TimeNET – A Toolkit for Evaluating Non-Markovian Stochastic Petri Nets

Reinhard German*, Christian Kelling, Armin Zimmermann**, and Günter Hommel

Technische Universität Berlin
Institut für Technische Informatik
Fachgebiet Prozeßdatenverarbeitung und Robotik
(Real-Time Systems and Robotics)
Franklinstr. 28/29
10587 Berlin, F. R. Germany

Abstract

This paper describes TimeNET (Timed Net Evaluation Tool), a software package for the modeling and evaluation of stochastic Petri nets with non-exponentially distributed firing times. TimeNET has been developed at the Technische Universität Berlin in several research projects. A graphical user interface is provided for the model specification and several specialized analysis and simulation components are used for the automated model evaluation. The implementation of the analysis and simulation components is based on recent research results. Both the general structure and the underlying algorithms of TimeNET are described. An example illustrates the modeling and evaluation process using TimeNET.

Keywords: Performance and Dependability Modeling Tool

Analysis and Simulation of Stochastic Petri Nets

Graphical User Interface

---

* R. German's work was supported by the Siemens Corporate Research and Development.

** A. Zimmermann's work was supported by the German Research Council under grant Ho 1257/4-1.

TimeNET – A Toolkit for Evaluating Non-Markovian Stochastic Petri Nets

1 Introduction

Stochastic Petri nets (SPNs) represent a graphical method for the modeling of discrete event systems like computer systems, communication systems, and manufacturing systems. The stochastic extensions to the pure Petri net formalism allow modeling and evaluating the performance and dependability of these systems. In order to deal with realistic models, software tools are needed which support the modeling and evaluation process. This paper describes the new software package TimeNET (Timed Net Evaluation Tool) which provides several specialized components for dealing with SPNs with non-exponentially distributed firing times.

During the last decade several classes of SPNs with different modeling power were proposed. Most commonly, the transitions may fire after an exponentially distributed delay. SPNs with both exponentially timed and immediate transitions firing without delay are referred to as *generalized stochastic Petri nets* (GSPNs, [7]). The underlying stochastic process of a GSPN is a continuous-time Markov chain. Several variants of SPNs with an underlying Markov chain have been proposed in the literature. Accordingly, we refer to this class of SPNs as *Markovian SPNs*. Markovian SPNs have been broadly used and accepted due to the availability of software packages which completely automate their solution process (e.g. GreatSPN [4], [6], SPNP [10], UltraSAN [11], and TOMSPIN [23]). However, the Markov assumption is not realistic in many cases. Delays may either be deterministic or may be non-exponentially distributed. Therefore non-Markovian SPNs are also required. *Deterministic and stochastic Petri nets* (DSPNs, [2]) allow exponentially distributed and deterministic firing delays. A numerical steady-state analysis is possible, if in each marking no more than one transition with a deterministic delay is enabled [2]. Therefore, transitions with a deterministic firing delay must not be concurrently enabled in a DSPN.

Based on the graphical user interface of GreatSPN, the software tool DSPNexpress was developed at the Technische Universität Berlin. DSPNexpress is especially tailored to the analysis of DSPNs and uses a refined, more efficient solution algorithm [24, 25]. It can also deal with marking-dependent deterministic firing delays [27], which can be used for the modeling of activities with a varying speed. The refined DSPN solution algorithm was also added to UltraSAN.

Recently, new analysis methods were proposed for more general non-Markovian SPNs. References [15, 8, 9] present general solution formulas for the steady-state analysis of SPNs with exponentially and generally distributed firing delays. Similar to DSPNs, the solution is subject to the restriction, that in each marking no more than one transition with a non-exponentially distributed firing delay is enabled. We refer to this class of SPNs as *extended DSPNs*. [15] and [9] present efficient numerical solution formulas for extended DSPNs in case the general distributions belong to a special class of distributions, called *expolynomial distributions*. Expolynomial distributions can be piecewise defined by exponential polynomials. The class of expolynomial distributions contains many known distributions as special cases (e.g. deterministic delay, uniform distribution, triangular distribution, truncated exponential distribution, finite discrete distribution) and allows the approximation of practically any distribution.

In case of DSPNs with concurrently enabled deterministic transitions (referred to as *concurrent DSPNs*) a new method for the approximate steady-state analysis was presented in [13]. Using pseudo-probabilities outside the interval [0,1] a generalized phase expansion with few phases leads to a good approximation.

If an SPN is still too complex for analysis, it is possible to use discrete event simulation for the evaluation. Simulation can deal with large state spaces and is not restricted to a certain class of SPNs but may cause long run times. In [18, 20, 21] it was proposed to use parallelization and variance reduction by control variates to achieve significant speedups.

TimeNET has been developed at the Technische Universität Berlin and is an extension of the software tool DSPNexpress. It provides a user-friendly graphical interface for the modeling of non-Markovian SPNs with deterministic, exponentially distributed, and expolynomially distributed firing delays. Depending on the class of the SPN, different analysis, approximation, and simulation components can be used for the evaluation. Analysis methods for DSPNs and extended DSPNs [15, 9], an approximation method for concurrent DSPNs [13], and a fast simulation method for arbitrary SPNs [18, 20, 21] are available. For the structural analysis and the generation of the reachability graph specialized efficient algorithms are used [29]. The contribution of TimeNET is that it contains new analysis and approximation components as well as fast simulation components, providing a unified framework for the evaluation of non-Markovian SPNs. It runs on Sun and DEC Alpha workstations under the X11 window system.

TimeNET and its predecessor DSPNexpress have been successfully applied for several modeling projects. [28] contains a performance analysis of a virtually shared memory architecture, [26] presents the performability analysis of a flexible manufacturing system, and [19] evaluates real-time properties of communication systems. There are other ongoing projects together with the Siemens AG and with the Daimler–Benz AG.

The remainder of this paper is organized as follows. Section 2 describes existing software packages and gives a motivation for developing TimeNET. Section 3 contains a description of the general structure of TimeNET and short descriptions of the implemented algorithms. Section 4 illustrates the modeling and evaluation of a queuing system using TimeNET. Concluding remarks are given in Section 5.

2 Existing Software Packages for the Evaluation of SPNs

In this section several software packages are described which are closely related to TimeNET. At the University of Torino the software package Graphical Editor and Analyzer for Timed and Stochastic Petri Nets (GreatSPN, [4], [6]) was developed. GreatSPN provides a user-friendly graphical interface running under XView which allows to interactively edit, validate, and evaluate SPN models. Version 1.4 supports the transient and steady-state analysis of GSPNs and the steady-state analysis of DSPNs [4]. In version 1.5 the DSPN analysis component was removed due to its high computational costs and a component for simulation has been added [6]. Moreover, several “compiling techniques” [5] have been exploited in order to improve the efficiency of the reachability graph generation. Additionally, GreatSPN supports SPNs with colored tokens. The development of the user interface of TimeNET has been influenced by GreatSPN. Opposed to GreatSPN, TimeNET contains components especially tailored to the evaluation of non-Markovian SPNs.

At the Duke University, the software package Stochastic Petri Net Package (SPNP, [10]) was developed. SPNP contains components for the transient and steady-state analysis of GSPNs. The entire package has been implemented in C and data structures are managed dynamically. SPNP uses an alphanumerical interface, models have to be specified in a C-based description language which allows a very high modeling flexibility. Moreover, very general reward specifications can be given. SPNP also comprises a component for sensitivity analysis. However, SPNP does not include a graphical user interface and cannot deal with non-Markovian SPNs.

