Non-Exponential Stochastic Petri Nets: an Overview of Methods and Techniques *

Andrea Bobbio
Dipartimento di Informatica
Università di Torino, 10149 Torino, Italy

Miklós Telek
Department of Telecommunications
Technical University of Budapest, 1521 Budapest, Hungary

Abstract

The analysis of stochastic systems with non-exponential timing requires the development of suitable modeling tools. Recently, some effort has been devoted to generalize the concept of Stochastic Petri nets, by allowing the firing times to be generally distributed. The evolution of the *PN* in time becomes a stochastic process, for which in general, no analytical solution is available. The paper surveys suitable restrictions of the *PN* model with generally distributed transition times, that have appeared in the literature, and compares these models from the point of view of the modeling power and the numerical complexity.

Key words: Stochastic Petri Nets, Non-exponential Distributions, Phase-type Distributions, Markov and Semimarkov Reward Models, Markov Regenerative Processes, Queueing Systems with Preemption.

1 Introduction

The usual definition of Stochastic Petri Net *(SPN)* implies that all the timed activities associated to the transitions are represented by exponential random variables, so that the evolution of the net in time is mapped into a continuous time Markov chain *(CTMC)*. There are, however, practical situations that are not covered by these models. In fact, many activities in computer, communication and manufacturing systems are more likely represented by random variables with low variability (or even deterministic). Moreover, some significant measures, introduced to characterize a stochastic system over an interval rather then at a time instant (like the distribution function of cumulative measures),

*A preliminary version of this paper has been presented at the First European Conference on Dependable Computing EDCC-1, Berlin, October 1994 [9]

cannot be evaluated by solving a set of linear first order differential equations typical of Markovian systems [40, 6] and require a more complex stochastic formulation.

In recent years, several classes of (*SPN)* models have been elaborated which incorporate some non-exponential characteristics in their definition. The semantics of *SPN*'s with generally distributed transition times has been discussed in [1]. We refer to this model as *Generally Distributed Transition_SPN (GDT_SPN).* In order to properly define a marking process associated to a *GDT_SPN*, each timed transition should be assigned a memory policy chosen among three possible alternatives: *resampling, enabling* and *age memory.* The *resampling policy* is mapped into a semi-markov marking process. The *enabling memory* policy is suited to realize an execution mechanism that in queueing theory is called *preemptive repeat different (prd)* policy. Whenever the task in service is preempted its service time is reset and the execution restarts from scratch. On the other hand, the *age memory* policy is suited to represent an execution mechanism usually referred to as *preemptive resume (prs)* policy: when a task is enabled again after preemption, its execution restarts from the point it was interrupted. In general, the stochastic process underlying a *GDT_SPN* does not have a tractable analytical formulation, while a simulative solution has been investigated in [29].

With the aim of providing a *modeler's representation* able to automatically generate an *analytical representation*, various restrictions of the general *GDT_SPN* model have been discussed in the literature [16, 9].

The semimarkov *SPN*, studied in [38, 5], seems of little practical interest since the firing of any transition forces a resetting of all the other transitions. A semimarkov *SPN* more suited for applications has been discussed in [23]. In this definition, the transitions are partitioned into three classes: exclusive, competitive and concurrent. Only exclusive or competitive transitions are allowed to be non-exponential.

Cumani [22] has realized a package in which each *PN*-transition can be assigned a *PH* [39] distributed firing time. We refer to this model as *PHSPN.* The peculiar feature of the *PHSPN* model is that it can support any combination of the memory policies defined in [1] and that the related analytical solution can be completely automatized.

A particular case of non-Markovian *SPN*, is the class of *Deterministic and SPN (DSPN).* A *DSPN* is defined in [3] as a Markovian *SPN* where, in each marking, a single transition is allowed to have associated a deterministic firing time. Only the steady state solution was provided in [3]. An improved steady state algorithm was presented in [35], and some structural extensions were investigated in [17]. Choi et al. [14] have observed that the marking process underlying a *DSPN* is a *Markov Regenerative Process (MRP)* for which a closed form expression for the transition probability matrix can be derived both as a function of the time and in steady state [19].

This observation has opened a very fertile line of research aimed at the definition of solvable classes of *SPN* models whose underlying marking process is a *MRP*, and therefore referred to as *Markov Regenerative SPN (MRSPN).* Choi et al. [15] have extended the *DSPN* model by allowing the presence in each marking of at most one transition with a generally distributed firing time. The solution proposed in [15] is based on the derivation of the time-dependent transition probability matrix in the Laplace transform domain,

followed by a numerical inversion. German and Lindemann [27] have proposed to derive the steady state solution of the same model by resorting to the method of supplementary variables [21]. The possibility of applying the methodology of supplementary variables to the transient analysis of *DSPN*'s is explored in [25], where, however, only very special cases are taken into consideration.

The main limitation of the models discussed in the mentioned references is that the generally distributed (or deterministic) transitions must be assigned a firing policy of enabling memory type [1].

A semantic generalization of the *DSPN* model, by including the possibility of modeling preemptive mechanisms of resume type has been proposed in [11]. A *prs* service policy means that the server is able to recover an interrupted job by keeping memory of the work already performed so that, upon restart, only the residual service needs to be completed. This modeling extension is crucial in connection with fault tolerant and parallel computing systems, where a single task may be interrupted either during a fault/recovery cycle or for the execution of a higher priority task, but when the cause originating the interruption is ceased, the dormant task is resumed from the point it was interrupted. Finally, a more general class of solvable *MRSPN* is defined in [10], where both enabling and age memory policies can be combined into a single model.

The aim of this paper is to compare the available *GDT_SPN* models recently appeared in the literature from two distinct and conflicting points of view: the modelling power and the analytical tractability. To this end, the main features of the different formulations are briefly described with the intent of stressing the basic assumptions and the complexity of the related analytical solution. A final example, based on the transient analysis of a closed queuing system with deterministic service time and various kinds of preemptive service policies, is developed in length in order to put in evidence the limits and the potentialities of the different approaches.

The *GDT_SPN* is formally defined in Section 2. Section 3 focuses on two main realizations of the *GDT_SPN*, namely: the *PHSPN* implemented by Cumani in [22] and the *DSPN* introduced by Ajmone and Chiola in [3] and further generalized by different authors [15, 16, 10] into the *MRSPN* model. A comparative discussion of the modeling power of the considered models is performed in Section 4 through an example. Starting from a simple queuing system, more complex modeling assumptions are introduced in order to show how the considered models react to the added structures. The algorithmic complexity of the numerical solutions is briefly addressed in Section 5.

2 Generally Distributed Transition_SPN

A marked Petri Net (*PN*) is a tuple $PN = (P, T, I, O, H, M)$, where:

- $P = \{p_1, p_2, \ldots, p_{np}\}$ is the set of places (drawn as circles);

[1] The enabling memory assumption is relaxed in [17] where a deterministic transition can be disabled in vanishing markings only. Since vanishing markings are transversed in zero time, this assumption does not modify the behavior of the marking process versus time

- $T = \{t_1, t_2, \ldots, t_{nt}\}$ is the set of transitions (drawn as bars);
- I, O and H are the input, the output and the inhibitor functions, respectively. The input function I provides the multiplicities of the input arcs from places to transitions; the output function O provides the multiplicities of the output arcs from transitions to places; the inhibitor function H provides the multiplicity of the inhibitor arcs from places to transitions.
- $M = \{m_1, m_2, \ldots, m_{np}\}$ is the marking. The generic entry m_i is the number of tokens (drawn as black dots) in place p_i, in marking M.

Input and output arcs have an arrowhead on their destination, inhibitor arcs have a small circle. A transition is enabled in a marking if each of its ordinary input places contains at least as many tokens as the multiplicity of the input function I and each of its inhibitor input places contains fewer tokens than the multiplicity of the inhibitor function H. An enabled transition fires by removing as many tokens as the multiplicity of the input function I from each ordinary input place, and adding as many tokens as the multiplicity of the output function O to each output place. The number of tokens in an inhibitor input place is not affected.

A marking M' is said to be *immediately reachable* from M, when it is generated from M by firing a single enabled transition t_k. The reachability set $\mathcal{R}(M_0)$ is the set of all the markings that can be generated from an initial marking M_0 by repeated application of the above rules. If the set T comprises both timed and immediate transitions, $\mathcal{R}(M_0)$ is partitioned into tangible (no immediate transitions are enabled) and vanishing markings, according to [2].

