Computational Restrictions for SPN with Generally Distributed Transition Times

Andrea Bobbio and Miklós Telek

Dipartimento di Elettronica per l'Automazione
Università di Brescia, 25123 Brescia, Italy

Department of Telecommunications
Technical University of Budapest, Budapest, Hungary

February 5, 2001

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 describes 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.

1 Introduction

The designer and the analyst of a system are in first instance interested in the solution of the modeling problem, not in how this solution is actually derived. They should be able to describe their system in such a way that it is easy and natural to use. The *modeler's representation* should include enough information to build up an *analytical representation* suitable for numerical solution, and should also contain the specification of the measures of interests. The modeler's representation should then automatically be transformed into the analytical representation. Finally the numerical results should be again automatically mapped back into the modeler's representation, so that the user of the tool can interpret them in that context. For Markovian systems several tools have been developed in recent years, based on various specification paradigms, as surveyed in [26].

There are, however, situations that are not covered by these tools. One typical situation occurs when the random times characteristic of the system are not exponential. A second situation occurs when the analyst requires the computation of stochastic measures (like the distribution function of cumulative measures [34, 7]) whose numerical evaluation cannot be performed by solving a set of linear first order equations typical of Markovian systems.

In recent years several classes of Stochastic Petri Net (*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 general, the stochastic process underlying a *GDT_SPN* does not have a numerically tractable analytical formulation, while a simulative solution has been investigated in [24].

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. Dugan et al. have studied the conditions under which the stochastic realization of the *GDT_SPN* is a semi-Markov process [21]. Cumani [20] has realized a package in which each *PN*-transition can be assigned a *PH* distributed firing time. We refer to this model in the following as *PHSPN.*

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 analysis was elaborated in [3]. An improved steady state algorithm was presented in [29], and some structural extensions were investigated in [15]. Choi et al. [13] have developed a technique for the transient analysis of the state probabilities of the *DSPN* model. Recently, Choi et al. [12] and German and Lindemann [23] have extended the potentiality of the model by allowing the presence in each marking of a transition with a generally distributed firing time. In [12], the authors have shown that the underlying stochastic process is a *Semi Markov Regenerative* process, for which a transient as well as a steady state solution can be given. For this reason, Choi et. al. refer to this model as *Markov Regenerative SPN (MRSPN)* [12]. A classification of *GDT_SPN*s and of the related underlying stochastic processes is in Ciardo et al. [14].

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 various restrictions considered in the literature are briefly described with the intent of stressing the basic modeling 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* model is formally defined in Section 2. In Section 3 a brief survey of the most recent restrictions appeared in the literaure is reported. Two restrictions are described in more details, namely the *PHSPN* model implemented by Cumani in [20] and the *DSPN* model described by Choi et al. in [13]. A comparative discussion of the modeling power of the considered models is reported in Section 4. In Section 5, starting from a simple queuing system, increasing modelling complexities are added in order to show how the considered models react to these added structures. The algorithmical complexity of the numerical solutions is discussed in Section 6.

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);
- $T = \{t_1, t_2, \ldots, t_{nt}\}$ 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 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 $X_{\mathcal{T}}(t)$, $(t \geq 0)$, 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 was defined in [1], with particular emphasis to the semantical interpretation of the model. We refer to this model as *Generally Distributed Transition_SPN (GDT_SPN)*.

Definition 3 - *A stochastic GDT_SPN is a marked SPN in which:*

- *To any 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 the (possibly marking dependent) Cumulative distribution function $G_k(x|M)$.*
- *A set of specifications are given for univocally defining the sstochastic 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 structure and the set of Cdf's $G_k(x|M)$. 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 semantics of different execution policies has been discussed in [1]. 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 transition is enabled in a given marking, the transition fires whose associated random delay is statistically the minimum. The *Memory Policy* is the part of the set of specifications of the *execution policy* that defines how the process is conditioned upon the past. We associate to each transition t_k an *age variable* a_k. The way in which a_k is related to the past history $Z_{(j)}$ determines the different memory policies. We consider three alternatives:

- *Age memory* - The age variable a_k accounts for the work performed by the activity corresponding to t_k from its last firing up to the current epoch. The firing distribution depends on the residual time needed for this activity to complete given a_k.
- *Enabling memory* - The age variable a_k accounts for the work performed by the activity corresponding to t_k from the last epoch in which t_k has been enabled. The firing distribution depends on the residual time needed for this activity to complete given a_k. When transition t_k is disabled (even without firing) the corresponding enabling age variable is reset.
- *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 present marking.