At the University of Arizona, the software package UltraSAN was implemented [11]. UltraSAN provides a user-friendly graphical interface running under X11 and components for the transient and steady-state analysis of Stochastic Activity Networks (SANs). SANs are similar to SPNs and include immediate and timed activities. UltraSAN uses *reduced base models* for deriving the measures of interest. UltraSAN provides transient and steady state analysis for SANs with exponential timing and also a steady-state analysis component for SANs with exponential and deterministic timing. Moreover, this tool comprises a component for the transient and steady-state simulation of SANs with non-exponentially timed activities. Although UltraSAN can deal with models containing both exponentially distributed and deterministic timing, it can not analytically evaluate models with more general timing. The simulation component of UltraSAN uses importance sampling whereas TimeNET uses variance reduction by control variates. Furthermore, TimeNET can execute single simulation runs in parallel.

Since 1991 the software package DSPNexpress [25] has been developed at the Technische Universität Berlin. DSPNexpress provides a user-friendly graphical user interface running under X11 and is especially tailored to the steady-state analysis of DSPNs. Transient and steady-state analysis of GSPNs is also provided. DSPNexpress uses a refined numerical DSPN-solution algorithm [24] which can be executed in parallel. However, DSPNexpress cannot deal with more general firing times than exponentially distributed and deterministic times. Furthermore no simulation component is provided. One drawback is therefore that no quantitative evaluation is possible in case transitions with a deterministic delay are concurrently enabled.

The software package TimeNET is a major revision of DSPNexpress. It contains all components of DSPNexpress, but supports the specification and evaluation of SPNs with an increased modeling power. TimeNET allows for the definition of expolynomially distributed firing delays of transitions. Therefore non-Markovian SPNs can be specified. Depending on the class a specified SPN belongs to, different evaluation components can be used. In case of an extended DSPN (in each marking at most one non-exponentially timed transition), TimeNET can compute the steady-state solution using the algorithm described in [14, 9]. In case of a concurrent DSPN (exponentially and deterministically timed transitions without structural restrictions), no analytical solution method is known. TimeNET can compute an approximate steady-state solution using the method described in [13]. In more general cases, if either the structural restrictions do not hold or if the state space becomes too large, TimeNET provides also a fast simulation component for obtaining quantitative results. The simulation component uses the results published in [18, 20, 21]. Since analysis, approximation, and simulation is performed for the same class of models, TimeNET provides a unified framework for the modeling and evaluation of non-Markovian SPNs. The graphical user interface is based on X11 running on Sun and DEC Alpha workstations.

The development of TimeNET was also influenced by two other software packages: ESP [12] and TOMSPIN [23]. ESP was written for the analysis of SPNs with phase-type distributed firing times. Although ESP cannot deal with immediate transitions, the basic algorithm was used and extended for the approximation component of TimeNET. TOMSPIN was developed by the Siemens Corporate, Research and Development. TOMSPIN provides an alphanumerical interface and components for the transient and steady-state analysis of GSPNs and can deal with very large models.

3 Description of TimeNET

The major components of TimeNET are the graphical user interface and the evaluation components. This section is organized as follows. In Section 3.1 the considered classes of SPNs are introduced and in Sections 3.2, 3.3, and 3.4 the analysis, approximation, and simulation components are described, respectively.

3.1 The Considered Classes of SPNs

TimeNET uses the customary SPN formalism as e.g. in [1]. A SPN consists of *places* and *transitions*, which are connected by *input*, *output*, and *inhibitor arcs*. In the graphical representation, places are drawn as circles, transitions are drawn as thin bars or as rectangles, and arcs are drawn as arrows (inhibitor arcs have a small circle at their destination). Places may contain undistinguishable *tokens*, which are drawn as dots. The vector representing the number of tokens in each place is the state of the SPN and is referred to as *marking*. The marking changes by the *firing* of the transitions. A marking-dependent multiplicity can be associated with each arc. Places that are connected with a transition by an arc are referred to as *input*, *output*, and *inhibitor places* of the transition, depending on the type of the arc. A transition is said to be *enabled* in a marking if each input place contains at least as many tokens as the multiplicity of the input arc and if each inhibitor place contains fewer tokens than the multiplicity of the inhibitor arc. A transition *fires* by removing tokens from the input places and adding tokens to the output places according to the multiplicities of the corresponding arcs. The *reachability graph* is defined by the set of vertices corresponding to the markings reachable from the initial marking and the set of edges corresponding to the transition firings. The transitions can be divided into *immediate transitions* firing without delay (drawn as thin bars) and *timed transitions* firing after a certain delay (drawn as rectangles). Immediate transitions have priority over timed transitions. Accordingly, the markings can be partitioned into *vanishing* and *tangible markings* and the *reduced reachability graph* is defined by the set of vertices corresponding to the reachable tangible markings and the set of edges given by the corresponding transition firings.

Stochastic specifications are added to the formalism such that a stochastic process is underlying an SPN. Possible conflicts between immediate transitions are resolved by assigning weights to them. The firing delays of the timed transitions are specified by deterministic delays or by random variables. Important cases are transitions with a deterministic delay (drawn as filled rectangles), with an exponentially distributed delay (drawn as empty rectangles), and with a generally distributed delay (drawn as dashed rectangles). In case of non-exponentially distributed firing delays firing policies have to be specified [1]. We assume that each transition restarts with a new firing time after being disabled, corresponding to “race with enabling memory” as defined in [1], although some of our algorithms can also deal with “race with age memory”.

In TimeNET a certain class of general distributions is used, referred to as *expolynomial distributions* [15, 9]. An expolynomial distribution can be piecewise defined by exponential polynomials and has finite support. An expolynomial distribution may contain jumps, therefore it can represent random variables with mixed continuous and discrete components. The class of expolynomial distributions contains many known distributions (e.g. deterministic delay, uniform distribution, triangular distribution, truncated exponential distribution, finite discrete distribution), allows the approximation of practically any distribution (e.g. by using splines), and is particularly well suited for the numerical analysis (see next section). Since the probability mass function (pmf) seems to be graphically more significant for the user than the cumulative distribution function (CDF), we decided to use the pmf for the specification in TimeNET. A context-free grammar was defined for the specification of expolynomial distributions. Using the Unix tools lex and yacc, a parser for expolynomial distributions was written and added to TimeNET.

In order to clarify the names for the different classes of SPNs, a short summarization is given in this paragraph. In *generalized stochastic Petri nets* (GSPNs) only exponentially timed transitions are allowed. In *deterministic and stochastic Petri nets* (DSPNs) also deterministically timed transitions are allowed under the restriction that at most one such transition is enabled in each marking. The restriction is caused by the numerical analysis method. TimeNET allows the evaluation of more general model classes: In *extended DSPNs* in each marking at most one expolynomially timed transition may be enabled. In *concurrent DSPNs* exponentially and deterministically timed transitions may be enabled without restrictions. The most general case, exponentially, deterministically, and expolynomially timed transitions without restrictions is simply referred to as SPNs.

3.2 The Analysis Component

The analysis of an SPN consists of several steps. First the structure of the SPN is examined. Based on that information the reachability graph can be efficiently generated. Depending on the type of the SPN different algorithms are then used for the numerical analysis.