A timed execution sequence $\mathcal{T}_E$ is a connected path in the reachability graph $\mathcal{R}(M_0)$ augmented by a non-decreasing sequence of real non-negative values representing the epochs of firing of each transition, such that consecutive transition firings correspond to ordered epochs $\tau_i \leq \tau_{i+1}$ in $\mathcal{T}_E$.

$$\mathcal{T}_E = \{ (\tau_0, M_{(0)}) ; (\tau_1, M_{(1)}) ; \ldots ; (\tau_i, M_{(i)}) ; \ldots \} \tag{1}$$

The time interval $\tau_{i+1} - \tau_i$ between consecutive epochs represents the period of time that the *PN* sojourns in marking $M_{(i)}$.

A variety of timing mechanisms have been proposed in the literature. The distinguishing features of the timing mechanisms are whether the duration of the events is modeled by deterministic variables or random variables, and whether the time is associated to the *PN* places, transitions or tokens. If a probability measure is assigned to the duration of the events represented by a transition, a timed execution sequence $\mathcal{T}_E$ is mapped into a stochastic process $\mathcal{M}(x)$, $(x \geq 0)$, called the *Marking Process*. *PN*s in which the timing mechanism is stochastic are referred to as Stochastic *PN* (*SPN*).

A *SPN* with stochastic timing associated to the *PN* transitions and with generally distributed firing times *(GDT_SPN)* was defined in [1], with particular emphasis on the semantic interpretation of the model.

Definition 1. *A GDT_SPN is a marked SPN in which:*

- *To any timed transition $t_k \in T$ is associated a random variable γ_k modeling the time needed by the activity represented by t_k to complete, when considered in isolation.*
- *Each random variable γ_k is characterized by its (possibly marking dependent) cumulative distribution function $G_k(w)$.*
- *A set of specifications are given for univocally defining the stochastic process associated to the ensemble of all the timed execution sequences $\mathcal{T}_E$. This set of specifications is called the* execution policy.
- *A initial probability is given on the reachability set.*

An *execution policy* is a set of specifications for univocally defining the stochastic process underlying the *GDT_SPN*, given the *PN* topology and the set of Cdf's associated to each timed transition. Indeed, the inclusion of non-exponential timings destroys the memoryless property and forces to specify how the system is conditioned upon the past history. The *execution policy* comprises two specifications: a criterion to choose the next timed transition to fire (the *firing policy*), and a criterion to keep memory of the past history of the process (the *memory policy*). A natural choice to select the next timed transition to fire is according to a *race policy*: if more than one timed transition is enabled in a given marking, the transition fires whose associated random delay is statistically the minimum. The *memory policy* defines how the process is conditioned upon the past. In *GDT_SPN*, the memory is represented by an *age variable* a_k, associated to each timed transition t_k, that increases with the time in which the corresponding transition is enabled. The way in which a_k is related to the past history determines the different memory policies. Three alternatives are considered:

- *Resampling* - The age variable a_k is reset to zero at any change of marking. The firing distribution depends only on the time elapsed in the current marking.
- *Enabling memory* - The age variable a_k accounts for the time elapsed from the last epoch in which t_k has been enabled. The firing distribution depends on the residual time needed for the transition to complete given a_k. When transition t_k is disabled (even without firing) a_k is reset.
- *Age memory* - The age variable a_k accounts for the total time in which t_k has been enabled from its last firing. The firing distribution depends on the residual time needed for the transition to complete given a_k.

At the entrance in a new tangible marking, the residual firing time is computed for each enabled timed transition given its age variable. The next marking is determined by the minimal residual firing time among the enabled timed transitions (*race policy*). Under an enabling memory policy the firing time of a transition is resampled from the original distribution each time the transition becomes enabled so that the time eventually spent without firing in prior enabling periods is lost. The memory of the underlying stochastic process cannot extend beyond a single cycle of enable/disable of the corresponding transition. On the contrary, if a transition is assigned an age memory policy, the age variable accounts for all the periods of time in which the transition has been enabled, independently of the number of enable/disable cycles. Hence, the age

memory is the only policy that allows a transition to have a non-null memory also in markings in which is not enabled.

Since exponential transitions do not have memory, the residual life time distribution is independent of the value of the age variable. Hence, the three policies have the same effect and we can conventionally assume that the age variables associated to exponential transitions are identically zero.

3 Computational Restrictions of the GDT_SPN

The marking process $\mathcal{M}(x)$ does not have, in general, an analytically tractable formulation, while a simulative approach has been described in [29, 30]. Various restrictions of the general model have been discussed in the literature [16, 9] such that the underlying marking process $\mathcal{M}(x)$ is confined to belong to a known class of analytically tractable stochastic processes.

3.1 Exponentially Distributed SPN

When all the random variables γ_k associated to the *PN* transitions are exponentially distributed, the dynamic behavior of the net is mapped into a *CTMC*, with state space isomorphic to the tangible subset of the reachability graph. This restriction is the most popular in the literature [37, 24, 2], and a number of tools are built on this assumption [13, 18, 20, 34].

3.2 Semi-Markov SPN

When all the *PN* transitions are assigned a resampling policy the marking process becomes a semi-Markov process. This restriction has been studied in [38, 5] but is of little interest in applications where it is difficult to imagine a situation where the firing of any transition of the *PN* has the effect of forcing a resampling to all the other transitions.

A more consistent and interesting semi-Markov *SPN* model has been discussed in [23]. In this definition, the transitions are partitioned into three classes: exclusive, competitive and concurrent. Provided that the firing time of all the concurrent transitions is exponentially distributed and that non-exponential competitive transitions are resampled at the time the transition is enabled, the associated marking process becomes a semi-Markov process.

3.3 Phase Type SPN (PHSPN)

A numerically tractable realization of the *GDT_SPN*, is obtained by restricting the random firing times γ_k to be *PH* distributed [39], according to the following:

Definition 2. *A PHSPN is a GDT_SPN in which:*

- *To any timed transition $t_k \in T$ is associated a PH random variable γ_k. The PH model assigned to γ_k has ν_k stages with a single initial stage numbered stage 1 and a single final stage numbered stage ν_k.*
- *To any timed transition $t_k \in T$ is assigned a memory policy among the three defined alternatives: resampling, enabling or age memory.*

The distinguishing feature of this model, is that it is possible to design a completely automated tool that responds to the requirements stated in [31], and that, at the same time, includes all the issues listed in Definition 2. The non-Markovian process generated by the *GDT_SPN* over the reachability set $\mathcal{R}(M_0)$ is converted into a *CTMC* defined over an expanded state space. The measures pertinent to the original process are defined at the *PN* level and can be evaluated by solving the expanded *CTMC*.

The program package *ESP* [22] realizes the *PHSPN* model according to Definition 2. The program allows the user to assign a *PH* distribution and a specific memory policy to each *PN* transition so that the different execution policies can be put to work. In the *ESP* tool, the algorithm for the generation of the expanded *CTMC* starts from the knowledge of the reachability graph $\mathcal{R}(M_0)$ and is driven by the different *PH* models and execution policies assigned to each transition.

The expanded *CTMC* is represented by a directed graph $H = (N_H, A_H)$ where N_H is the set of nodes (states of the expanded *CTMC*) and A_H is the set of directed arcs (transitions of the expanded *CTMC*). The nodes in N_H are pairs (M, W), where $M \in \mathcal{R}(M_0)$ is a marking of the original non-expanded *SPN* and W is a n_t-dimensional vector of integers, whose kth entry w_k $(1 \leq w_k \leq \nu_k)$ represents the stage of firing of t_k in its *PH* distribution.

Arcs in A_H are represented by 5-tuples $(N, N'; k, i, j)$, where N is the source node, N' the destination node, and (i, j) is an arc in the *PH* model of transition t_k. Therefore, $(N, N'; k, i, j) \in A_H$ means that in the expanded graph the process goes from node N to node N' when the stage of firing of t_k goes from stage i to stage j.

The expanded graph H is generated by an iterative algorithm [22]. Let H be initially empty; the algorithm starts by putting in N_H the initial node $N_H^{(1)} = (M_0, [1, 1, \ldots, 1])$ (the *PN* is in its initial marking M_0 and all the n_t random variables γ_k are in stage 1). $N_H^{(1)}$ is marked as a non-expanded node. An expansion step is then performed on each non-expanded node $N_H^{(\ell)} = (M^{(\ell)}, W^{(\ell)})$. For each transition $t_k^{(\ell)}$ enabled in $M^{(\ell)}$, we search for all the possible successors of stage numbered $w_k^{(\ell)}$ in the *PH* model of $t_k^{(\ell)}$ (where $w_k^{(\ell)}$ is the kth component of $W^{(\ell)}$, i.e. it is the stage of γ_k in state $N_H^{(\ell)}$). Let j $(j = 1, \ldots, \nu_k)$ be one of such possible successors. Two cases may arise depending whether $j = \nu_k$ or $j \neq \nu_k$; i.e. transition $t_k^{(\ell)}$ has reached its terminal stage, or not.