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 stocastic process cannot etend beyond a single cycle of enable/disable of the transition with enabling memory policy. 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. The memory of the process extends up to the first epoch in which the transition has been enabled for the first time after a firing.

3 Computational Restrictions

The marking process $X_{\mathcal{T}}(\tau)$ does not have, in general, an analytically tractable formulation, while a simulative approach has been described in [24, 25]. Various restrictions of the general model have been discussed in the literature such that the underlying marking process $X_{\mathcal{T}}(\tau)$ is confined to belong to a known class of analytically tractable problems.

3.1 Exponentially Distributed SPN

When the random variables γ_k associated to the *PN* transitions are exponentially distributed, the dynamic behaviour of the net can be mapped into a continuous time homogeneous Markov chain *(CTMC)*, with state space isomorphic to the reachability graph of the net. This restriction is the most popular in the literature [31, 22, 2], and a number of packages are built on this model [11, 16, 30, 28].

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 [32, 5] but is of little interest in applications where it is difficult to imagine a situation where the firing of each transition of the *PN* has the effect of forcing a resampling resetting to all the other transitions. Only the case in which each transition is competing with all the other ones seems to be appropriate for this model.

A more interesting semi-Markov *SPN* model has been discussed in [21]. In this definition, the transitions are partitioned into three classes: exclusive, competitive and concurrent. Provided that the firing time of all concurrent transitions is exponentially distributed and that 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 firing time random variables γ_k to be *PH* distributed [33], according to the following:

Definition 1 *A PHSPN is a GDT_SPN in which:*

- *To any transition $t_k \in T$ is associated a PH random variable γ_k with Cdf $G_k(x|M)$. The PH model assigned to transition t_k has ν_k stages with a single initial stage numbered stage 1 and a single final stage numbered stage ν_k.*
- *To any transition $t_k \in T$ is assigned a memory policy among the three defined alternatives: age, enabling or resampling 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 [26], and, at the same time, includes all the issues listed in *Definition 4*. The non-markovian process generated by the *GDT_SPN* is converted into a *CTMC* defined over an expanded state space. The measures pertinent to the original process can be evaluated by solving the expanded *CTMC*.

The program package *ESP* [20] realizes the *PHSPN* model according to *Definition 4*. The program allows the user to assign a specific *memory policy* to each *PN* transition so that the different execution policies can be put to work. In the *ESP* tool, the expanded *CTMC* is generated from the model specifications (the *PN* topology, and the *PH* models assigned to each timed transition). The generation algorithm is driven by the different execution policies that the user assignes to each transition.

The expanded *CTMC* is represented by an oriented 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 oriented arcs (transitions of the expanded *CTMC*). The nodes in N_H are pairs (M, W), where $M \in R(M_0)$ is a marking and W is an integer n_t-dimensional vector, 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 illlustrated in details in [20]. The marking $M^{(\ell)}$ of the original reachability set, is mapped into a macro state $\mathcal{M}^{(\ell)}$ 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*.

The cardinality n_H of the expanded state space is of the order of magnitude of 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 .

An alternative approach for the implementation of a *PHSPN* model could consist in including the *PH* models for each transition at the *PN* level, thus expanding the *PN*. This approach has been strongly discouraged in [1] on the basis of the following motivations:

- The inclusion of a subnet for each transition makes the expanded *PN* very intrigued and difficult to understand just because some primitive elements (places, transitions and arcs) are added, that only refer to the stochastic behaviour of a single transition and hiden the general structure of the model. The fascinating simplicity of the *PN* language to represent complex logical interactions between objects is destroyed.
- It seems hardly possible to automatize a procedure for generating the *PHSPN* model exapnding the basic *PN* and taking into account all the possible interaction among the introduced memory policies.