3.2.1 Structural Analysis

The first analysis step carried out by TimeNET is the computation of several structural properties of the given model. Examining these properties, the modeler can check whether the model has been specified correctly or not. Additionally, some of the structural properties are used for the efficient generation of the reachability graph, using the idea described in [29, 5, 7]. Since these properties can be obtained directly from the net structure, the computational cost is very small in relation to the further analysis steps.

The algorithm proposed in [30] is employed to obtain the minimal-support place invariants of the net. From these invariants an upper bound of the number of tokens is derived for each place, which is used for a space-efficient storing of the reachable markings following [5]. Furthermore, the *extended conflict sets* of immediate transitions are computed. For this reason, the concepts of *causally connectedness* and *indirect conflict* are used as described in [7]. Because *mutually exclusive* transitions cannot be in conflict, this property has to be checked for a correct computation of the e*xtended conflict sets*. Two transitions are mutually exclusive, if they are *structural* mutual exclusive, *marking* mutual exclusive or if they have a different priority. The structural mutual exclusiveness can be checked on the net structure, while for the marking mutual exclusive transitions the place invariants have to be considered.

Furthermore, the net is tested to be *confusion free*, because the reachability graph generation algorithm employed in TimeNET requires the absence of confusions (see below). We extended the definition of *confusion freeness* from [7] to cope with marking-dependent arc cardinalities and marking-dependent firing weights for immediate transitions [29]. It is possible to examine the computed *extended conflict sets* and the *minimal-place invariants* to allow the modeler to check the correctness of the SPN model at an early stage of the analysis process.

Note that the results of the structural analysis are valid, even though the considered class of SPNs allows for priorities of transitions, inhibitor arcs, and finite support of firing time distributions. All place invariants are still valid in presence of these model features. However, it may happen that the list of detected place invariants is not exhaustive. All other structural properties of the net, e.g. conflicts and confusions, are only checked for immediate transitions. The results of the structural analysis are therefore valid for non-Markovian SPNs, even if the firing time distributions of timed transitions have finite support.

3.2.2 Efficient Generation of the Reachability Graph

The main problem with the generation of the reachability graph of a Petri net lies in the computational effort and required memory space. To overcome the conflict between a fast execution time and a small main memory space usage, TimeNET provides two variants of the algorithm: a *time efficient* and a *space efficient* one. Both variations of the algorithm are based on [3], generating the reduced reachability graph of a timed Petri net model after a decomposition of the net into subnets by removing the timed transitions. The timed transitions constitute a barrier between the subnets of immediate transitions. This is still the case for non-exponentially timed transitions; therefore the method is applicable for non-Markovian SPNs as well.

The memory requirements of a reachability graph generation algorithm should not exceed the main memory available on a given workstation. This hampers the employment of the time-efficient generation algorithm for some models, even if the “compiling techniques” introduced in [5] are used to optimize the data structures. Using the method described in [29], the memory requirements of the reachability graph generation algorithms could be estimated, resulting in an a-priori selection of the appropriate variation. This estimation exploits the place invariants of the model, which can be derived directly from the net structure. If the estimated memory requirements exceed the main memory on a given workstation, the space efficient variation is used.

The time efficient variation of the algorithm computes and stores all possible firing paths from a vanishing submarking to its reachable tangible submarkings together with their associated probabilities. Thus, the set of reachable tangible submarkings and the corresponding probability vector has to be derived only once even for multiply visited submarkings.

In the case of the space efficient variation of the reachability graph algorithm, the memory required for the reachability graph of a subnet is released immediately after this subnet has been processed, resulting in substantially less main memory space usage.

Additionally, our algorithm uses independent firing of immediate transitions, which reduces the number of possible firing sequences while examining vanishing submarkings. This method is only applicable in a confusion free net, because in this case, the subsequent behavior of the model does not depend on the order of firing of the immediate transitions enabled in a given marking.

3.2.3 Numerical Analysis

If the SPN is either a GSPN, a DSPN or an extended DSPN, numerical analysis is possible. In case of a *GSPN* all timed transitions have an exponentially distributed delay. The reduced reachability graph is isomorphic to a continuous-time Markov chain. The steady-state solution is obtained by solving the corresponding linear system of equations and the transient solution is obtained by solving the corresponding system of differential equations. In TimeNET successive over relaxation (SOR) [22] and sparse Gaussian elimination [33] are used for the steady-state solution and Jensen’s method, also called randomization or uniformization [16], is used for the transient solution.

In case of a *DSPN* (i.e. the SPN contains transitions with deterministic delay with the restriction that in each marking no more than one of these transitions is enabled), steady-state analysis is also possible [2]. The analysis is based on the observation that the underlying stochastic process enjoys the absence of memory at certain instants of time, referred to as *regeneration points* [27]. The definition of the regeneration points depends on whether a deterministically timed transition is enabled or not. In markings enabling only exponentially timed transitions the next regeneration point is chosen to be the instant after the next firing of a transition. In all other markings the next regeneration point is chosen to be the instant after the deterministically timed transition has fired or has become disabled by the firing of another transition. An *embedded Markov chain* (EMC) with a discrete time parameter can be defined at the regeneration points. The one-step transition probabilities of the EMC are denoted by the stochastic matrix **P**. The entry p_{ij} of **P** represents the probability of being in marking j in the next regeneration point given that the marking in the last regeneration point was i. The steady state solution γ of the EMC is given by the linear system of equations:

$$\boldsymbol{\gamma} \cdot \mathbf{P} = \mathbf{P}, \quad \boldsymbol{\gamma} \cdot \mathbf{e} = 1 \tag{1}$$

where **e** denotes a vector with all entries equal to one. Since the EMC does not reflect the time spent in the markings up to regeneration, the solution has to be converted in order to obtain the solution of the DSPN. Therefore a matrix **C** of conversion factors is computed. The entry c_{ij} of **C** represents the average sojourn time in marking j up to regeneration given the marking in the last regeneration point was i. The solution π of the DSPN is obtained by multiplying the EMC solution by the matrix of conversion factors and a subsequent normalization step in order to obtain a vector of proper probabilities:

$$\gamma' = \gamma \cdot \mathbf{C}, \quad \pi = \frac{1}{\gamma' \cdot \mathbf{e}} \cdot \gamma' \tag{2}$$

The main problem of the algorithm lies in the computation of the entries of **P** and **C**. In case a deterministically timed transition is enabled, the possible evolution of the stochastic process during its enabling has to be taken into account. Since only exponentially timed transitions may fire during the enabling, this process is a continuous-time Markov chain, referred to as *subordinated Markov chain* (SMC) of the deterministically timed transition. Let the generator matrix of an SMC be denoted by **Q** and let τ denote the deterministic firing delay. The state probabilities in the SMC in the instant of the deterministic firing time and the average sojourn times in the states of the SMC up to the deterministic firing time are given by the transient solution and by the cumulative transient solution of the SMC, respectively:

$$e^{\mathbf{Q}\tau}, \quad \int_0^{\tau} e^{\mathbf{Q}t}\, dt \tag{3}$$

The matrix exponential and the integral of the matrix exponential have thus to be computed for the SMC of each deterministically timed transition. The result has then to be inserted into the matrices **P** and **C**. For the mapping of the transient quantities to the matrices **P** and **C**, we refer to [24] and [27]. In TimeNET, Jensen's method is used for the transient and cumulative transient analysis of the subordinated Markov chains. Furthermore, the algorithm detects isolated components of the SMCs, tests them for possible isomorphisms, and starts a process for the analysis for each component. Since the components are independent, the processes are executed in parallel on different workstations. For the computation of the solution of the EMC, SOR is used.