<u>*CASE 1*</u> - $j \neq \nu_k$

Transition $t_k^{(\ell)}$ has made a jump in its *PH* model without firing. Then a new node $N_H'^{(\ell)} = (M^{(\ell)}, W'^{(\ell)})$ is generated, with $w_k'^{(\ell)} = j$ and $w_l'^{(\ell)} = w_l^{(\ell)}$ $(\forall\, t_l \in T\,;\; t_l \neq t_k)$.

<u>*CASE 2*</u> - $j = \nu_k$

Transition $t_k^{(\ell)}$ has reached the final node of its *PH* model and thus has fired. In this

case, the new node $N_H'^{(\ell)} = (M'^{(\ell)}, W'^{(\ell)})$ is generated according to the following rule: $M'^{(\ell)}$ is the marking immediately reached from $M^{(\ell)}$ by firing $t_k^{(\ell)}$ ($M^{(\ell)} - t_k^{(\ell)} \rightarrow M'^{(\ell)}$), and $w_k'^{(\ell)} = 1$ since firing of $t_k^{(\ell)}$ resets its stage count to 1. The values of the other entries of vector $W'^{(\ell)}$, corresponding to the transitions enabled in $M^{(\ell)}$ are set according to the memory policy attached to the corresponding transition:

- *always set equal to 1 in the resampling case;*
- *not modified in the age memory case;*
- *conditionally reset in the enabling memory case (i.e. if the transition is still enabled in the new marking $M'^{(\ell)}$ the corresponding stage count is not modified otherwise is set to 1).*

In both cases, the new node $N_H'^{(\ell)} = (M'^{(\ell)}, W'^{(\ell)})$ is entered in N_H (if not already there) and a new arc $A_H'^{(\ell)} = (N_H^{(\ell)}, N_H'^{(\ell)};\ k, w_k^{(\ell)}, w_k'^{(\ell)})$ is added to A_H. The above expansion step is iterated for all the transitions enabled in the current marking $M^{(\ell)}$, and until all the corresponding *PH* distributions have reached their terminal stage. At this point the expansion of the node $N_H^{(\ell)}$ is terminated and the node itself is marked as expanded. The algorithm then searches for the subsequent non-expanded node until all the nodes have been searched for.

The cardinality n_H of the expanded state space is upper bounded by the cross product of the cardinality of the reachability set of the basic *PN* times the cardinality of the *PH* distributions of the n_t random variables γ_k [1]. The actual value of n_H is difficult to evaluate a priori. In practical cases this number can be very much lower that the upper bound, and is, in any case, increasing with the complexity of the assigned memory policies. The resampling policy is the one that generates the expanded *CTMC* with the lower number of states, while the age policy generates the expanded *CTMC* with the larger number of states.

The marking $M^{(\ell)}$ of the original reachability set, is mapped into a macro state formed by the union of all the nodes $N_H(M, W)$ of the expanded graph such that $M = M^{(\ell)}$. This mapping allows the program to redefine the measures calculated as solution of the Markov equation over the expanded graph in terms of the markings of the original *PN*.

An alternative approach for the inclusion of *PH* distributions into *SPN* models consists in substituting each transition, at the *PN* level, with a proper sub-*PN* realizing the required *PH* model. However, a naive application of this approach, as proposed in [37], can not correctly accommodate the different memory policies. An attempt to realize a sub-*PN* able to account for all the three memory policies is in [12] (but restricted to *PH* distributions with equal diagonal elements). Nevertheless, the expansion at the *PN* level has been strongly discouraged in [1] on the basis of the following motivations:

- The inclusion of a sub-*PN* for each transition makes the expanded *PN* very intricate and difficult to understand. The added primitive elements (places, transitions and arcs) refer only to the stochastic behavior of a single transition and hide the general structure of the model. The fascinating simplicity of the *PN* language is destroyed.
- It seems hardly possible to automatize a procedure for generating the *PHSPN* model expanding the basic *PN* and taking into account general *PH* distributions and interactions among different memory policies.

3.4 Deterministic SPN

The *Deterministic and Stochastic PN* model has been introduced in [3], with the aim of providing a technique for considering stochastic systems in which some time variables assume a constant value. In [3] only the steady state solution has been addressed. An improved algorithm for the evaluation of the steady state probabilities has been successively presented in [35], and some structural extensions have been proposed in [17].

Definition 3 - *A DSPN [3] is a GDT_SPN in which:*

- *The set T of transitions is partitioned into a subset T_e of exponential transitions (EXP) and a subset T_d of deterministic transitions (DET), such that $T = T_e \cup T_d$.*
- *To any EXP transition $t_k \in T_e$ is associated an exponentially distributed random variable γ_k.*
- *To any DET transition $t_j \in T_d$ is associated a deterministic firing time d_j.*
- *At most, a single DET transition is allowed to be enabled in each marking.*
- *The only allowed execution policy for the DET transition is the* race *policy with* enabling memory.

According to Definition 3, during the firing of a DET transition, the marking process can undergo EXP transitions only, thus describing a *CTMC* called the subordinated process. The steady state solution technique, originally proposed in [3], is based on the evaluation of the subordinated *CTMC* at a time corresponding to the duration of the DET transition. Various improvements and extensions of the original algorithm are in [35, 17]. Tools currently supporting steady state measures for *DSPN* models are DSPNexpress [36], UltraSAN [20] and TimeNET [26].

Choi et al. [14] have shown that the marking process associated to a *DSPN* is a Markov regenerative process (*MRP*), for which steady state and transient solution equations are available [19]. In order to prove their assertion, Choi et al. have introduced the following modified execution sequence:

$$\mathcal{T}_E = \{ (\tau_0^*, M_{(0)}) ; (\tau_1^*, M_{(1)}) ; \ldots ; (\tau_i^*, M_{(i)}) ; \ldots \} \tag{2}$$

Epoch τ_{i+1}^* is derived from τ_i^* as follows:

1. If no DET transition is enabled in marking $M_{(i)}$, define τ_{i+1}^* to be the first time after τ_i^* that a state change occurs.
2. If a DET transition is enabled in marking $M_{(i)}$, define τ_{i+1}^* to be the time when the DET transition fires or is disabled as a consequence of the firing of a competitive EXP transition.

According to case 2) of the above definition, during $[\tau_i^*, \tau_{i+1}^*)$, the *PN* can evolve in the subset of $\mathcal{R}(M_0)$ reachable from $M_{(i)}$, through EXP transitions concurrent with the given DET transition. The marking process during this time interval is the *CTMC* subordinated to marking $M_{(i)}$. Therefore, if a DET transition is enabled in $M_{(i)}$, the sojourn

time is given by the minimum between the first passage time out of the subordinated *CTMC* and the constant firing time associated to the DET transition.

Choi et al. show that the sequence τ_i^* forms a sequence of regenerative time points, so that the marking process $\mathcal{M}(x)$ is a Markov regenerative process *MRP*. According to [15, 19], we define the following matrix valued functions:

$$\begin{array}{rclcrcl} \mathbf{V}(x) & = & [V_{ij}(x)] & \text{such that} & V_{ij}(x) & = & Pr\{\mathcal{M}(x) = j \,|\, \mathcal{M}(0) = i\} \\ \mathbf{K}(x) & = & [K_{ij}(x)] & " & K_{ij}(x) & = & Pr\{M_{(1)} = j \,,\, \tau_1^* \leq x | \, \mathcal{M}(0) = i\} \\ \mathbf{E}(x) & = & [E_{ij}(x)] & " & E_{ij}(x) & = & Pr\{\mathcal{M}(x) = j \,,\, \tau_1^* > x | \, \mathcal{M}(0) = i\} \end{array} \quad (3)$$

Matrix $\mathbf{V}(x)$ is the transition probability matrix and provides the probability that the marking process $\mathcal{M}(x)$ is in marking j at time x given it was in i at $x = 0$. The matrix $\mathbf{K}(x)$ is the *global kernel* of the *MRP* and provides the probability that the regeneration interval ends in marking j at time x, given that it started in marking i at $x = 0$. Finally, the matrix $\mathbf{E}(x)$ is the *local kernel* and describes the behavior of the marking process inside two consecutive regeneration time points. The transient behavior of the *DSPN* can be evaluated by solving the following generalized Markov renewal equation (in matrix form) [19, 15]:

$$\mathbf{V}(x) \,=\, \mathbf{E}(x) \,+\, \mathbf{K} \,*\, \mathbf{V}(x) \quad (4)$$

where $\mathbf{K} * \mathbf{V}(x)$ is a convolution matrix, whose (i, j)-th entry is:

$$[\mathbf{K} \,*\, \mathbf{V}(x)]_{ij} \,=\, \sum_k \int_0^x dK_{ik}(y)\, V_{kj}(x - y) \quad (5)$$

Equation (4) can be solved numerically in the time domain. An alternative approach, suggested in [14], consists in transforming the convolution equation (4) in the Laplace domain. By denoting the Laplace Stieltjes transform *(LST)* of a function $F(x)$ by $F^{\sim}(s) = \int_0^{\infty} e^{-sx}\, dF(x)$, Equation (4) becomes:

$$\mathbf{V}^{\sim}(s) \,=\, \mathbf{E}^{\sim}(s) \,+\, \mathbf{K}^{\sim}(s)\; \mathbf{V}^{\sim}(s) \quad (6)$$

whose solution is:

$$\mathbf{V}^{\sim}(s) \,=\, [\mathbf{I} \,-\, \mathbf{K}^{\sim}(s)]^{-1}\; \mathbf{E}^{\sim}(s) \quad (7)$$

The time domain solution of (7) can be obtained by numerical inversion [32].

