employment of the randomization technique enhanced by a stable calculation of Poisson probabilities the proposed algorithm correctly calculates steady-state solution of such stiff models with reasonable computational effort.

The presented results indicate that the computational costs of both the refined randomization algorithm and the matrix exponentiation algorithm of GreatSPN grow asymptotically linearly with the product $q\tau_i\eta$. But as shown for the $E_r/D/1/K$ model the absolute computational effort of the described algorithm is substantially smaller than the absolute effort of the algorithm of GreatSPN. This is due to the large number of tangible markings in which exponential transitions are competitively or concurrently enabled with the deterministic transition. The ratio of such markings to the total number of tangible markings is smaller for the DSPN model of the fault-tolerant clocking system. Moreover, the state space cardinality of this model grows more than linear for increasing number of clocks. As a consequence the saving in computational cost of the randomization algorithm is still significant for this DSPN model, but not as evident as in case of the $E_r/D/1/K$ model. To reduce the growth of the state space a decomposition of a DSPN model into subnets of exponential and immediate transitions may be performed such that the behavior of each subnet is independent from the deterministic transition with delay τ_i currently enabled [4]. The transient solution of these subnets at the instant of time τ_i can also be effectively calculated with the algorithm described in this paper.

Hence, a combination of the described numerical algorithm and the proposed automated decomposition technique on the net level provide a powerful solution algorithm for DSPN models. Furthermore, since the Markov chain defined by exponential transitions competitively or concurrently enabled with a deterministic transition is often acyclic, it seems promising to investigate the applicability of the algorithm proposed in [13] to the DSPN solution process.

References

[1] M. Ajmone Marsan, G. Balbo, and G. Conte, "A Class of Generalized Stochastic Petri Nets for the Performance Analysis of Multiprocessor Systems", *ACM Trans. on Comp. Systems, 2*, pp. 93-122, 1984.

[2] M. Ajmone Marsan and G. Chiola, "On Petri Nets with Deterministic and Exponentially Distributed Firing Times", in: *G. Rozenberg (Ed.) Advances in Petri Nets 1986, Lecture Notes in Computer Science 266*, pp. 132-145, Springer 1987.

[3] M. Ajmone Marsan, G. Chiola, and A. Fumagalli, "An Accurate Performance Model of CSMA/CD Bus LAN", in: *G. Rozenberg (Ed.) Advances in Petri Nets 1986, Lecture Notes in Computer Science 266*, pp. 146-161, Springer 1987.

[4] M. Ajmone Marsan, G. Chiola, and A. Fumagalli, "Improving the Efficiency of the Analysis of DSPN Models", in: *G. Rozenberg (Ed.) Advances in Petri Nets 1989, Lecture Notes in Computer Science 424*, pp. 30-50, Springer 1990.

[5] P. Chen, S.C. Bruell, and G. Balbo, "Alternative Methods for Incorporating Non-exponential Distribution into Stochastic Timed Petri Nets", *Proc. 3rd Int. Workshop on Petri Nets and Performance Models, Kyoto Japan*, pp. 186-197, 1989.

[6] G. Chiola, "A Graphical Petri Net Tool for Performance Analysis", *Proc. 3rd Int. Conf. on Modeling Techniques and Tools for Performance Analysis, Paris France*, pp. 323-333, 1987.

[7] G. Chiola, "GreatSPN 1.5 Software Architecture", *Proc. 5th Int. Conf. on Modeling Techniques and Tools for Performance Analysis, Torino Italy*, pp. 117-132, 1991.

[8] G. Ciardo, J. Muppala, K.S. Trivedi, "SPNP: Stochastic Petri Net Package", *Proc. 3rd Int. Workshop on Petri Nets and Performance Models, Kyoto Japan*, pp. 142-151, 1989.

[9] B.L. Fox and P.W. Glynn, "Computing Poisson Probabilities", *Comm. of the ACM, 31*, pp. 440-445, 1988.

[10] D. Gross and C.M. Harris, *"Fundamentals of Queueing Theory", 2nd Edition*, John Wiley & Sons, 1985.

[11] D. Gross and D.R. Miller, "The Randomization Technique as a Modeling Tool and Solution Procedure for Transient Markov Processes", *Operations Research, 32*, 345-361, 1984.

[12] M. Lu, D. Zhang, and T. Murata, "Analysis of Self-Stabilizing Clock Synchronization by Means of Stochastic Petri Nets", *IEEE Trans. on Computers, 39*, pp. 597-604, 1990.

[13] R.A. Marie, A.L. Reibman, and K.S. Trivedi, "Transient Analysis of Acyclic Markov Chains", *Performance Evaluation, 7*, pp. 175-194, 1987.

[14] A.L. Reibman and K.S. Trivedi, "Numerical Transient Analysis of Markov Models", *Computers & Operations Research, 15*, pp. 19-36, 1988.

[15] W.H. Sanders and J.F. Meyer, "METASAN: A Performability Evaluation Tool based on Stochastic Activity Networks", *Proc. of the ACM-IEEE Comp. Soc. 1986 Fall Joint Conf.*, pp. 807-816, 1986.

[16] A.H. Sherman, "NSPIV, A FORTRAN Subroutine for Sparse Gaussian Elimination with Partial Pivoting", *ACM Trans. on Math. Softw., 4*, pp. 391-398, 1978.