In [27] it was shown how these formulas can be generalized in order to deal with marking-dependent deterministic firing delays. Marking-dependent deterministic firing delays can be used for the modeling of activities with a constant speed which may vary due to other events (e.g. a fault-tolerant service facility with a slower performing redundant unit). For the proper definition of the regeneration points, the marking-dependent deterministic delays can be normalized and the rates of the exponentially distributed delays can be scaled appropriately. This algorithm is included in the analysis component of TimeNET.

In case of *extended DSPNs* (i.e. the SPN contains transitions with generally distributed delays with the restriction that in each marking no more than one of these transitions is enabled), steady-state analysis is still possible. The structural restriction is very similar to that of ordinary DSPNs: non-exponentially timed transitions must not be concurrently enabled. The analysis is either possible by the method of supplementary variables [15], or by means of an embedded Markov chain [8], [9]. Since the second algorithm is a straightforward extension of the described solution algorithm for DSPNs, we decided to implement this one. The definition of the regeneration points is very similar to the definition for ordinary DSPNs: in markings enabling only exponentially timed transitions the next regeneration point is chosen to be the instant after the next firing of a transition. In all other markings the next regeneration point is chosen to be the instant after the non-exponentially timed transition has fired or has become disabled by the firing of another transition. The definition of the EMC, the computation of the solution of the EMC, the conversion and normalization are identical to a DSPN. Only the computation of the one-step transition probabilities of the EMC and of the conversion factors is different. Let **Q** denote the generator matrix of the SMC of a transition with a generally distributed delay and let $F(t)$ denote its probability distribution function. The state probabilities of the SMC in the instant of firing are given by the Stieltjes integral of the matrix exponential with respect to $F(t)$ from zero to infinity; the average sojourn times in the states of the SMC up to firing are given by the ordinary integral of the matrix exponential multiplied with the complement of $F(t)$ from zero to infinity, respectively:

$$\int_0^\infty e^{\mathbf{Q}t}\, dF(t), \quad \int_0^\infty e^{\mathbf{Q}t} \cdot (1 - F(t))\, dt \tag{4}$$

The integrals (4) are a generalization of the expressions (3): substituting $F(t)$ by the unit step function in τ (which is a probability distribution representing a deterministic delay) leads to the expressions (3). The solution formulas are valid for arbitrary distributions. In [15, 9] it was shown how Jensen's method can be generalized for the efficient numerical computation of the integrals in case the general distribution is an expolynomial distribution. This algorithm was implemented and is provided by TimeNET.

3.3 The Approximation Component

The numerical steady-state analysis of DSPNs is subject to the restriction that deterministically timed transitions must not be concurrently enabled. TimeNET provides an algorithm for the approximate steady-state analysis in case that restriction is relaxed. The approximation is based on replacing the deterministically timed state transitions by sequences of exponential phases. Most commonly, an Erlang distribution, consisting of a sequence of identical phases, is used for the approximation. The replacement leads to an expanded state space which is isomorphic to a continuous-time Markov chain. The solution can be obtained by solving the corresponding linear system of equations. Appropriately summing up the results of the expanded states leads to an approximate steady-state solution of the DSPN. For good approximations a large number of phases of the Erlang distribution is required (e.g. 10 phases), leading to a state space explosion.

In order to avoid this state space explosion, it was proposed in [13] to use a a *generalized Cox distribution function* (GCDF) for the approximation. A GCDF consists also of a sequence of identical exponential phases, but has a switching probability after the completion of the first phase. Using *pseudo-probabilities* outside the interval [0,1] leads formally to a smaller variance than in the case of an Erlang distribution with the same number of phases. The replacement of the deterministic times by GCDFs leads again to an expanded state space. Since the GCDF comprises negative rates, the state space is isomorphic to a generalized continuous-time Markov chain. Solving the corresponding linear system and appropriate summing-up leads to an approximate solution. In some cases very good approximations can be obtained by using just two phases. Unfortunately, for some structures the approximation yields negative values and the result is useless. At the moment no criterion is known for the prediction whether the result is good or not. Therefore the approximation based on GCDFs can only be heuristically used and must be validated by simulation for a given DSPN.

In TimeNET deterministic delays can automatically be replaced either by Erlang distributed delays or by GCDFs. The underlying expanded state space is generated by an algorithm taken from [12]. The (generalized) continuous-time Markov chain is then solved and the results are automatically summed up.

3.4 The Simulation Component

The main problem of all analytical evaluation methods remains the size of the state space to be generated. Real-life models tend to be very detailed and the state space of their underlying stochastic process becomes unmanageable. Therefore in some situations simulation is the only feasible approach for performance evaluation. Furthermore, the simulation component can serve as a validation tool for new analysis methods. TimeNET contains a simulation component that has been designed to deal with complex Petri net models with non-exponentially distributed firing times.

3.4.1 Concepts

Simulation accompanies the whole design and evaluation process of the SPN model. While in early stages of the modeling an interactive simulation support is necessary in order to verify the model behavior, for the evaluation a fast and automated simulation component is required. The interactive tokengame of TimeNET implements an animated framework for testing and debugging purposes. It allows the user to control the firing sequences using the graphical net representation. The fast, automated simulation for reward measure estimations is realized by a second simulator.

In general, a SPN simulation is performed by executing the following steps cyclically:

- find enabled transitions
- if immediate transitions are enabled: according to priorities and weights choose one immediate transition to fire
  if timed transitions are enabled: according to the (random) firing delays choose one timed transition to fire
- fire the selected transition, i.e. compute the new marking
- update the statistics of the reward measures (if necessary)

This discrete-event simulation is a stochastic experiment. Therefore all samples drawn by the simulation procedure are random variates. Reward measures can be obtained by estimating the mean value of the sampled data. Assumptions concerning the precision of this estimate are usually based on confidence intervals derived from the sample variance. Since most statistical methods require that the input data satisfy special properties (e.g. independence, normal distribution) but obtained samples do not have these properties in general, some effort is indispensable to ensure valid results.

Even though the application of the simulator does not require special knowledge about the implemented methods, an outline of the techniques is given in the next section, and after this some implementation aspects of the simulator are discussed.

3.4.2 Statistical Methods

The simulation component contains statistical techniques for reliable and robust estimation of the user-defined reward measures:

- The user can specify the accuracy of the reward measure estimates giving the confidence level and the maximum relative error that can be tolerated in the final result.
- Samples drawn from the initial transient phase of a simulation run induce bias into the mean value estimate. Therefore, the initial transient phase is detected automatically and data is only taken from the steady-state stage. We apply a test proposed by Schruben et al. [32] preceded by a simple heuristic. The procedures are adopted from [31].
- Statistical analysis of the sample data is performed by spectral variance analysis [17, 31]. This technique with an enhancement developed in [18] allows flexible and robust variance estimates in single as well as multiple replication scenarios. It does not make any assumptions concerning the correlation structure and therefore it is also applicable if only a few replications are available.
- The simulation run length required for the pre-specified precision is determined automatically. Upper bounds for the run length and the number of samples can be given.

