Recent Developments in Non-Markovian Stochastic Petri Nets

Andrea Bobbio[1], Antonio Puliafito[2], Miklós Telek[3], Kishor S. Trivedi[4]

[1] Dipartimento di Informatica, Università di Torino
10149 Torino - Italy
e-mail: bobbio@di.unito.it

[2] Istituto di Informatica, Università di Catania
Viale A. Doria 6, 95025 Catania - Italy
e-mail: ap@iit.unict.it

[3] Department of Telecommunications
Technical University of Budapest, 1521 Budapest - Hungary
e-mail: telek@hit.bme.hu

[4] Center for Advanced Comp. & Comm.
Department of Electrical and Computer Engineering, Duke University
Durham, NC 27708-0291 - USA
e-mail: kst@ee.duke.edu

Abstract

Analytical modeling plays a crucial role in the analysis and design of computer systems. Stochastic Petri Nets represent a powerful paradigm, widely used for such modeling in the context of dependability, performance and performability. Many structural and stochastic extensions have been proposed in recent years to increase their modeling power, or their capability to handle large systems. This paper reviews recent developments by providing the theoretical background and the possible areas of application. Markovian Petri nets are first considered together with very well established extensions known as Generalized Stochastic Petri nets and Stochastic Reward Nets. Key ideas for coping with large state spaces are then discussed. The challenging area

of non-Markovian Petri nets is considered, and the related analysis techniques are surveyed together with the detailed elaboration of an example. Finally new models based on Continuous or Fluid Stochastic Petri Nets are briefly discussed.

1 Introduction

Analytical evaluation of computer/communication systems is increasingly becoming an integral part of the whole design process. Many diverse model specification techniques have been proposed. Petri net models have gained a widespread acceptance [103, 100] since they provide a graphical language that can be rather concise in its specification, provide a natural way to represent complex logical interactions among parts or activities in a system and are closer to a designer's intuition about what a model should look like. Original Petri nets did not carry any notion of time. In order to make the technique useful for quantitative analysis, a variety of timing extensions have been proposed in the literature.

The distinguishing features of the timing extensions are whether the duration of the events is modeled by deterministic or random variables, and whether the time is associated with places, transitions or tokens. Petri nets (*PN*) in which the timing is stochastic are referred to as Stochastic *PN* (*SPN*), and the most common assumption is that time is assigned to the duration of events represented by the transitions. The time evolution of a *SPN* is captured by a stochastic process, referred to as its *Marking Process*.

SPN can be used to automatically generate the underlying marking process, which can then be analyzed to yield results in terms of the original Petri net model. This is a case where the *user-level representation* of a system is translated into an *analytic representation* [72]. The analytic representation is processed and the results are cast back to the user-level representation. The most updated and valuable source of references for the theoretical developments and the possible application areas of models based on stochastic *PN* is the series of international workshops known as *Petri Nets and Performance Models - PNPM*. This series was initiated in Torino (Italy) in 1985, then moved to the USA, Japan, Australia and France. The sevents edition was held in Saint-Malo (France) in 1997.

The most common assumption, in the literature, is to assign to the *PN* transitions an exponentially distributed firing time [94, 95, 101], so that the resulting marking process is a Continuous Time Markov Chain *(CTMC)*. Almost all the *PN*-based tools are based on this assumption.

In principle, simple and tractable equations can be derived for both transient and steady-state analysis of *CTMC*s. But practical limitations arise from the fact that the cardinality of the state space grows much faster than the number of components in the system being modeled. One line of research has been devoted to dealing with large system models resorting to distributed algorithms, aggregation, hierarchical composition or approximation techniques.

The use of exponentially distributed firing time has been regarded as a restriction in the application of *PN*-based models. Indeed, there are many phenomena whose times to occurrence are not exponentially distributed. The hypothesis of exponential distributions, in those cases, allows the construction of models which can give a more qualitative rather

than quantitative analysis of real systems. The existence of deterministic or other non-exponentially distributed events, such as timer expiration, propagation delay, transmission of fixed length packets, hard deadlines in real-time systems etc., give rise to stochastic models that are non-Markovian in nature [92].

In recent years, a considerable effort has been devoted to enrich the *PN* formalism in order to deal with generally distributed delays [42, 19]. However, the inclusion of non-exponential distributions destroys the memoryless property of the associated marking process, and further specification is needed at the *PN* level in order to uniquely define how the marking process is conditioned on the past history.

In this paper, we review the main structural and stochastic extensions of *PN*s, by providing an updated treatment of the theoretical background and the possible areas of application.