3.5 Markov Regenerative SPN (MRSPN)

A natural extension of the *DSPN* has been proposed in [15], where the DET transition in Definition 3 is replaced by a GEN transition. This model is referred to by the authors as *MRSPN**.

Definition 4. A *MRSPN* is a GDT_SPN in which:*

- *The set T of transitions is partitioned into a subset T_e of exponential transitions (EXP) and a subset T_g of generally distributed transitions (GEN), such that $T = T_e \cup T_g$.*
- *To any EXP transition $t_k \in T_e$ is associated an exponentially distributed random variable γ_k.*
- *To any GEN transition $t_j \in T_d$ is associated a generally distributed random variable γ_j.*
- *At most, a single GEN is allowed to be enabled in each marking.*
- *The only allowed execution policy for the GEN transition is the* race *policy with* enabling memory.

By Definition 4, during the firing of a GEN transition only EXP transitions can concurrently fire: the process subordinated to a GEN transition is a *CTMC*. The matrices $\mathbf{K}(x)$ and $\mathbf{E}(x)$ in (3) depend on the specific Cdf's assumed in the model. In [15], closed form expressions are derived when the Cdf of the GEN transitions is the uniform distribution.

3.6 MRSPN with non overlapping activity cycles

With the aim of extending the modeling power of *MRSPN*s by including GEN transitions with age memory policy, Bobbio and Telek [11, 10] have investigated a class of models characterized by the fact that the subordinated process between two consecutive regeneration time points is a Semimarkov Reward Process [40].

Since in a *GDT_SPN* the memory of the marking process is related to the positive value assumed by the age variables attached to the GEN transitions (EXP transitions do not carry memory), a regeneration time point occurs at the entrance in a marking in which all the age variables are reset.

Definition 5. *A a Markov Regenerative Stochastic Petri Nets (MRSPN) [11] is a GDT_SPN, for which an embedded Markov renewal sequence $(\tau_n^*, M_{(n)})$ exists such that at the epoch τ_n^* of entrance in the tangible marking $M_{(n)}$ all the age variables are equal to 0.*

In order to restrict Definition 5 to a class of solvable models, the concept of *MRSPN* with non overlapping activity cycles has been introduced in [10]. This new class encompasses and generalizes all the models previously appeared in the literature and mentioned in the previous sections.

Definition 6 - *A GEN transition is dormant in those markings in which the corresponding age variable is equal to zero and is active in those markings in which the age variable is greater than zero. The* activity cycle *of a GEN transition is the period of time in which a transition is active between two dormant periods.*

Let t_g be a GEN transition. The activity cycle of t_g is influenced by its memory policy, and can be characterized in the following way:

- *Resampling Memory* - If t_g is a resampling memory transition, its activity cycle starts as soon as t_g becomes enabled, and ends at the first subsequent firing of any transition (including t_g itself). During the activity cycle of a resampling memory transition no change of marking is possible.
- *Enabling Memory* - If t_g is an enabling memory transition its activity cycle starts as soon as t_g becomes enabled when dormant, and ends either when t_g fires, or when it becomes disabled by the firing of a competitive transition. During the activity cycle the marking can change inside the subset of connected markings in which t_g is enabled. The age variable associated to t_g grows continuously during the activity cycle starting from 0.
- *Age Memory* - If t_g is an age memory transition, its activity cycle starts as soon as t_g becomes enabled when dormant, and ends only at the firing of t_g itself. During the activity cycle of an age memory transition there is no restriction on the markings reachable by the marking process. The age memory policy is the only policy in which a transition can be active even in markings in which is not enabled. During the activity cycle, the age variable is non-decreasing in the sense that increases continuously in those markings in which t_g is enabled and maintains its constant positive value in those markings in which t_g is not enabled. The enabling/disabling condition of t_g during its activity cycle is tracked by introducing a reward (indicator) variable which is set to 1 in those markings in which t_g is enabled and set to 0 in those markings in which t_g is not enabled. With this assignment, the value of the age variable versus time can be computed as the total accumulated reward.

Definition 7 - *Activity cycles are non-overlapping if there exists a dominant transition whose activity cycle strictly contains the activity cycles of all the active transitions.*

Definition 8 - *A MRSPN with non-overlapping activity cycles is a MRSPN in which all the regeneration periods are dominated by a single transition: any two successive regeneration time points correspond to the start and to the end of the activity cycle of the dominant transition.*

Definition 8, includes the possibility that the activity cycles of GEN transitions are completely contained into the activity cycle of the dominant one, hence allowing the simultaneous enabling of different GEN transitions inside the same subordinated process. In particular in [10], the class of models for which the process subordinated to any GEN transition is a reward semimarkov process is defined and analyzed. In order to arrive to closed form expressions for the global and local kernels of the underlying *MRP*, the following situations are examined separately.

3.6.1 Enabling memory dominant transition

The dominant GEN transition t_g is of enabling type. The next regeneration time point occurs because one of the following two mutually exclusive events:

- t_g fires: this event can be formulated as a completion time problem [33, 7] when the age variable a_g reaches a value equal to the firing requirement γ_g.
- t_g is disabled: this event can be formulated as a first passage time in the subset of states in which t_g becomes disabled.

Accordingly, the following two types of subordinated processes can be distinguished.

TYPE A - The subordinated process is a CTMC: no other GEN transitions are activated during the activity cycle of t_g.

A subordinated process of Type A is the only one arising from Definition 3 for *DSPN* [3, 14, 35] and from Definition 4 for *MRSPN** [15, 27]. All the examples reported in the mentioned references belong to this case.

TYPE B - The subordinated process is a semimarkov process: during the activity cycles of t_g other GEN transitions can be activated one at the time (or more generally according to the rules stated in [23]).

The steady state analysis of a *MRSPN* with Type B subordinated process has been considered in [16]. The proposed algorithm is based on an efficient computational extension of the randomization technique [28], assuming that GEN distributions are piecewise defined by polynomials multiplied by exponential expressions. This class of distributions is called *expolynomial*.

3.6.2 Age memory dominant transition

The situation in which the dominant GEN transition t_g is of age type has been addressed for the first time in [11]. The only criterion for the termination of the activity cycle is the firing of t_g, and the state space of the subordinated process contains all the states reachable during the activity cycle of t_g. During its activity cycle transition t_g can be either enabled or disabled. The corresponding binary reward variable is set equal to 1 in the states of the subordinated process in which t_g is enabled and equal to 0 in the states of the subordinated process in which t_g is not enabled. The firing of t_g can be formulated as a completion time problem [33, 7] in a reward stochastic model when the age variable a_g (calculated as the total accumulated reward) reaches the firing requirement γ_g.

Two types of subordinated processes can be distinguished also in this case.

TYPE C - The subordinated process is a reward CTMC: during the activity cycle of t_g no other GEN transitions are activated.

TYPE D - The subordinated process is a reward semi-Markov process: during the activity cycle of t_g other GEN transitions can be activated one at the time (or more generally according to the rules stated in [23]).

The next Section reports an example in which Type C and Type D subordinated processes are combined in the same *PN*.

3.6.3 DET transitions

A particular case arises when all the GEN transitions are assumed to be of DET type. The *MRSPN* with non overlapping activity cycles and DET transitions can be considered as a direct generalization of the *DSPN* model of Definition 3, where the modeling power is augmented to accommodate the presence of memory policies of both enabling and age type. In [10], a procedure is given to derive the $\mathbf{K}(x)$ and $\mathbf{E}(x)$ matrices (3) when the subordinated process is of Type D (a semimarkov reward model) and the non-exponential transitions are DET.