3.4.3 Implementation

Simulation experiments may be very time consuming. Several approaches have been proposed in order to reduce the computational overhead for simulation runs. We use different techniques to accelerate the traditional Petri net simulation.

An obvious way to obtain efficiency is to use structural properties of the SPN. Since the search for enabled transitions in a new marking may be very costly if all transitions are considered, so called *causally connectedness* is used to determine the transitions that can become enabled after a particular transition has fired. This reduces significantly the overhead to find new enabled transitions [5].

Parallelization is sometimes a well suited approach to speed up sequential programs. The simulator component is able to run multiple independent replications at several workstations in a LAN-environment. This means that several *simulation engines*, i.e. program components realizing the firing sequences of the model, are started in the distributed system and one centralized control instance collects and analyses the data samples. This is an easy way to achieve significant speedup for simulation experiments, as long as models are not too small and therefore the distribution overhead is greater than the simulation run length. The simulator applies UNIX standard communications techniques via sockets for the transmission of the data samples. Further details can be found in [21].

Figure 1 shows the achievable speedup for parallel runs. With speedup we denote the ratio of the execution time using one slave and the execution time using *n* slaves. In the considered case, one simulation engine generates 10,000 samples of the reward measure in 30 seconds of execution time. This indicates a reward measure obtained from relatively rare events of the model. The distribution overhead is approximately 5 seconds and the curves represent different number of samples required to obtain a predefined accuracy.

Variance reduction techniques (VRTs) are among the most promising methods to reduce the overhead of stochastic simulation experiments. Most VRTs require special knowledge about the simulation model under study and individually adapted simulation algorithms. This makes its incorporation into flexible, user-friendly simulation tools more difficult. One of the more generally applicable methods is the control variates VRT. The basic idea is to use the correlation between an estimator of interest and another stochastic parameter of the model (called the control variate CV) to reduce the variance of the estimator. We use special selection rules, also exploiting the net structure, for effective CVs and can achieve a reduction of the simulation run length between 30% and 90%, depending on the model parameters. For details we refer to [20].

4 A Modeling Example: MMPP/G/1/K queueing system

The modeling and evaluation process using TimeNET is now illustrated by an example. An MMPP/G/1/K queueing system with vacations is considered. This kind of queueing systems is very common for the modeling of ATM-switches in Broadband-ISDN. The arrival process is a 2-state Markov modulated Poisson process (MMPP), the service time is generally distributed, and the buffer capacity is restricted to *K* places. Additionally, the server may take repetitive vacations after busy periods. The duration of a vacation is also generally distributed. The MMPP is specified by four parameters: high and low arrival rate λ_1 and λ_2, rate of changing from high to low level r_1 and rate of changing from low to high level r_2.The mean rate λ is thus given by $\lambda = (\lambda_1 r_2 + \lambda_2 r_1)/(r_1 + r_2)$ and the burstiness *b* is defined as the ratio of the high to the mean arrival rate $b = \lambda_1 / \lambda$.

Using TimeNET, the queueing system can easily be modeled as an extended DSPN. Figure 2 shows the user interface. The main window contains the graphical representation of the model. The buttons and icons on the left side allow the user to edit the model interactively.

The pull-down menues on the upper side allow file operations and validation/evaluation of the model.

In this paragraph the SPN model is described. The arrival of data cells is modeled by the exponential transition T1, the modulation of the arrival by the subnet consisting of P1, T2, P2, and T3. T1 has no input place and is therefore enabled in each marking, its mean firing delay depends on whether a token is in P1 or P2. Tokens in P3, P4, and P5 represent cells immediately after arrival, waiting for service, or receiving service, respectively. A token in P6 models an idle server and a token in P7 a server vacation. The number of free buffer places is represented by tokens in P8. Initially all K places are free. A cell arriving at the system may either enter the system (firing of immediate transition t5) if a buffer place is available, otherwise it gets lost (firing of immediate transition t4). A cell waiting for service may enter the service facility if it is idle (firing of immediate transition t6). Transition T7 with a generally distributed firing time represents the service process. When the server facility becomes idle, it may take a vacation (firing of immediate transition t8). A vacation may only be taken if no cell is waiting for service. This is modeled by assigning a higher priority to t6 than to t8 (default priority is 1, therefore priority 2 is assigned to t6). The duration of the vacation itself is modeled by transition T9 with a generally distributed firing time.

TimeNET provides several special purpose editors for the definition of certain parts of the model. These editors allow the specification of probability distribution functions of timed transitions, marking-dependent delays of timed transitions, marking-dependent weights of immediate transitions, marking-dependent arc multiplicities, and result measures. Figure 3 shows the editor for specifying expolynomially distributed delays. The user may specify the probability mass function (pmf) of an expolynomial distribution using expressions in a context-free grammar. The editor then immediately draws the curve of the pmf or the distribution function, as desired. An on-line help is provided for all operations. A special syntax may also be used for the definition of result measures. For example, the mean queue length can be expressed as:

Mean: E{#P4}+E{#P5};

and the probability that less than 10 cells are inside the system as:

Threshold: P{#P8<10}; (5)

The modeling process is thus very efficient, just some minutes were needed for creating the example model description.

Now the model behavior can be validated. The first possibility is to play the tokengame: in that mode all transitions enabled in a marking are blinking, mouse clicking causes firing of a transition leading to a marking change. Second, the minimal place invariants can be computed. In the example the three invariants #P1 + #P2 = 1, #P4 + #P5 + #P8 = *K*, #P5 + #P6 + #P7 = 1 are determined. Additionally, the extended conflict sets (ECS) of immediate transitions can be checked. In the example it turns out that t5 and t8 belong to the same ECS. This can be interpreted as follows: although t5 and t8 are not in direct conflict, the choice whether t5 fires or not may influence whether t6 can fire. Although this situation does not occur in the given model, assigning priority 2 to t5 avoids this unintended semantics. After the assignement all immediate transitions are in different ECS.

Now it is possible to evaluate the model. The following sets of experiments are performed on a DEC Alpha workstation (DEC 3000 Model 800, 1/2 GB main memory). In all curves the measure Threshold as defined in (5) is determined for a varying burstiness *b*. The following parameters of the queueing system are adopted: $K = 50$, $\lambda = 0.5$, $\lambda_2 = 0.4$, $\rho_2 = 1/r_2 = 7312$. λ_1 and $\rho_1 = 1/r_1$ can then be computed for a given value of *b*. The mean service and vacation time is set to 1.

Figure 4 shows analytical results for different distributions of the service and vacation time. The mean of the distribution is kept fixed for all curves. An exponential distribution with rate 1, an uniform distribution with support from 0 to 2, and a deterministic time equal to 1 are used, respectively. The tangible state space contains 202 states and approximately 5 sec of CPU time are required for the computation of a result for a given value of *b*. The figure shows that depending on the distribution the results may differ in several orders of magnitude. This can especially be observed for smaller values of *b*, whereas for higher values the difference becomes less significant.

In another experiment the service and vacation times are assumed to be deterministic. The approximation component is used for obtaining the results shown in Figure 5. The solid line corresponds to the exact analytical results already shown in Figure 4. The dashed lines show the results obtained with an Erlang distribution with two phases, and with a generalized Cox distribution function with two phases and a squared coefficient of variation of 1/100. In both cases the expanded state space contains 404 states, while 6 sec of CPU time are required for the computation of a result for a given value of *b*. The curves show that the generalized Cox distribution leads to a much better approximation, although the computational costs are the same in both cases.