3.4 Deterministic SPN

The *Deterministic and Stochastic PN* model has been introduced in [1], with the aim of providing a technique for considering stochastic systems in which some time variables assume a constant value. In [1] only the steady state solution has been addressed. An improved algorithm for the evaluation of the steady state probabilities has been successively presented in [29]. Recently, the *DSPN* model has been revisited in [14] and [13] where the transient solution is provided.

Definition 5 - *A DSPN is a GDT_SPN in which:*

- *To any transition $t_k \in T$ is associated an exponentially distributed random variable γ_k.*
- *At most, a single deterministic transition (DET) is allowed to be enabled in each marking and the firing time of the deterministic transition is marking independent.*
- *The time elapsed in a* DET *cannot be remembered when the transition becomes disabled; the only allowed execution policy is the* race *policy with* enabling memory.

In order to prove that the marking process associated to a *DSPN* is a Markov regenerative process (*MRP*), Choi et al. [13] 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 exponential 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 exponential transitions concurrent with the given DET transition. The marking process during this time interval is a $CTMC$ called the *subordinated CTMC* of 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 $[\tau^*_i$ forms a sequence of regenerative time points, so that the marking process $X_{\mathcal{T}}(\tau)$ is a Markov regenerative process *MRP*. According to [12, 17], we define the following matrix valued functions:

$$\begin{array}{rclcrcl}
\mathbf{V}(t) & = & [V_{ij}(t)] & \text{such that} & V_{ij}(t) & = & Pr\{X_{\mathcal{T}}(t)=j \,|\, X_{\mathcal{T}}(0)=i\} \\
\mathbf{K}(t) & = & [K_{ij}(t)] & \text{''} & K_{ij}(t) & = & Pr\{M_{(1)}=j\,,\, \tau^*_1 \le t |\, X_{\mathcal{T}}(0)=i\} \\
\mathbf{E}(t) & = & [E_{ij}(t)] & \text{''} & E_{ij}(t) & = & Pr\{X_{\mathcal{M}}(t)=j\,,\, \tau^*_1 > t |\, X_{\mathcal{M}}(0)=i\}
\end{array} \tag{3}$$

Matrix $\mathbf{V}(t)$ is the transition probability matrix and provides the probability that the marking process $X_{\mathcal{T}}(t)$ is in marking j at time t given it was in i at $t=0$. The matrix $\mathbf{K}(t)$ is the *global kernel* of the *MRP* and provides the cdf of the regeneration interval given that the next regeneration marking is j, starting in marking i at $t=0$. Finally, the matrix $\mathbf{E}(t)$ 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) [17, 12]:

$$\mathbf{V}(t) = \mathbf{E}(t) + \mathbf{K} * \mathbf{V}(t) \tag{4}$$

Equation (4) can be solved numerically in the time domain. An alternative approach suggested by the authors consists in transforming the transient solution in the Laplace transform domain, and then deriving the time solution by a numerical inversion technique. The paper proposes to use the Jagerman's method [27], as adapted by Chimento and Trivedi [10].

3.5 Markov Regenerative SPN (MRSPN)

A further extension, called *Markov Regenerative SPN*, has been developed in [12] and a classification of the stochastic process underlying a *GDT_SPN* has been discussed in [15].

Definition 2 *A MRSPN is a GDT_SPN in which:*

- *To any transition $t_k \in T$ is associated an exponentially distributed random variable γ_k.*
- *At most, a single transition with generally distributed firing time is allowed to be enabled in each marking.*
- *The only allowed execution policy is the* race *policy with* enabling memory*. This means that the firing time of the generally distributed transition is sampled at the time the transition is enabled and cannot change until the transition either fires or is disabled.*
- *The firing time distribution may depend upon the marking at the time the transition is enabled.*

The convolution equation (4) still holds; however, the analytic kernel expressions depend on the specific Cdf's assumed in the model. In [12], closed form expressions are derived when the Cdf of the generally distributed transitions is the uniform distribution. A further approach, resorting to the method of supplementary variables proposed by Cox [18], is discussed in [23, 15], where the use of polyexponomial distributions is investigated.