[17] V.L. Wallace and R.S. Rosenberg, "Markovian Models and Numerical Analysis of Computer Systems Behavior", *Proc. of the ACM-IEEE Comp. Soc. Spring Joint Comp. Conf.*, pp. 141-148, 1966.

Appendix

The closed-form solution presented in [12] neglects the state transition caused by firing of *both* exponential transitions T8 and T9 during the enabling interval of the deterministic transition T10. This appendix presents a revised analytical solution of this DSPN which is used for determining the numerical accuracy achieved by the described algorithm and the algorithm implemented in GreatSPN. The notation of the state transition probabilities of the embedded Markov chain is similar as in [12]. Figure 9 shows the revised state transition graph of the embedded Markov chain.

Note, a state transition from S_4 to S_7 has been added with a transition probability βz_4. The value for z_4 can be derived from a hypoexponential distribution with parameters μ and λ or from a 2-phase Erlang distribution with parameter μ depending on whether or not $\mu \neq \lambda$. Moreover, the state transition probability $P(S_4 \rightarrow S_3)$ and the factor y_4 have been modified.

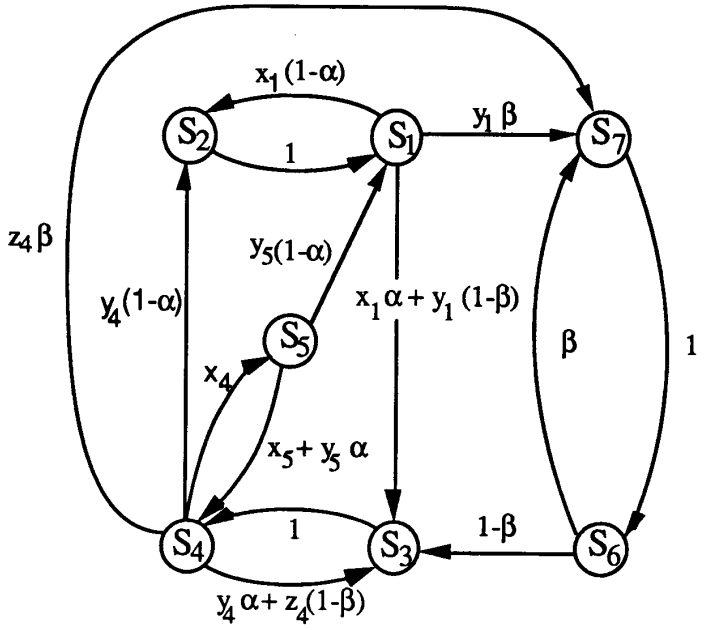

Figure 8. The revised embedded Markov chain of the DSPN from [12]

$$x_1 = e^{-\lambda T}$$

$$y_1 = 1 - e^{-\lambda T}$$

$$x_4 = e^{-\mu T}$$

$$y_4 = \begin{cases} \frac{\mu}{\lambda - \mu}\left(e^{-\mu T} - e^{-\lambda T}\right) & \mu \neq \lambda \\ \mu T e^{-\mu T} & \mu = \lambda \end{cases}$$

$$z_4 = \begin{cases} 1 - \frac{1}{\lambda - \mu}\left(\lambda e^{-\mu T} - \mu e^{-\lambda T}\right) & \mu \neq \lambda \\ 1 - e^{-\mu T} - \mu T e^{-\mu T} & \mu = \lambda \end{cases}$$

$$x_5 = e^{-\mu \delta}$$

$$y_5 = 1 - e^{-\mu \delta}$$

As a consequence, the formulas of the conversion factors of state S_4 are modified as follows:

$$w_{44} = \int_0^T e^{-\mu\theta} d\theta$$

$$w_{41} = \begin{cases} \int_0^T \frac{\mu}{\lambda - \mu}\left(e^{-\mu\theta} - e^{-\lambda\theta}\right) d\theta & \mu \neq \lambda \\ \int_0^T \mu\theta e^{-\mu\theta} d\theta & \mu = \lambda \end{cases}$$

$$w_{46} = \begin{cases} \int_0^T \left(1 - \frac{1}{\lambda - \mu}\left(\lambda e^{-\mu\theta} - \mu e^{-\lambda\theta}\right)\right) d\theta & \mu \neq \lambda \\ \int_0^T \left(1 - e^{-\mu\theta} - \mu\theta e^{-\mu\theta}\right) d\theta & \mu = \lambda \end{cases}$$

Subsequently, the linear system of the balance equations is given by:

$$\pi_1 = \pi_2 + y_5(1-\alpha)\pi_5$$

$$\pi_2 = x_1(1-\alpha)\pi_1 + y_4(1-\alpha)\pi_4$$

$$\pi_3 = \left[x_1\alpha + y_1(1-\beta)\right]\pi_1 + \left[y_4\alpha + z_4(1-\beta)\right]\pi_4 + (1-\beta)\pi_6$$

$$\pi_4 = \pi_3 + (x_5 + y_5\alpha)\pi_5$$

$$\pi_5 = x_4\pi_4$$

$$\pi_6 = \pi_7$$

$$\pi_7 = y_1\beta\pi_1 + z_4\beta\pi_4 + \beta\pi_6$$

$$1 = \pi_1 + \pi_2 + \pi_3 + \pi_4 + \pi_5 + \pi_6 + \pi_7$$