In an additional experiment we compare the results of the analysis and simulation components. Figure 6 shows the curves in case of exponentially distributed and deterministic service and vacation times. The solid lines correspond to analytical results and the dashed lines correspond to results obtained by simulation. The confidence level is set to 95%, the maximum relative error of the confidence intervals to 5%, and sequential simulation is used. Due to the logarithmic scale, the graphical representation of the confidence intervals is contained in the dots of the mean values. The required CPU time for determining a result for one value of *b* ranges from 5 sec (*b* = 2.4) to 50 min (*b* = 1.7). This wide range of execution time demonstrates the sensitivity of simulation techniques to the probability of the event of interest.

The model could easily be modified. For example, a MMPP comprising more states or more complex dependencies between the queue length and the service process could be specified. In the following we consider a tandem queueing system: after receiving service the customers proceed to a second queueing system with deterministic service and vacations. Figure 7 shows a SPN model of the tandem queueing system. Similar steps for the validation and evaluation as for the simple queueing system can be performed. It is interesting to investigate how the burstiness influences the behavior at the second queue. Figure 8 shows the probability that the second queue has less than 10 free buffer places, defined as:

Threshold2: P{#P14 < 10};

The results are obtained by sequential simulation. The required CPU time for determining a result for one value of *b* ranges from 10 sec (*b* = 2.4) to 30 min (*b* = 1.7).

5 Conclusions

TimeNET is a software package which supports the modeling and evaluation of discrete event systems by means of stochastic Petri nets. It provides a user-friendly graphical interface and several specialized components for the evaluation of non-Markovian stochastic Petri nets. Analysis components for SPNs with deterministic, exponentially and generally distributed firing delays, an approximation component for SPNs with deterministic firing delays, and an efficient simulation component for arbitrary SPNs are available. For the structural analysis and the generation of the reachability graph specialized efficient algorithms are used. The general structure and a description of the algorithms has been given. An example illustrating the modeling and evaluation process was also given. The tool has already been successfully employed for several modeling projects.

In future work, several extensions of TimeNET are intended: implementation of an analysis component for SPNs with discrete timing [34], development of a transient analysis component for DSPNs [14], and incorporation of colored tokens. In the application field we plan to model and evaluate telecommunication systems in cooperation with our industrial partner, Siemens AG.

Acknowledgements

The authors would like to thank Ulrich Hertlein, Kolja Koischwitz, Thomas Kuhlmann, Jörg Mitzlaff, and Joachim Schneider for the implementation of the new components of TimeNET.

References

[1] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, A. Cumani. The Effect of Execution Policies on the Semantics of Stochastic Petri Nets. *IEEE Trans. Softw. Engin.,* **15** (1989) 832–846.

[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] G. Balbo, G. Chiola, G. Franceschinis, G. Molinar Roet. On the Efficient Construction of the Tangible Reachabilty Graph of Generalized Stochastic Petri Nets. *Proc. of the 2nd Int. Workshop on Petri Nets and Performance Models*, Madison, WI, USA, pp. 85–92, August 1987.

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

[5] G. Chiola. Compiling Techniques for the Analysis of Stochastic Petri Nets. *Proc. 4th Int. Conf. on Modeling Techniques and Tools for Computer Performance Evaluation*, Palma de Mallorca, Spain, pp. 11–24, 1988.

[6] 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.

[7] G. Chiola, M. Ajmone Marsan, G. Balbo, and G. Conte. Generalized Stochastic Petri Nets: A Definition at the Net Level and Its Implications. *IEEE Trans. Softw. Engineering,* **19** (1993) 89–107.

[8] H. Choi, V. G. Kulkarni, and K. S. Trivedi. Markov Regenerative Stochastic Petri Nets. *Perf. Eval.,* **20** (1994) 337–357.

[9] G. Ciardo, R. German, C. Lindemann. A Characterization of the Stochastic Process Underlying a Stochastic Petri Net. *Trans. on Softw. Eng.,* **20** (1994) 506–515.

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

[11] J. Couvillion, R. Freire, R. Johnson, W.D. Obal, M.A. Qureshi, M. Rai, W.H. Sanders, and J.E. Twedt. Performability Modeling with UltraSAN. *IEEE Software,* **8** (1991) 69–80.

[12] A. Cumani. ESP - A Package for the Evaluation of Stochastic Petri Nets with Phase-Type Distributed Transition Times. *Proc. 1st Int. Workshop Timed Petri Nets*, Torino, Italy, pp. 144–151, 1985.

[13] R. German. A New Approach to the Approximation of Deterministic Time in Continuous

Time Stochastic Models. *Short Papers and Tools Descriptions 7th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, Vienna, Austria, pp. 91–94, 1994.

[14] R. German. Transient Analysis of Deterministic and Stochastic Petri Nets by the Method of Supplementary Variables. *Proc. Int. Workshop Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '95)*, Durham, NC, USA, 1995, (to appear).

[15] R. German, C. Lindemann. Analysis of Stochastic Petri Nets by the Method of Supplementary Variables. *Perf. Eval.*, **20** (1994) 317–335.

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

[17] P. Heidelberger, P.D. Welch. A Spectral Method for Confidence Interval Generation and Run Length Control in Simulations. *Communications of the ACM,* **24** (1981) 233–245.

[18] C. Kelling. A New Method to Determine the Initial Checkpoint of the Spectral Variance Analysis. *Proc. 7th GI/ITG Conf. on Modeling, Measurement and Evaluation of Computing Systems (MMB '93),* Aachen, Germany, pp. 37–42, 1993.

[19] C. Kelling, G. Hommel. Modeling Priority Schemes with Timed Petri Nets. *Proc. 2nd Int. Workshop on Parallel and Distributed Real-Time Systems*, Cancun, Mexico, IEEE Press, 1994, (also to appear in *Int. J. of Mini- and Microcomputers*).

[20] C. Kelling. Control Variate Selection Strategies for Timed Petri Nets. *Proc. of the European Simulation Symposium 1994.* Instanbul, Turkey, pp. 73–77, 1994.

[21] C. Kelling. TimeNET-SIM – a Parallelsimulator for Stochastic Petri Nets. Internal paper (submitted for publication).

[22] D.R. Kincaid, J.R. Respess, and D.M. Young. ITPACK 2C: A Fortran Package for Solving Large Sparse Linear Systems by Adaptive Accelerated Iterative Methods. *ACM Trans. on Math. Softw.*, **8** (1982) 302–322.

[23] G. Klas, R. Lepold. TOMSPIN, a Tool for Modeling with Stochastic Petri Nets. *Proc. of the 6th Annual European Computer Conference*, Le Hague, Netherlands, pp. 618–623, 1992.

[24] C. Lindemann. An Improved Numerical Algorithm for Calculating Steady-State Solutions of Deterministic and Stochastic Petri Net Models. *Perf. Eval.*, **18** (1993) 79–95.

[25] C. Lindemann. DSPNexpress: A Software Package for the Efficient Solution of Deterministic and Stochastic Petri Nets. *Perf. Eval.*

[26] C. Lindemann, G. Ciardo, R. German, and G. Hommel. Performability Modeling of an Automated Manufacturing System with Deterministic and Stochastic Petri Nets. *Proc. IEEE Int. Conf. on Robotics and Automation,* Atlanta, Georgia, USA, pp. 576–581, 1993.