4 Modeling Power

The considered models differ because of the different classes of distribution functions they are able to support, and by the way in which the history of the process is taken into account to condition the future evolution of the net.

Under the *enabling memory* policy the time accumulated by a *PN* transition is reset as soon as the transition is disabled, while under the *age memory* policy the time accumulates whenever the transition is enabled before firing. The *enabling memory* policy is suited to realize the interaction mechanism among tasks in service that in queueing theory or fault-tolerant systems is called a *preemptive repeat different (prd)* policy. Whenever the task in service is preempted a corresponding *PN* transition is disabled resetting the accumulated time. Hence, when the preempted task restarts its work requirement should be resampled from the same distribution [6]. On the other hand, the *age memory* policy is suited to represent an interaction mechanism usually referred to as *preemptive resume* policy: the server does not loose memory of the work already done even if the task is preempted (and the corresponding *PN* transition disabled). When the task is enabled again the execution restarts from the point it was interrupted.

The *DSPN* model, combining constant times with exponential random times, offers an innovative approach in many practical applications. The main limitation of the *DSPN* model in the present state of the art, is that only enabling memory policy is supported. Hence only systems with a service discipline of preemptive different type can be represented with this approach. Moreover, there no tools available for the automatic generation of the matrices $\mathbf{V}(t)$, $\mathbf{K}(t)$ and $\mathbf{E}(t)$, and the solution of the convolution equation is performed by means of standard packages for symbolic manipulation.

Onthe contrary, the *PHSPN* model fully supports all the defined *memory policies*, and, in particular, the *age memory* policy. The modeler is allowed to represent in a natural way *prs* interaction mechanisms. 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, 9]. A tool is conceivable [20] for supporting the generation and analysis of the model according to the requirements specified in [26]. The expansion of the state space is, of course, a cause of nonnegligible 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.

5 Example - Finite Queue with Preemption

We carry on a comparison between the modeling power and the numerical results obtained from the *DSPN* and the *PHSPN* models through the analysis of a simple finite queueing systems with different kinds of preemption. We consider, as a base example, the M/G/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 [13] for what concerns the transient behavior. We initially compare the results obtained by approximating a *DSPN* by means of a *PHSPN* and then we introduce various kinds of preemptive mechanisms.

5.1 Non Preemptive Queue

The *PN* for the M/G/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_a 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_g is the job service time; when the job is completed the customer returns in his thinking state. Transition t_i is an immediate transition modelling the start of service i.e. the transfer of the job from the queue to the server.

In [3, 13], the following assumptions were made. t_a is exponentially distributed with rate $m_1 \cdot \lambda$ being m_1 is the number of tokens in p_1 and $\lambda = 0.5$ *job/hour*. t_g is a *DET* transition modeling a constant service time of duration $d = 1.0$ *hour*.

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, called s_1, s_2 and s_3 in Figure 1b. The *PN* of

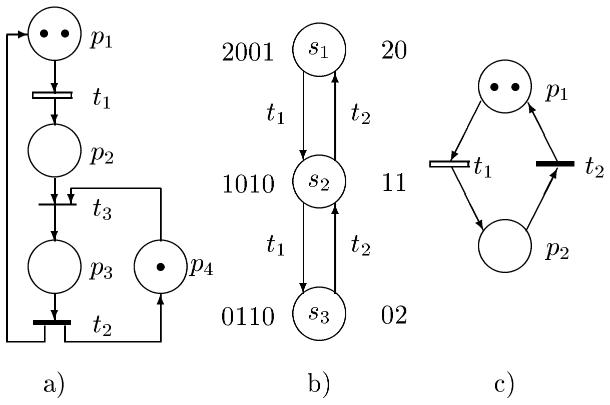

Figure 1 - a) - PN modelling the atomic operation of a M/D/1/2/2 (after [3]); b) - corresponding reduced reachability graph; c) - simplified PN.