The paper is organized as follows. Section 2 defines the basic *PN* model, and introduces the most common structural extensions that are an integral part of the standard definition in many software packages. Section 3 shows how a Petri net can be augmented with stochastic timing associated with the transitions. When all the firing times are exponentially distributed the marking process is a *CTMC*. This assumption is by far the most common in practice and is reviewed in Section 4 together with a useful extension, known as *GSPN* [2], which divides the transitions into two classes: exponentially timed and immediate. The measures that can be obtained from a Markovian model are recalled, and it is explicitly shown how they cast into a *PN* model. Stochastic reward nets (*SRN*) are introduced in Section 5, and it is shown how useful measures at the *SPN* level can be compactly obtained by a suitable definition of the reward structure superimposed on the *SPN*. Some directions of research to deal with very large Markovian models, generated by an *SPN*, are summarized in Section 6. Non-Markovian *SPN*s are dealt with in Section 7. In particular, three approaches are discussed: the first one is based on the Markov regenerative theory, the second one is based on the use of supplementary variables, and the third one is based on state expansion techniques. A fully developed example is reported in Section 8. A possible new direction of research is based on *SPN*s that generate a partially discrete and partially continuous state space [5]. These models are, sometimes, referred to as fluid-*SPN*s and are considered in Section 9. Section 10 is the concluding section.

2 Definition of the basic Petri Net Model

Formally, a marked *PN* [103] is a tuple $PN = (P, T, I, O, M)$, 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 and O are the input and the output 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.

- $M = \{m_1, m_2, \ldots, m_{np}\}$ is the marking of the *PN*. The generic entry m_i is the number of tokens (drawn as black dots) in place p_i, in marking M. The initial marking is M_0.

Input and output arcs have an arrowhead on their destination. A transition is enabled in a marking if each of its input places contains at least as many tokens as the multiplicity of the input function I. An enabled transition fires by removing as many tokens as the multiplicity of the input function I from each input place, and adding as many tokens as the multiplicity of the output function O to each output place.

A marking M' is said to be *directly 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 rule.

*PN*s can be used to capture the behavior of many real-world situations including sequencing, synchronization, concurrency, and conflict. The enabling of a transition corresponds to the *starting* of an activity, while the firing corresponds to the *completion* of an activity. When the firing of a transition causes a previously enabled transition to become disabled, it means that the corresponding activity was *interrupted* before being completed.

2.1 Structural Extensions

Various structural extensions have been proposed in the past to increase either the class of problems that can be represented or the ability and the ease with which real systems can be modeled. In [39], Ciardo defines the *modeling power* as the ability of a *PN* formalism to represent classes of problems. He also defines *modeling convenience* as the practical ability to represent a given behavior in a simpler, more compact or more natural way. *Decision power* is defined to be the set of properties that can be analyzed. Increasing the modeling power decreases the decision power. Thus each possible extension to the basic formalism requires an in depth evaluation of its effect upon modeling and decision power [103].

Extensions which only affect modeling convenience can be removed by using basic constructs, so they can usually be adopted without introducing any further analytical complexity. Some extensions have proven so effective that they are now considered part of the standard *PN* definition. They are:

- inhibitor arcs,
- transition priorities,
- marking-dependent arc multiplicity.

Inhibitor arcs connect a place to a transition and are drawn with a small circle on their destination. An inhibitor arc from a place p_i to a transition t_k disables t_k when p_i is not empty. It is possible to use the arc multiplicity extension together with inhibitor arcs. In this case, a transition t_k is disabled whenever place p_i contains at least as many tokens as the multiplicity of the inhibitor arc. The number of tokens in an inhibitor input place is not affected by a firing operation.

Priorities are integer numbers assigned to the transitions. A transition is enabled in a marking if and only if no higher priority transitions are enabled. If this extension is introduced, some markings of the original *PN* may no longer be reachable.

Marking-dependent arc multiplicity was introduced in [40, 41] with the intent to model situations in which the number of tokens to be transferred along the arcs (or to enable a transition) depends upon the system state. Arcs with marking dependent multiplicity are indicated by a 'Z' on the arc, and allow simpler and more compact *PN*s than would otherwise be possible without this construct. In many practical problems, their use can dramatically reduce the complexity of the *PN* model.