The GEN case can be derived from the solution of the DET case by the following argument. Let us fix a value $\gamma_g = w$ of the r.v. associated to the dominant transition, and let us derive $K_{ij}(x|w)$ as in the pure DET case. Then, given that $G_g(w)$ is the Cdf of γ_g:

$$K_{ij}(x) = \int_{w=0}^{\infty} K_{ij}(x|w)\, dG_g(w) \tag{8}$$

A similar argument holds for the entries of matrix $\mathbf{E}(x)$. The solutions for subordinated processes of Types A, B and C can be derived as special cases of the solution of Type D.

4 Example: Finite Queue with Preemption

We carry on a comparison between the modeling power and the numerical results obtained from the Laplace Transform Method *(LTM)* applied to *MRSPN* (or *DSPN* in particular cases) and the *PHSPN* model through the analysis of a simple queueing systems with different kinds of preemption. We consider, as a base example, the M/D/1/2/2 (a closed queueing system with two buffer positions and two customers) introduced in [3]. The non-preemptive service mechanism has been already analyzed in [3] for what concerns the steady state measures and revisited in [14] for what concerns the transient behavior. In the figures, EXP transitions are drawn as empty rectangles, DET transitions as filled rectangles and immediate transitions as thin bars.

4.1 Case I - Non Preemptive Queue

The *PN* for the M/D/1/2/2 system, proposed in [3], is reported in Figure 1. Place p_1 contains "thinking" customers (i.e. awaiting to submit a job) and transition t_1 represents the submission of jobs. Jobs queueing for service are represented by tokens in p_2. A token in p_3 means that the server is busy while a token in p_4 means that the server is idle. Transition t_2 is the job service time; when the job is completed the customer returns in his thinking state. Transition t_3 is an immediate transition modelling the start of service i.e. the transfer of the job from the queue to the server.

In [3, 14], the following assumptions were made. t_1 is EXP with firing rate $m_1 \cdot \lambda$ being m_1 the number of tokens in p_1 and $\lambda = 0.5$ *job/hour*. t_2 is a DET transition modeling a constant service time of duration $d = 1.0$ *hour*.

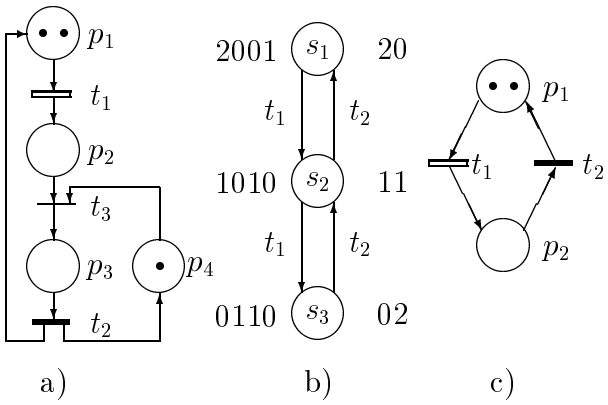

Figure 1 - a) - PN modelling the atomic operation of a M/D/1/2/2 (after [3]); b) - corresponding reduced reachability graph; c) - simplified PN.

The reduced reachability graph of the *PN* (after eliminating the vanishing markings arising from the immediate transition t_i [2]) is composed of three states, denoted by s_1, s_2 and s_3 in Figure 1b. The *PN* of Figure 1a is a *DSPN* according to Definition 3 and shows in details the atomic steps by which a customer submits a job and the job is serviced. Figure 1c shows, however, a simpler *PN* isomorphic to the one of Figure 1a.

Tokens in place p_1 of Figure 1c represent customers in the thinking state, while p_2 contains the jobs in the queue (included the one under service). t_1 models the submitting time and t_2 is the service time. It is easy to verify that the above *PN* generates the same marking process $\mathcal{M}(x)$ of Figure 1b when t_1 is EXP with rate $m_1 \cdot \lambda$ and t_2 is DET. The probabilities versus time of the two states s_1 and s_3 are reported in Figure 2 in solid line, computed by means of the Laplace transform method *(LTM)*.

Approximating the *DSPN* of Figure 1c by means of the *PHSPN* model is straightforward. Transition t_2 is assigned a *PH* distribution and an enabling memory policy. Since the Erlang distribution is the *PH* with the minimum coefficient of variation [4] it is appropriate to approximate the *DSPN* by assigning t_2 an Erlang distribution of increasing order. In Figure 2 we compare the results obtained from the *PHSPN* model, by reporting the behavior of the state probabilities versus time in two cases: when *i)* the random firing time assigned to t_2 is Erlang(5) (dashed line), and *ii)* when is Erlang(100) (dotted line). In both cases the expected value of the Erlang matches with the value $d = 1.0\, hours$ of the DET model, being all the other parameters unchanged. It is interesting to observe that the local maxima and minima in the probability behavior do not appear with the Erlang(5), while the visual agreement is very satisfactory in the case of the Erlang(100).

As a further comparison, Table I shows the values of the steady state probabilities calculated from *LTM* for the *DSPN* and from the *PHSPN* model when t_2 is assumed to be Erlang(5), Erlang(10), Erlang(100) and Erlang(1000), respectively. It should be stressed that the present case can be considered as a worst case example since a DET type variable can be closely approximated by a *PH* only as the number of stages grows

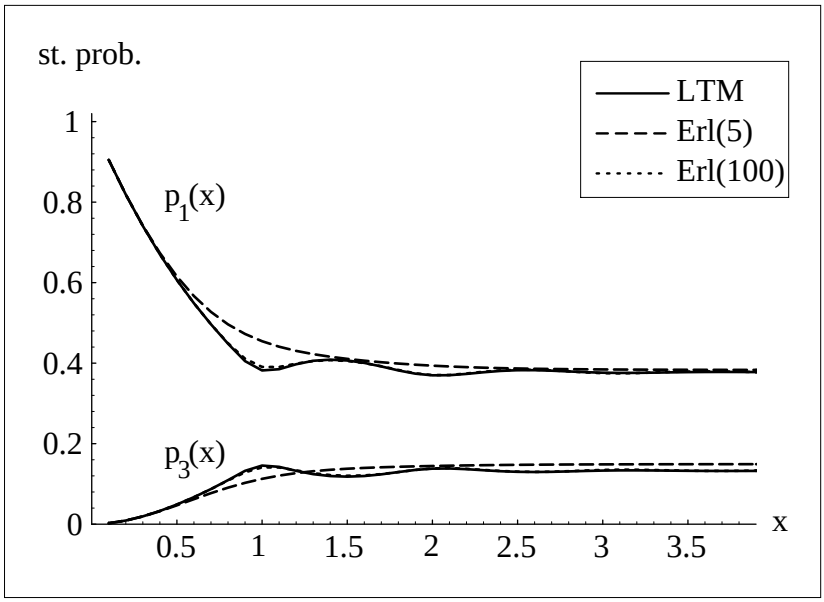

Figure 2 - Transient behavior of the state probabilities for the non preemptive M/D/1/2/2

to ∞ [4].

4.2 Preemptive Queue with Identical Customers

Let us assume a M/D/1/2/2 with a preemptive service and the same kind of customers. The job in execution is preempted as soon as a new job joins the queue. Two cases can be considered depending whether the job restarted after preemption is resampled from the same distribution function *(prd)*, or is resumed *(prs)*.

4.2.1 Case II - prd policy

With reference to Figure 1c, each time transition t_1 fires (a thinking customer submits a job) while p_2 is marked (a job is currently under service) transition t_2 should be reset and resampled. In the *PHSPN* model this mechanism can be simply realized by assigning to t_2 a resampling policy. It is easy to prove that the underlying process $\mathcal{M}(x)$ is a semi-Markov process, since each time the DET transition t_2 is entered, a regeneration point is produced since a new job starts.