Figure 1 is intended to show in details the atomic steps by which a customer submits a job and the job is serviced. Figure 2 shows, however, a simpler *PN* isomorphic to the one of Figure 1.

Tokens in place p_1 of Figure 2 represent customers in the thinking state, while p_2 contains the jobs in the queue (included the one under service). t_1 is the submitting time and t_2 is the service time. It is easy to verify that the above *PN* generates the same marking process $X_{\mathcal{T}}(\tau)$ of Figure 1b) when t_1 is exponential 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 3 in solid line.

Approximating the *DSPN* of Figure 2 by means of the *PHSPN* model is straightforward. Transition t_2 is assigned a *PH* distribution and an enabling memory policy, in conformity with point 3) of *Definition 5*. 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 3 we compare the results obtained from the *PHSPN* model, by reporting the behavior of the state probabilities versus time in two cases: when a) the random firing time assigned to t_2 is Erlang(5) (dashed line), and b) 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 with the Erlang(5) the local maxima and minima in the probability behavior does not appear, while the visual agreement is very satisfactory in the case of the Erlang(100).

As a further comparison, Table I shows the values for the steady state probabilities calculated from the *DSPN* model 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 to ∞ [19, 9].

5.2 Preemptive Queue

Let us assume a M/G/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 (*preemptive repeat different policy - prd)*, or is resumed (*preemptive resume policy - prs).*

5.2.1 prd policy

With reference to Figure 2, 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

Table - *Steady state probabilities*

State	*DSPN*	*PHSPN*			
		Erl(5)	*Erl(10)*	*Erl(100)*	*Erl(1000)*
TABLE I - Non preemptive policy					
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
TABLE II - Preemptive 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
TABLE III - Preemptive prs policy					
s_1	0.35015	0.36194	0.35618	0.35076	0.35021
$s_2 + s_3$	0.45429	0.44193	0.44801	0.45365	0.45423
s_4	0.19556	0.19613	0.19582	0.19558	0.19556

that the underlying process $X_{\mathcal{T}}(\tau)$ is a semi-Markov process, since each time the (generally distributed) 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 naturally generated from the current definition of *DSPN*. 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 4 describes the preemption without this anomaly. Place p_1 in Figure 4 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 (if transition t_3 fires before t_2). t_1 and t_3 are assigned the exponential submitting time and transitions t_2 and t_4 the generally distributed service time. Assigning an enabling memory policy to t_2 and t_4 the M/G/1/2/2 system with *prd* preemption is generated.

Table II compares the steady state probabilities assuming the submitting and service time distributions identical to the non preemptive case. The transient behavior is compared in Figure 5 where the results from the *DSPN* model are drawn in solid line while the results from the *PHSPN* model and with the service time given by an Erlang(5) and an Erlang(100) are drawn in dashed and in dotted line, respectively.

5.2.2 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 resumed and put to execution from the point of preemption without loss of the work previously performed.

The *prs* mechanism for the M/G/1/2/2 queue corresponds to the *PN* of Figure 4 when t_2 and t_4 is assigned an age memory policy. The preemption mechanism does not fit the rules of *Definition 5* and thus cannot be modeled in the framework of the actual implementation of the *DSPN* model. When t_2

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

and t_4 are both Erlang(100) the numerical results for the steady state probabilities are $s_1 = 0.4$, $s_2 = 0.4$, $s_3 = 0.2$. The transient behaviour is depicted in Figure 9 as *Case III*.

5.3 Preemptive Queue with Different Classes of Customers

A 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/G/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 6. 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, restarted after preemption, is resampled (*prd* case) or is resumed (*prs* case). In the *PHSPN* model, the two policies can be naturally realized by assigning to the service transitions t_2 and t_4 an enabling memory policy in the *prd* case and a age memory policy in the *prs* case.