As an example of an efficient and convenient use of the above introduced structural extensions, consider the *PN* model for an ISDN channel shown in Figure 1 [110]:

- Voice and data packets arrivals are modeled through transitions $Tarrival - voice$ and $Tarrival - data$, respectively;
- Voice and data processing times are modeled through transitions $Tser - voice$ and $Tser - data$;
- The transmitter contains a buffer (place *data*) to store a maximum of k data packets. This is modeled by the inhibitor arc from place *data* to transition $Tarrival - data$ with multiplicity k. When k tokens are resident in place *data*, transition $Tarrival - data$ is inhibited and cannot fire.
- A voice packet can enter the channel (place *voice*) only if there are no packets (voice or data) waiting to be transmitted. This is modeled by the two inhibitor arcs to transition $Tarrival - voice$.
- If a voice transmission is in progress, data packets cannot be serviced, but are buffered. This priority mechanism is modeled by the inhibitor arc from place *voice* to transition $Tser - data$;
- The data buffer can eventually be flushed, if some asynchronous event occurs (transition $flush$). This is modeled by the marking-dependent arc multiplicity between place *data* and transition $remove$. The arc removes as many tokens as resident in place *data*. This flushing action might be also obtained without resorting to the special construct of the marking-dependent arc multiplicity, but at the cost of a much more complex *PN*.

Other extensions are possible to increase the modeling convenience. They are, usually, in the form of *guard*s or *enabling function*s [44], where besides the standard enabling rules, a transition is enabled if the value of a boolean function related to various conditions on the *PN* evaluates to true.

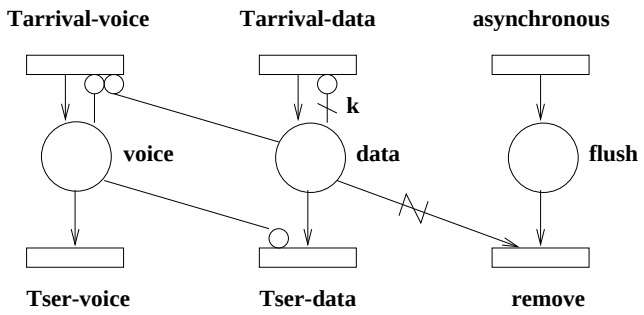

Figure 1: Using PN structural extension to model an ISDN channel

3 Stochastic Petri Nets

The most common way to include time into a *PN*, is to associate a duration with the activities that induce state changes, hence with the transitions. The duration of each activity is represented by a non-negative random variable with a known *cdf*. Let $\Gamma = (\gamma_1, \gamma_2, \ldots, \gamma_{nt})$ be the set of the n_t random variables associated with the n_t transitions, and $G = (G_1(t), G_2(t), \ldots, G_{nt}(t))$ be the set of their *cdf*'s.

When a waiting time γ_k is associated with a transition t_k, the transition becomes enabled according to the rules of the untimed *PN*, but it can fire only after a time equal to γ_k has elapsed. The time between the enabling and the firing is referred to as the *firing time*. Let $\{\mathcal{M}(t),\ t \geq 0\}$ be the marking process, i.e., $\mathcal{M}(t)$ represents the marking reached by the *PN* at time t.

In the following, we restrict our analysis to *SPN*s in which the random firing times have continuous *cdf* with infinite support $(0, \infty]$. With this assumption, the marking process $\mathcal{M}(t)$ is a right-continuous, piecewise constant, continuous-time, discrete-state stochastic process whose state space is isomorphic to the reachability graph of the untimed *PN*. Intrigued semantic interpretations related to the possibility of contemporary firings are avoided [96, 75, 41, 48].

Given a marking in which more than one transition is enabled (with the same priority level if priority is used), the *firing policy* determines the transition that will fire next. Two possible alternatives have been discussed in [1]:

i) Under the *race policy*, the transition whose firing time elapses first is assumed to be the one that will fire next,

ii) Under the *preselection policy*, the next transition to fire is chosen according to an externally specified probability mass function independent of their firing times.

By far the most common firing policy for timed transitions is the *race policy*. Preselection policy is commonly used for *immediate* transitions, introduced for the first time in Markovian *SPN* in [2].

Once the *firing policy* is defined, the *execution policy* must be specified. The *execution policy* consists of a set of specifications for uniquely defining the stochastic process $\{\mathcal{M}(t)\}$ underlying an *SPN*. Two elements characterize the *execution policy*: a criterion to keep memory of the past history of the process (the *memory policy*), and an indicator of the resampling status of the firing time. The *memory policy* defines how the process is conditioned upon the past. An *age variable* a_g associated with the timed transition t_g keeps track of the time for which the transition has been enabled. A timed transition fires as soon as the memory variable a_g reaches the value of the firing time γ_g. The *activity period* of a transition is the period of time during which its age variable is not 0.

The random firing time γ_g of a transition t_g can be sampled at a time instant prior to the beginning of an activity period. To keep track of the resampling condition of the random firing time associated with a timed transition, we assign to each timed transition t_g a binary indicator variable ι_g that is equal to 1 when the firing time is to be sampled and equal to 0 when the firing time is to be not sampled. We refer to ι_g as the *resampling indicator variable*. Hence, in general, the (continuous) memory of a transition t_g is captured by the tuple (a_g, ι_g). At any time epoch t, transition t_g has memory (its firing process depends on the past) if either a_g or ι_g is different from zero.