Even if the class of semi-markov processes is a proper subclass of the Markov regenerative processes, the above preemptive mechanism cannot be generated from the *DSPN* of Definition 3. In fact, since t_1 is not competitive with respect to t_2, the firing of the former does not disable the latter, that indeed is not resampled. The *PN* in Figure 3a describes a correct *DSPN* with the required preemption policy. Place p_1 in Figure 3a

Table I - *Steady state probabilities for the non preemptive queue (Case I)*

State	*DSPN* (*LTM*)	*PHSPN*			
		Erl(5)	*Erl(10)*	*Erl(100)*	*Erl(1000)*
s_1	0.37754	0.38307	0.38039	0.37783	0.37757
s_2	0.48984	0.46773	0.47845	0.48867	0.48972
s_3	0.13262	0.14920	0.14116	0.13350	0.13271

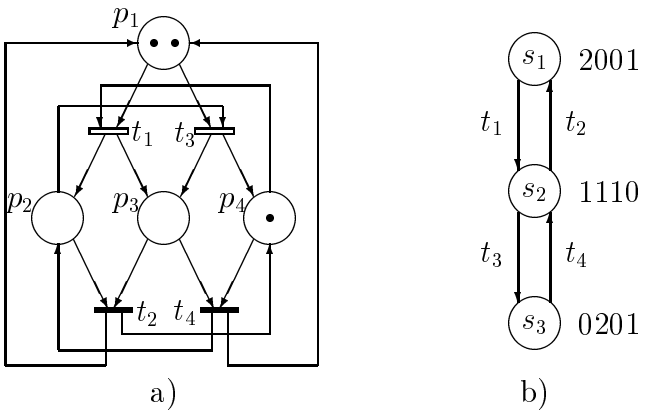

Figure 3 - Preemptive M/D/1/2/2 ith identical customers

contains the customers thinking, while place p_2 contains the number of submitted jobs (included the one under service). Place p_3 represents a single job getting service: service is interrupted (t_2 is disabled) if a new job joins the queue (transition t_3 fires before t_2). t_1 and t_3 are assigned the EXP submitting time and transitions t_2 and t_4 a DET service time. Assigning an enabling memory policy to t_2 and t_4 the *prd* service mechanism is generated, while respecting the requirements of Definition 3.

Table II compares the steady state probabilities assuming the submitting and service time distributions identical to the non preemptive case. The transient behaviors of the probabilities versus time of states s_1 and s_3 are reported in Figure 5 (Case II) computed by means of the *LTM* for the *DSPN* model. Similarly, Figure 6 (Case II) shows the behaviors for the *PHSPN* model with the service time given by an Erlang(100).

Table II - *Steady state probabilities with equal customers*

State	*DSPN* (*LTM*)	*PHSPN*			
		Erl(5)	*Erl(10)*	*Erl(100)*	*Erl(1000)*
Case II - prd policy					
s_1	0.33942	0.35317	0.34642	0.34014	0.33950
s_2	0.44038	0.43122	0.43572	0.43991	0.44034
s_3	0.22019	0.21561	0.21786	0.21995	0.22017

4.2.2 Case III - prs policy

The *prs* policy means that when a new job joins the queue the job under service is preempted until the newly arrived job completes his service. The preempted job is then resumed and put in execution from the point of preemption without loss of the previously performed work.

The *prs* mechanism for the M/D/1/2/2 queue corresponds to the *PN* of Figure 3 when t_2 and t_4 is assigned an age memory policy. Each time t_2 is disabled without firing (t_3 fires before t_2) the age variable a_2 is not reset. Hence, as the second job completes (t_4 fires), the system returns in s_2 keeping memory of the value of a_2, so that the time to complete the interrupted job can be evaluated as the residual service time given a_2.

With the above assignments, the *PN* of Figure 3a is a *MRSPN* with non overlapping activity cycles. A detailed analysis of this example is in [11]. The regeneration time points in the marking process $\mathcal{M}(x)$ correspond to the epochs of entrance in markings in which all the age variables are equal to zero (Definition 5). By inspecting Figure 3b, the regeneration time points result to be the epochs of entrance in s_1 and of entrance in s_2 from s_1. The process subordinated to state s_1 is a single step *CTMC* (being the only enabled transition t_1 exponential) and includes the only reachable state s_2.

The process subordinated to state s_2 is dominated by transition t_2 and includes the states s_3, s_2. Since s_2 is the only state in which t_2 is enabled, the reward variable is set equal to 1 in s_2 and equal to 0 elsewhere. With the given reward rates, a_2 is equal to the cumulative sojourn time in s_2 and, therefore, counts the total time during which t_2 is enabled before firing. During the firing of t_2 the subordinated process alternates between states s_2 and s_3. Since t_4 is non-exponential (DET in this case) the subordinated process is a reward semi-Markov process of Type D (Section 3.6.2).

The transient behavior is depicted in Figure 5 (*Case III*) computed by the *LTM* for *MRSPN* and in Figure 6 computed from the *PHSPN* model with the DET transitions approximated by an Erlang(100).

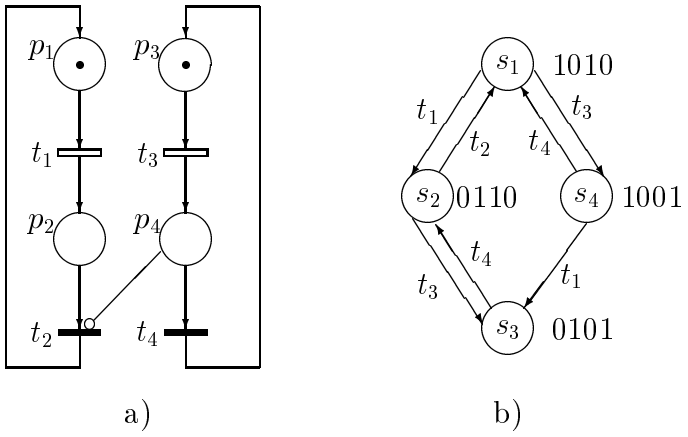

Figure 4 - Preemptive M/D/1/2/2 queue with two classes of customers

In this case, the service mechanism is a preemptive LIFO [41] and satisfies the requirements of symmetric queues. Hence, the steady state probabilities are insensitive to the distribution of the service time and depend only on its mean d ($d = 1$ in this example) according to the following balance equations [41]:

$$2\lambda\, d\, \pi_{s1} \;=\; \pi_{s2} \quad ; \quad \lambda\, d\, \pi_{s2} \;=\; \pi_{s3}\ .$$

Solving the above balance equations provides: $\pi_{s1} = 0.4$, $\pi_{s2} = 0.4$ and $\pi_{s3} = 0.2$.

4.3 Preemptive Queue with Different Classes of Customers

An interesting case arises when the two customers are of different classes, and customer of class 2 preempts customer of class 1 but not vice versa. A *PN* illustrating the M/D/1/2/2 queue in which the jobs submitted by customer 2 have higher priority over the jobs submitted by customer 1 is reported in Figure 4. Place p_1 (p_3) represents customer 1 (2) thinking, while place p_2 (p_4) represent job 1 (2) under service. Transition t_1 (t_3) is the submission of a job of type 1 (2), while transition t_2 (t_4) is the completion of service of a job of type 1 (2). The inhibitor arc from p_4 to t_2 models the described preemption mechanism: as soon as a type 2 job joins the queue the type 1 job eventually under service is interrupted.

If we assume that the service time is not exponentially distributed, two possible preemption policies can be considered depending whether the job of type 1 is resampled after preemption (*prd* case) or is resumed (*prs* case).

4.3.1 Case IV - prd policy

Since this policy can be realized by assigning to the service transitions t_2 and t_4 an enabling memory policy, the present case is included in the *DSPN* model of Definition 3. Table III shows the steady state probability values computed from (7) and from the *PHSPN* model with various Erlang approximations.

Table III - *Steady state probabilities with different customers*

State	*MRSPN*	*PHSPN*			
	(LTM)	*Erl(5)*	*Erl(10)*	*Erl(100)*	*Erl(1000)*
Case IV - prd policy					
s_1	0.35015	0.36194	0.35618	0.35076	0.35021
$s_2 + s_4$	0.45429	0.44193	0.44801	0.45365	0.45423
s_3	0.19556	0.19613	0.19582	0.19558	0.19556
Case V - prs policy					
s_1	0.39291	0.39458	0.39377	0.39300	*n.a.*
$s_2 + s_4$	0.42835	0.42166	0.42493	0.42800	*n.a.*
s_3	0.17874	0.18375	0.18130	0.17900	*n.a.*

4.3.2 Case V - prs policy

Under a *prs* service policy, after completion of the type 2 job, the interrupted type 1 job is resumed continuing the new service period from the point reached just before the last interruption. In the *PN* of Figure 4a this service policy is realized by assigning to transitions t_2 and t_4 an age memory policy.