[27] C. Lindemann, R. German. Modeling Discrete Event Systems with State-Dependent Deterministic Service Times. *Discrete Event Dynamical Systems: Theory and Applications,* **3** (1993) 249-270.

[28] C. Lindemann, F. Schön. Performance Evaluation of Memory Consistency Models for

Multiprocessor Systems with Vitually Shared Memory. *Proc. 26th Hawaii Int. Conf. on System Sciences*, Maui, Hawaii, 1993.

[29] C. Lindemann, A. Zimmermann. An Adaptive Algorithm for the Efficient Generation of the Tangible Reachability Graph of a Stochastic Petri Net. *Technical Report 1994-8*, Technische Universität Berlin, Germany, 1994.

[30] J. Martinez, M. Silva. A Simple and Fast Algorithm to Obtain All Invariants of a Generalized Petri Net. In: *C. Girault, W. Reisig, (Eds.) Informatik Fachberichte 52*, pp. 301-310, Springer 1982.

[31] K. Pawlikowski. Steady-State Simulation of Queueing Processes: A Survey of Problems and Solutions. *ACM Computing Surveys,* **22** (1990) 123–170.

[32] L.W. Schruben, H. Singh, L. Tierney. Optimal tests for initialization bias in simulation output. *Operations Research,* **31** (1983) 1167–1178.

[33] A.H. Sherman. Algorithms for sparse Gaussian elimination with partial pivoting. *ACM Trans. on Math. Softw.,* **4** (1978) 330–338.

[34] R. Zijal, R. German. A new Approach to Discrete Time Stochastic Petri Nets. *Proc. 11th Int. Conf. on Analysis and Optimization of Systems*, Sophia-Antipolis, France, pp. 198–204, 1994.

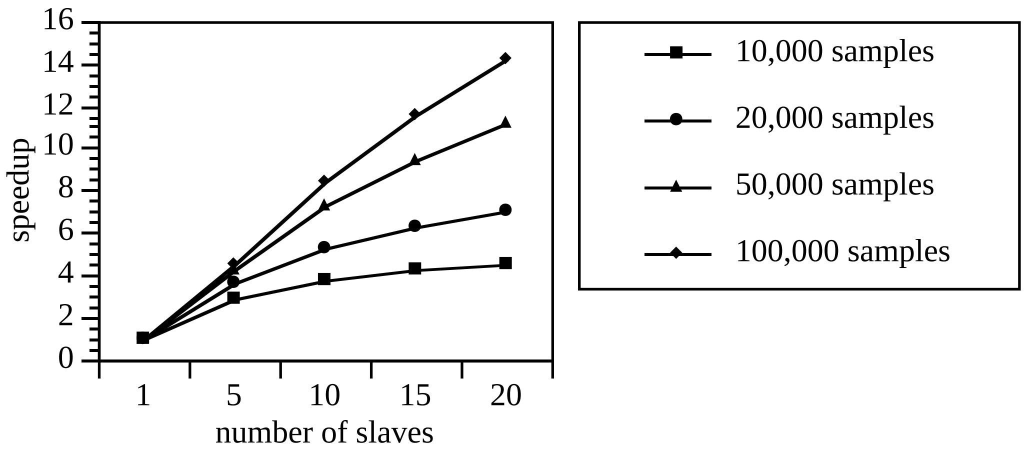
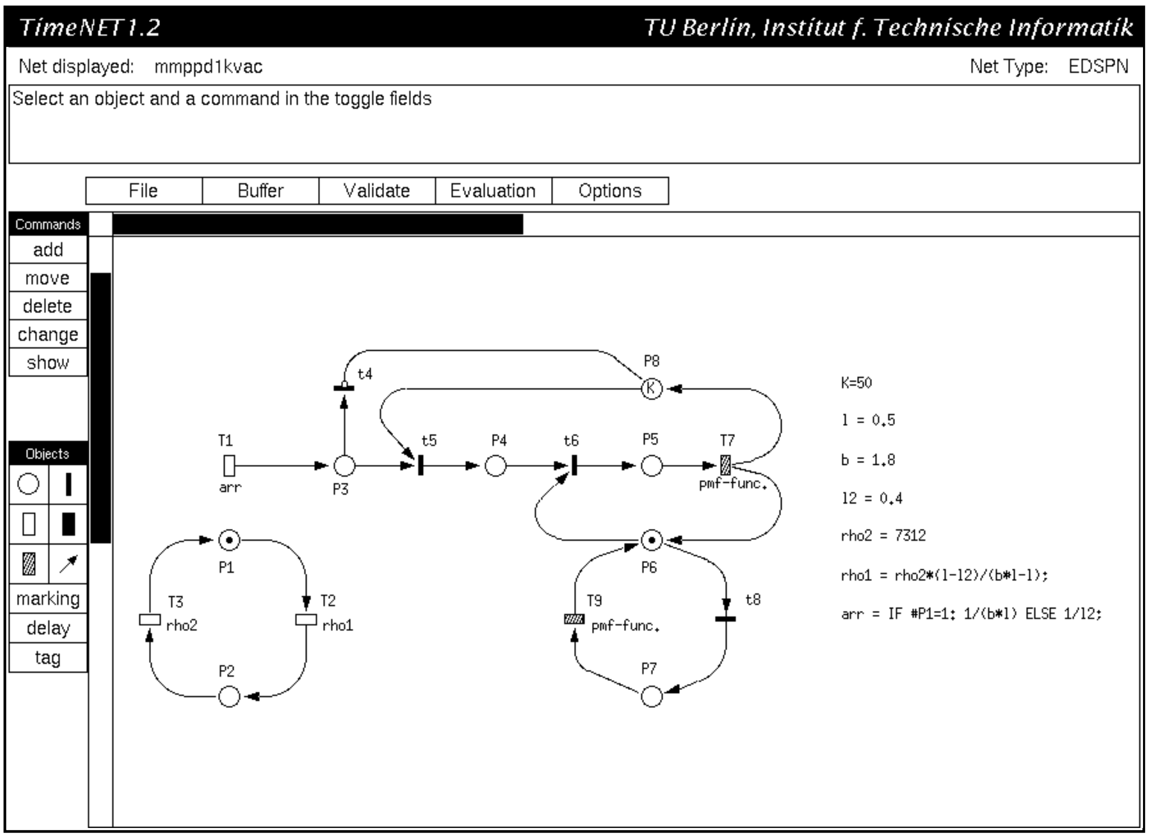


Figure 1. Gain of Parallelization (Simulation)



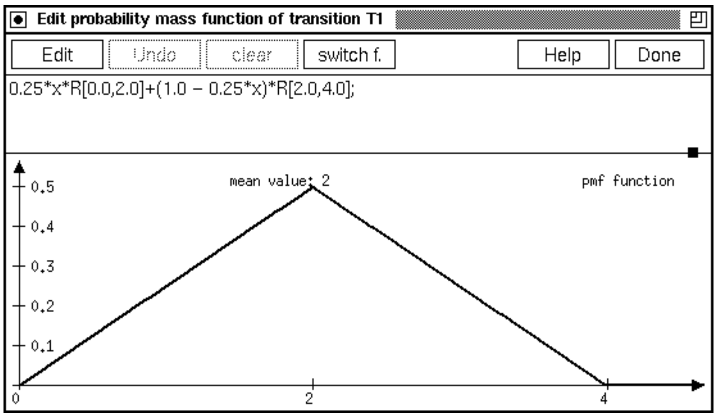
Net displayed: mmppd1kvac

Net Type: EDSPN

Select an object and a command in the toggle fields

```
K=50

l = 0.5

b = 1.8

l2 = 0.4

rho2 = 7312

rho1 = rho2*(l-l2)/(b*l-l);

arr = IF #P1=1: 1/(b*l) ELSE 1/l2;
```