Since in the *DSPN* only the *prd* policy is supported the transient results for *prd* policy by the different methods are reported in Figure 7 and in Table III for what concerns the steady state probability values. The effect of the different kinds of preemptions are compared for the *DSPN* in Figure 8 and for the *PHSPN* in Figure 9. (Case I refers to the non preemptive system, Case II to the preemptive system with identical customers and *prd* policy, Case III to the preemptive system with identical customers and *prs* policy, Case IV to the preemptive system with different customers and *prd* policy, and Case V to the preemptive system with different customers and *prs* policy). In the *DSPN* model only cases I, II and IV can be computed.

6 Computational complexity

Let us briefly summarize the elementary computational steps for the two considered methodologies (*DSPN* and *PHSPN*), taking into account that the *DSPN* solution requires manual and automatic manipulation, while the *PHSPN* solution is fully supported by a tool.

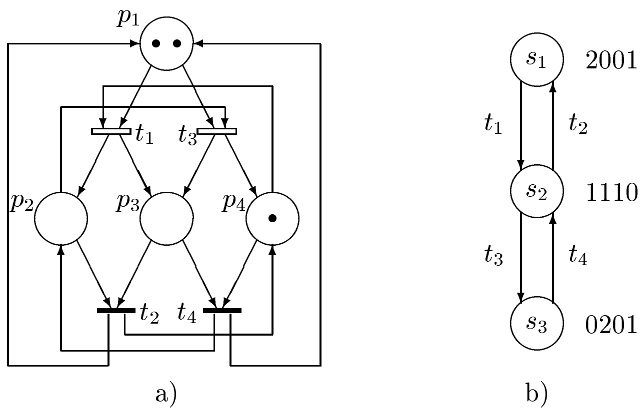

Figure 3 - Preemptive M/D/1/2/2 ith identical customers

6.1 Evaluation of *DSPN* model

According to [13], we can divide the computational method in the following steps:

1. generation of the reachability tree;
2. manual derivation of the entries of the $\mathbf{K}(t)$ and $\mathbf{E}(t)$ matrices symbolically in Laplace transform domain;
3. symbolical matrix inversion and matrix multiplication by using a standard package (e.g. MATHEMATICA) in order to obtain the $\mathbf{V}(t)$ (Equation 4) matrix in the LT domain;
4. time domain solution obtained by a numerical inversion of the entries of the $\mathbf{V}(t)$, resorting to the Jagerman's method [27]. For the sake of uniformity, this step has been implemented in MATHEMATICA language.

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 *CTMC*s subordinated to the different deterministic transitions. 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 (which is similar to the difficulty of the first step). 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 graph.

6.2 Evaluation of *PHSPN* model

For the evaluation of this model we used the ESP tool ([20]). The procedure can be devided 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

Figure 4 - Transient behavior of the state probabilities for the preemptive M/D/1/2/2 with identical customers.

transitions of order n the cardinality of the expanded *CTMC* is $2n+1$ in Case I, $2n+1$ in Case II, n^2+n+1 in Case III, $3n+1$ in Case IV, n^2+2n+1 in Case V. In this trivial example, with $n=100$ (*Erl100*) the generation of the *CTMC* takes 2 m for Caes II and V, and the whole analysis two further minutes on the same IBM RISC 6000 computer.

7 Conclusion

The development of methodologies able to accommodate non exponential random variables is of increasing interest in the analysis of stochastic systems. The paper has examined and compared *PN* based models whose definition allows the modeler to associate, to some extent, non exponential distributions to timed *PN* transitions.

The modeling power and the numerical capabilities are investigated, with particular reference to the *DSPN* model, in which a single deterministic transition can be assigned in each marking (being all the other transitions exponential), and the *PHSPN* model in which each transition can be assigned a *PH* distributed firing time.

A simple queueing system is completely analysed. 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 modeling the deterministic transition with an Erlang(5) and an error of the order of 10^{-4} by modeling the deterministic transition with an Erlang(1000). However, the use of *PH* distribution and of the *PHSPN* model offers the modeler a more flexible tool for defining a more extended interactions between the server and the job in progress.