¿From Figure 4b, it is easily recognized that s_1, s_2 and s_4 can all be regeneration states, while s_3 can never be a regeneration state (in s_3 either a job of type 1 or 2 is always in execution so that their corresponding memory variables are never simultaneously 0). Only exponential transitions are enabled in s_1 and the next regeneration states can be either s_2 or s_4 depending whether t_1 or t_3 fires first. In s_4 the dominant transition is t_4 and the next regeneration marking can be either state s_1 or s_2 depending whether during the execution of the type 2 job a type 1 job does require service (but remains blocked until completion of the type 2 job) or does not. The reward variable is set to 1 in states s_3 and s_4, where t_4 is enabled, so that the age variable a_4 counts the total time spent in either of the two states s_3 or s_4. The subordinated process is a Type C reward *CTMC*. ¿From s_2 the dominant transition is t_2 and the next regeneration state can be only s_1. During the firing of t_2 multiple cycles (s_2 - s_3) can occur depending whether type 2 jobs arrive to interrupt the execution of the type 1 job. The reward variable is set to 1 in state s_2 so that the subordinated process is a Type D reward-*SMP* (t_4 is GEN). Subordinated processes of both Types C and D are mixed in the same net. A detailed

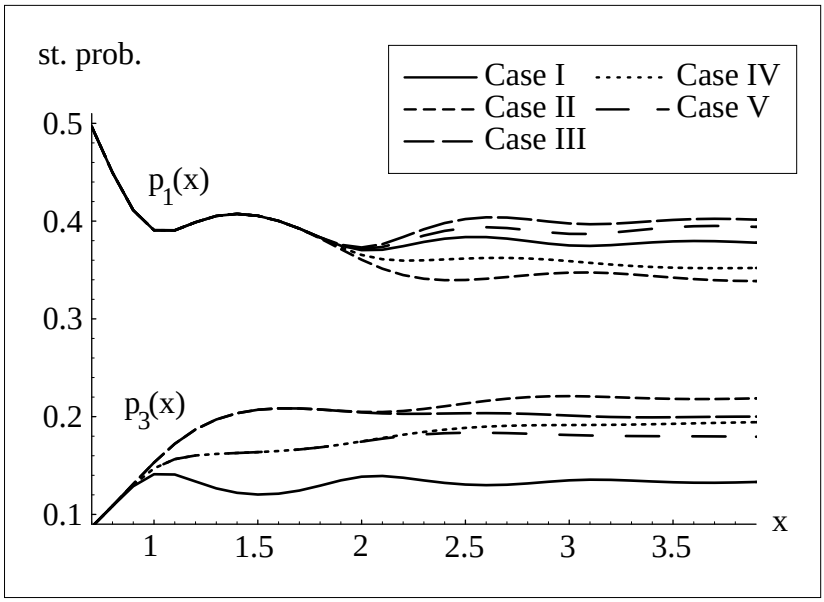

Figure 5 - Comparison of the state probabilities computed by the Laplace transform method for *DSPN* (Case I, Case II and Case IV) and for *MRSPN* (Case III and Case V).

derivation of the closed form equations for this example is in [11].

The steady state results are reported in Table III. Note that the values corresponding to the Erlang(1000) are not available *(n.a.)* due to the explosion of the state space (see Section 5.1). The transient probabilities for Case IV and Case V are reported in Figure 5 computed by means of the *LTM* for the *DSPN* (Case IV) and for the *MRSPN* with non overlapping intervals (Case V). The corresponding results, computed for the *PHSPN* model, with an Erlang(100) assigned to the DET transitions, are similarly shown in Figure 6.

The steady state results for Case I, Case II and Case IV where also checked using the package TimeNET [26] based on the supplementary variable technique.

5 Computational complexity

Let us briefly summarize the elementary computational steps for the evaluation of the transient solution in the two considered methodologies (*MRSPN* and *PHSPN*). The *PHSPN* solution is fully supported by a tool [22], while the Laplace transform method for the transient analysis of *MRSPN* requires manual and automatic manipulation. The method of supplementary variables for *DSPN*, implemented in TimeNET [26], is also fully supported by a tool but is presently restricted to the steady state solution only.

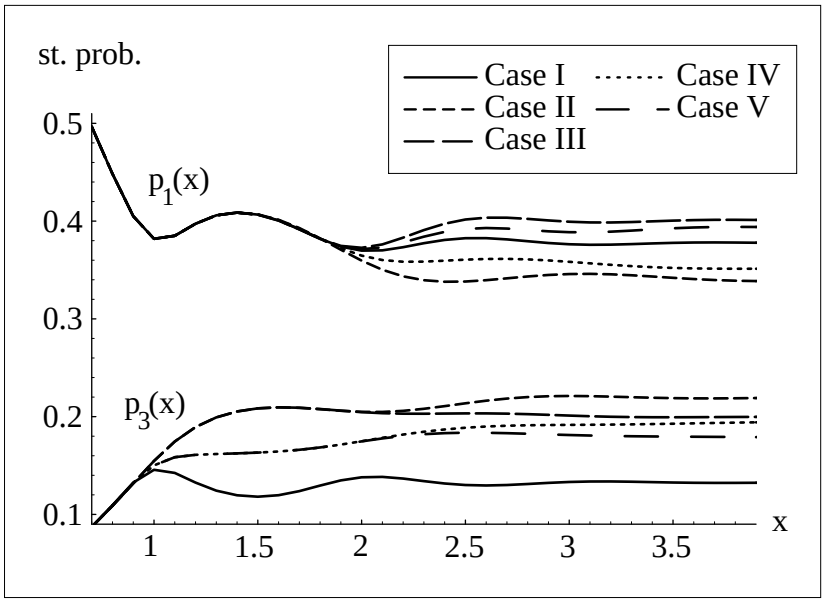

Figure 6 - Comparison of the state probabilities computed from the *PHSPN* for the 5 examined Cases and with Erlang(100).

5.1 Evaluation of *PHSPN* model

For the evaluation of this model we used the ESP tool ([22]). The procedure can be divided into the following steps:

1. generation of the reachability tree;
2. generation of the expanded *CTMC*;
3. solution of the resulting *CTMC*.

Step 1) is standard. The computational complexity of steps 2) and 3) depends on the number of tangible states and on the order of the *PH* distribution associated to each transition. With *PH* distributions of order ν the cardinality of the expanded *CTMC* is $n_H = 2\nu+1$ in Case I, $n_H = 2\nu+1$ in Case II, $n_H = \nu^2+\nu+1$ in Case III, $n_H = 3\nu+1$ in Case IV and $n_H = \nu^2+2\nu+1$ in Case V. As mentioned in Section 3.3, the cardinality of the expanded state space n_H is strongly influenced by the memory policies. In this trivial example, with $\nu = 100$ (Erlang100) the generation of the *CTMC* takes 2 m for Cases III and V, and the whole analysis two further minutes on a IBM RISC 6000 computer.

5.2 Laplace Transform Method for MRSPN

Let us first suppose that all the GEN transitions are DET. The computational method can be divided in the following steps [14]:

1. generation of the reachability tree;

2. manual derivation of the entries of the $\mathbf{K}^{\sim}(s)$ and $\mathbf{E}^{\sim}(s)$ matrices symbolically in the Laplace transform domain;

3. symbolical matrix inversion and matrix multiplication by using a standard package (e.g. MATHEMATICA) in order to obtain the $\mathbf{V}^{\sim}(s)$ matrix (Equation 7) in the LT domain;

4. time domain solution obtained by a numerical inversion of the entries of the $\mathbf{V}^{\sim}(s)$, resorting to the Jagerman's method [32]. For the sake of uniformity, this step has been implemented in MATHEMATICA language.

In the GEN case, point 2) in the above list should be replaced by

$2'$ manual derivation of the entries of the $\mathbf{K}^{\sim}(s)$ and $\mathbf{E}^{\sim}(s)$ matrices symbolically in Laplace transform domain as in the DET case;

$2''$ unconditioning of the entries of the $\mathbf{K}^{\sim}(s)$ and $\mathbf{E}^{\sim}(s)$ matrices, according to the Cdf of the GEN distributions (Equation 8).

Step 1) can be performed with any *PN* package. Step 2) is done manually, and its difficulty depends on the non-zero entries of the involved matrices, and on the complexity of the process subordinated to the dominant GEN transitions. The complexity increases going from subordinated processes of Type A and C (*CTMC*) to subordinated processes of Type B and D (*SMP*), and going from Types A and B (all the states have the same reward) to Types C and D (states are assigned a binary reward).

The computational complexity of step 3) depends on the dimension of the matrices (i.e. the number of tangible markings) and the complexity of the elements of the kernels (the difficulty of step 3 is related to the difficulty of step 2). The complexity of the numerical inversion at step 4) also depends on two factors; the complexity of the function to invert, and the prescribed accuracy. For the example described in the previous section, the computational time for the symbolic inversion was not significant, while the numerical inversion required about 30 s on an IBM RISC 6000 machine, for each point of the transient solution.

5.3 Discussion

Even if the deterministic distribution is typically non *PH*, an approximation error for the steady state probabilities of the order of 10^{-2} is reached by replacing the DET transition with an Erlang(5) and an error of the order of 10^{-4} by replacing the DET transition with an Erlang(1000). The use of *PH* distributions and of the *PHSPN* model offers the modeler a flexible tool for prescribing various interactions among the timed activities. Moreover, if the random variables of the system to be modeled are really of *PH* type, the *PHSPN* provides exact results. Otherwise, a preliminary step is needed in which the random times of the system are approximated by *PH* random variables resorting to a suitable estimation technique [8]. The expansion of the state space is, of course, a cause of non-negligible difficulties, since it worsens the problem of the exponential growth of the

state space both with the model complexity, and with the order of the *PH* distribution assigned to each transition.