Figure 2. User interface of TimeNET



0.25*x*R[0.0,2.0]+(1.0 - 0.25*x)*R[2.0,4.0];

Figure 3. Editor for specifying expolynomial pmfs

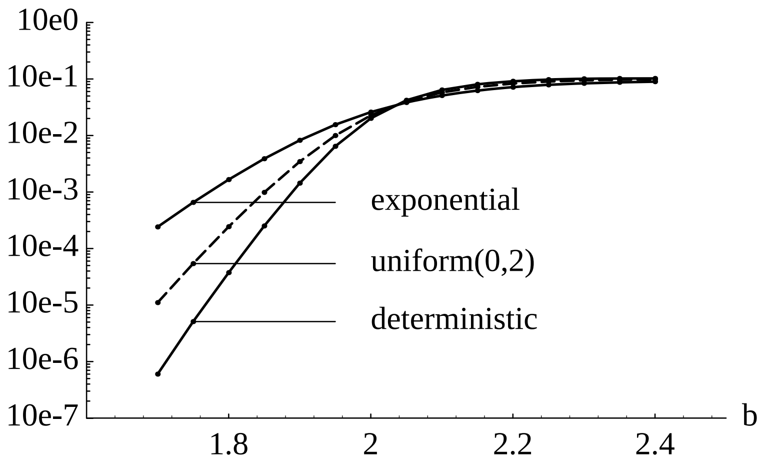


Figure 4. Threshold vs. burstiness for different distributions (analysis)

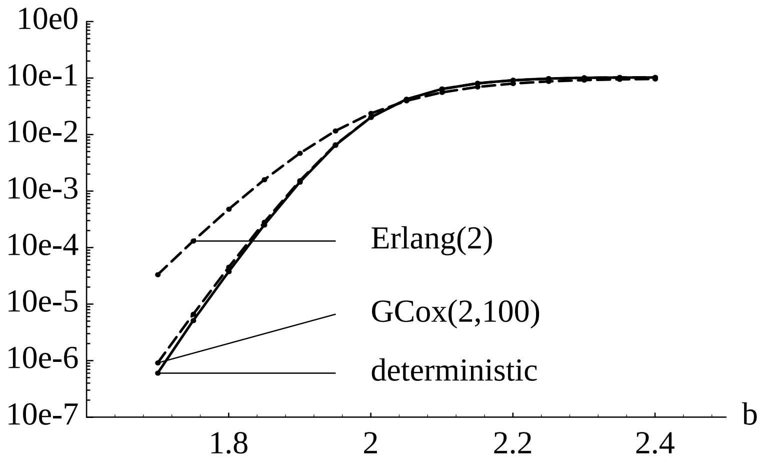


Figure 5. Threshold vs. burstiness (analysis and approximation)

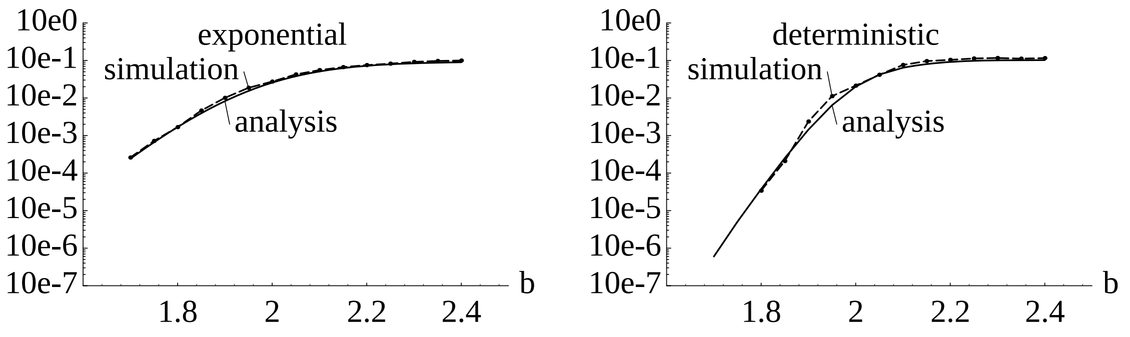


Figure 6. Threshold vs. burstiness (analysis and simulation)

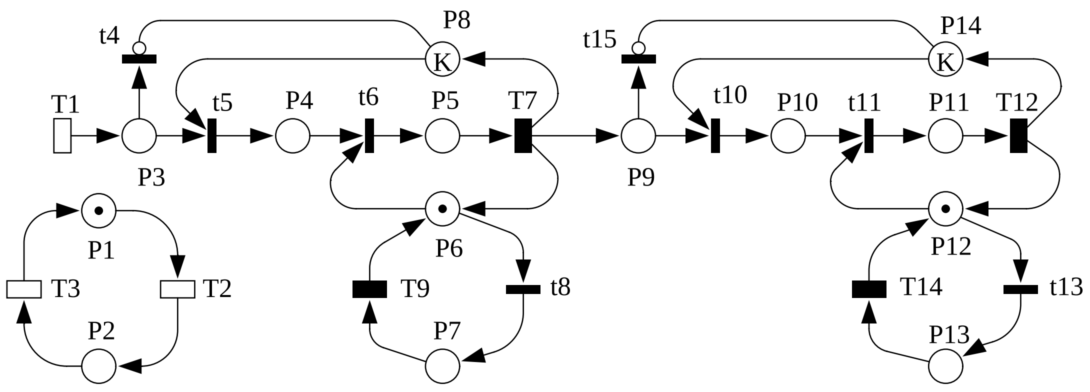


Figure 7. Tandem queueing system

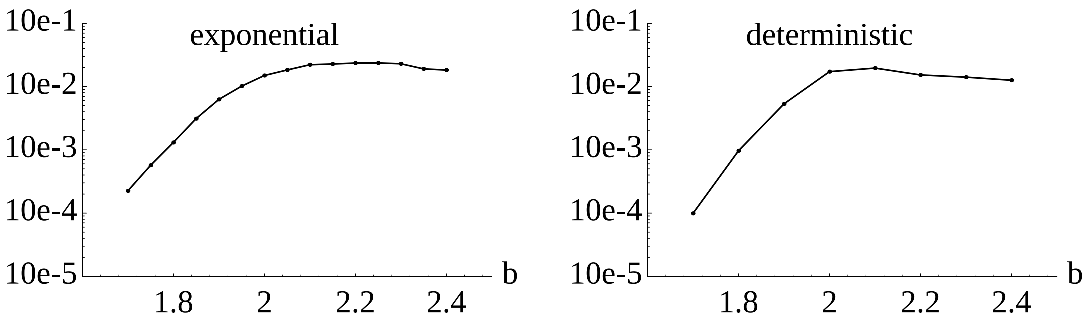


Figure 8. Threshold2 vs. burstiness (simulation)