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.

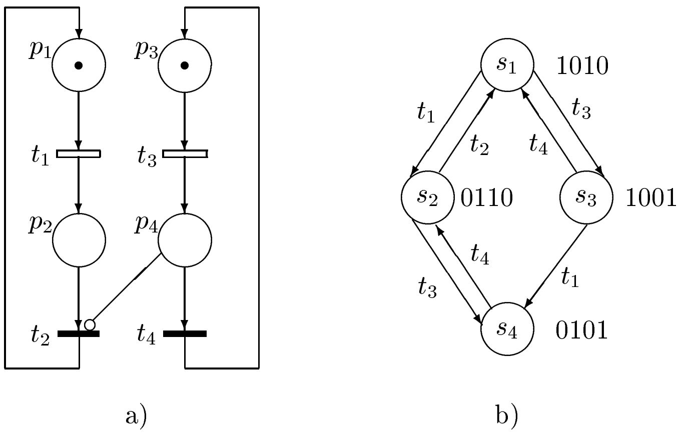

Figure 5 - Preemptive M/D/1/2/2 queue with two classes of customers

[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. Petri nets generating Markov reward models for performance/reliability analysis of degradable systems. In R. Puigjaner and D. Poitier, editors, *Modeling Techniques and Tools for Computer Performance Evaluation*, pages 353–365. Plenum Press, 1989.

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

[8] A. Bobbio and A. Cumani. ML estimation of the parameters of a PH distribution in triangular canonical form. In G. Balbo and G. Serazzi, editors, *Computer Performance Evaluation*, pages 33–46. Elsevier Science Publishers, 1992.

[9] A. Bobbio and M. Telek. Parameter estimation of Phase type distributions. Technical report, R.T. 423, Istituto Elettrotecnico Nazionale Galileo Ferraris, 20-th European Meeting of Statisticians, August 1992.

[10] P.F. Chimento and K.S. Trivedi. The completion time of programs on processors subject to failure and repair. *IEEE Transactions on Computers*, 42:1184–1194, 1993.

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

[12] Hoon Choi, V.G. Kulkarni, and K. Trivedi. Markov regenerative stochastic Petri nets. In G Iazeolla ans S.S. Lavenberg, editor, *Proceedings International Conference PERFORMANCE'93*, pages 339–356, 1993.

Figure 6 - Transient behavior of the state probabilities of the preemptive M/D/1/2/2 with *prd* policy and two classes of customers.

[13] 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.

[14] G. Ciardo, R. German, and C. Lindemann. A characterization of the stochastic process underlying a stochastic Petri net. In *Proceedings International Workshop on Petri Nets and Performance Models - PNPM93*, pages 170–179. IEEE Computer Society, 1993.

[15] 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.

[16] G. Ciardo, J. Muppala, and K.S. Trivedi. On the solution of GSPN reward models. *Performance Evaluation*, 12:237–253, 1991.

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

[18] 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.

[19] D.R. Cox and H.D. Miller. *The theory of stochastic processes.* Chapman and Hall, London, 1965.

[20] 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.

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

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

[23] R. German and C. Lindemann. Analysis of SPN by the method of supplementary variables. In G Iazeolla ans S.S. Lavenberg, editor, *Proceedings International Conference PERFORMANCE'93*, pages 320–338, 1993.

Figure 7 - Comparison of the state probabilities computed from the *DSPN* model with different kinds of preemption.

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

[25] 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.

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

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

[28] 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.

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

[30] J.F. Meyer, A. Movaghar, and W.H. Sanders. Stochastic activity networks: structure, behavior and application. In *Proceedings International Workshop Timed Petri Nets*, pages 106–115, Torino (Italy), 1985. IEEE Computer Society Press no. 674.

[31] M.K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers*, C-31:913–917, 1982.

[32] 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.

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

Figure 8 - Comparison of the state probabilities computed from the *PHSPN* model with different kinds of preemption.

[34] 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.