The *MRSPN* model, combining GEN (or DET) firing times with exponential firing times, offers an innovative approach in many practical applications. At the present state of the art, no automatized tools are available for the generation of the matrices $\mathbf{K}(x)$ and $\mathbf{E}(x)$ and for the solution of the convolution equation versus time. The Laplace transform technique, used in the examples, does not seem suited for a complete numerical automatization. An alternative numerical approach could be based on the direct solution of the convolution equation (4) in time domain or in the solution of a system of partial differential equations arising from the inclusion of supplementary variables [25].

Acknowledgments

The authors thank R. German for making available the package TimeNET. The work of Andrea Bobbio was partially supported by CNR grant No. 96.01939.CT12. Miklós Telek was partially supported by OTKA grant No. T-16637.

References

[1] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. The effect of execution policies on the semantics and analysis of stochastic Petri nets. *IEEE Transactions on Software Engineering*, SE-15:832–846, 1989.

[2] M. Ajmone Marsan, G. Balbo, and G. Conte. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 2:93–122, 1984.

[3] M. Ajmone Marsan and G. Chiola. On Petri nets with deterministic and exponentially distributed firing times. In *Lecture Notes in Computer Science*, volume 266, pages 132–145. Springer Verlag, 1987.

[4] D. Aldous and L. Shepp. The least variable phase type distribution is Erlang. *Stochastic Models*, 3:467–473, 1987.

[5] A. Bertoni and M. Torelli. Probabilistic Petri nets and semi Markov processes. In *Proceedings 2-nd European Workshop on Petri Nets*, 1981.

[6] A. Bobbio. Stochastic reward models in performance/reliability analysis. *Journal on Communications*, XLIII:27–35, January 1992.

[7] A. Bobbio and M. Telek. Task completion time. In *Proceedings 2nd International Workshop on Performability Modelling of Computer and Communication Systems (PMCCS2)*, 1993.

[8] A. Bobbio and M. Telek. A benchmark for PH estimation algorithms: results for Acyclic-PH. *Stochastic Models*, 10:661–677, 1994.

[9] A. Bobbio and M. Telek. Computational restrictions for SPN with generally distributed transition times. In D. Hammer K. Echtle and D. Powell, editors, *First European Dependable Computing Conference (EDCC-1), Lecture Notes in Computer Science*, volume 852, Springer-Verlag, pages 131–148, 1994.

[10] A. Bobbio and M. Telek. Markov regenerative SPN with non-overlapping activity cycles. In *International Computer Performance and Dependability Symposium - IPDS95*, pages 124–133. IEEE CS Press, 1995.

[11] A. Bobbio and M. Telek. Transient analysis of a preemptive resume M/D/1/2/2 through Petri nets. Technical report, Department of Telecommunications - Technical University of Budapest, April 1994.

[12] P. Chen, S.C. Bruell, and G. Balbo. Alternative methods for incorporating non-exponential distributions into stochastic timed Petri nets. In *Proceedings International Workshop on Petri Nets and Performance Models - PNPM89*, pages 187–197. IEEE Computer Society, 1989.

[13] G. Chiola. *GreatSPN* 1.5 Software architecture. In G. Balbo and G. Serazzi, editors, *Computer Performance Evaluation*, pages 121–136. Elsevier Science Publishers, 1992.

[14] Hoon Choi, V.G. Kulkarni, and K. Trivedi. Transient analysis of deterministic and stochastic Petri nets. In *Proceedings of the 14-th International Conference on Application and Theory of Petri Nets*, Chicago, June 1993.

[15] H. Choi, V.G. Kulkarni, and K. Trivedi. Markov regenerative stochastic Petri nets. *Performance Evaluation*, 20:337–357, 1994.

[16] G. Ciardo, R. German, and C. Lindemann. A characterization of the stochastic process underlying a stochastic Petri net. *IEEE Transactions on Software Engineering*, 20:506–515, 1994.

[17] G. Ciardo and C. Lindemann. Analysis of deterministic and stochastic Petri nets. In *Proceedings International Workshop on Petri Nets and Performance Models - PNPM93*, pages 160–169. IEEE Computer Society, 1993.

[18] G. Ciardo, J. Muppala, and K.S. Trivedi. SPNP: stochastic Petri net package. In *Proceedings International Workshop on Petri Nets and Performance Models - PNPM89*, pages 142–151. IEEE Computer Society, 1989.

[19] E. Cinlar. *Introduction to Stochastic Processes.* Prentice-Hall, Englewood Cliffs, 1975.

[20] J.A. Couvillon, R. Freire, R. Johnson, W.D. Obal, M.A. Qureshi, M. Rai, W. Sanders, and J.E. Tvedt. Performability modeling with UltraSAN. *IEEE Software*, 8:69–80, September 1991.

[21] D.R. Cox. The analysis of non-markovian stochastic processes by the inclusion of supplementary variables. *Proceedings of the Cambridge Phylosophical Society*, 51:433–440, 1955.

[22] A. Cumani. Esp - A package for the evaluation of stochastic Petri nets with phase-type distributed transition times. In *Proceedings International Workshop Timed Petri Nets*, pages 144–151, Torino (Italy), 1985. IEEE Computer Society Press no. 674.

[23] J. Bechta Dugan, K. Trivedi, R. Geist, and V.F. Nicola. Extended stochastic Petri nets: applications and analysis. In *Proceedings PERFORMANCE '84*, Paris, 1984.

[24] G. Florin and S. Natkin. Les reseaux de Petri stochastiques. *Technique et Science Informatique*, 4:143–160, 1985.

[25] R. German. *Transient Analysis of deterministic and stochastic Petri nets by the method of supplementary variables.* Internal Report Technische Universität Berlin (to be presented MASCOT'95), 1994.

[26] R. German, C. Kelling, A. Zimmermann, and G. Hommel. *TimeNET - A toolkit for evaluating non-markovian stochastic Petri nets.* Report No. 19 - Technische Universität Berlin, 1994.

[27] R. German and C. Lindemann. Analysis of stochastic Petri nets by the method of supplementary variables. *Performance Evaluation*, 20:317–335, 1994.

[28] D. Gross and D. Miller. The randomization technique as a modeling tool and solution procedure for transient Markov processes. *Operations Research*, 32:343–361, 1984.

[29] P.J. Haas and G.S. Shedler. Regenerative stochastic Petri nets. *Performance Evaluation*, 6:189–204, 1986.

[30] P.J. Haas and G.S. Shedler. Stochastic Petri nets with simultaneous transition firings. In *Proceedings International Workshop on Petri Nets and Performance Models - PNPM87*, pages 24–32. IEEE Computer Society, 1987.

[31] B.R. Haverkort and K. Trivedi. Specification techniques for Markov Reward Models. *Discrete Event Dynamic Systems: Theory and Applications*, 3:219–247, 1993.

[32] D.L. Jagerman. An inversion technique for the Laplace transform. *The Bell System Technical Journal*, 61:1995–2002, October 1982.

[33] V.G. Kulkarni, V.F. Nicola, and K. Trivedi. On modeling the performance and reliability of multi-mode computer systems. *The Journal of Systems and Software*, 6:175–183, 1986.

[34] R. Lepold. PEPNET: A new approach to performability modelling using stochastic Petri nets. In *Proceedings 1st International Workshop on Performability Modelling of Computer and Communication Systems*, pages 3–17, University of Twente - Enschede (NL), 1991.

[35] C. Lindemann. An improved numerical algorithm for calculating steady-state solutions of deterministic and stochastic Petri net models. *Performance Evaluation*, 18:75–95, 1993.

[36] C. Lindemann. DSPNexpress: a software package for the efficient solution of deterministic and stochastic Petri nets. *Performance Evaluation*, 22:3–21, 1995.

[37] M.K. Molloy. On the integration of delay and throughput measures in distributed processing models. Technical report, Phd Thesis, UCLA, 1981.

[38] S. Natkin. Les reseaux de Petri stochastiques et leur application a l'evaluation des systemes informatiques. Technical report, These de Docteur Ingegneur, CNAM, Paris, 1980.

[39] M.F. Neuts. *Matrix Geometric Solutions in Stochastic Models.* Johns Hopkins University Press, Baltimore, 1981.

[40] A. Reibman, R. Smith, and K.S. Trivedi. Markov and Markov reward model transient analysis: an overview of numerical approaches. *European Journal of Operational Research*, 40:257–267, 1989.

[41] R.W. Wolff. *Stochastic Modeling and the Theory of Queues.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1989.