At the entrance in a marking, the remaining firing time ($rft_g = \gamma_g - a_g$) is computed for each enabled transition given its currently sampled firing time γ_g and the age variable a_g. According to the race policy, the next marking is determined by the minimal of the rft's.

Now, the following different execution policies are defined. A timed transition t_g can be:

- *Preemptive repeat different (prd)*:
  If both the age variable a_g and the resampling indicator ι_g are reset each time t_g is disabled or it fires.

- *Preemptive resume (prs)*:
  If both the age variable a_g and the resampling indicator ι_g are reset only when t_g fires.

- *Preemptive repeat identical (pri)*:
  If the age variable a_g is reset each time t_g is disabled or fires but the resampling indicator ι_g is reset only when t_g fires.

Figure 2 gives a pictorial description of these preemption policies for a single transition t_g. In the figure, the time instants marked with *E*, *D* and *F* indicate the enabling, disabling and firing time points of t_g, respectively. Each preemption policy is illustrated via the evolution of the age variable a_g associated with the transition t_g and of its remaining firing time ($rft_g = \gamma_g - a_g$). The horizontal lines below the diagrams indicate the periods of time when $\iota_g = 1$.

Transition t_g is prd - Each time a *prd* transition is disabled or it fires, its memory variable a_g is reset and its indicator resampling variable ι_g is set to 0 (the firing time must be resampled from the same distribution when t_g becomes reenabled). With reference to Figure 2a, t_g is

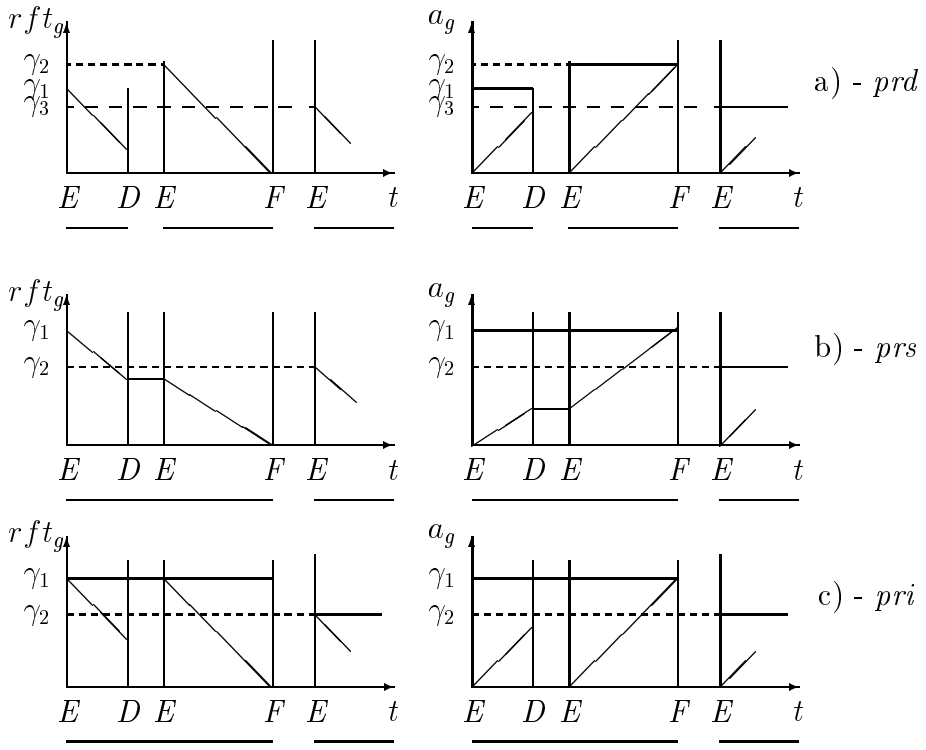

Figure 2: Pictorial representation of different firing time sampling policies

enabled for the first time at $t = 0$: its memory variable a_g starts increasing linearly, ι_g is set to 1 and the firing time is sampled from its distribution to a value, say, γ_1. At time D, t_g is disabled and the memory is reset ($a_g = 0$, $\iota_g = 0$). At the next enabling time instant E, a_g restarts from zero, ι_g is set to 1 and the firing time is resampled from the same distribution assuming a different value, say γ_2. When t_g fires (point F) both a_g and ι_g are reset. At the successive enabling point E, a_g restarts and the firing time is resampled (γ_3). Thus, a *prd* transition looses its memory at any D and F points. The memory of the transition is confined to the periods of time in which t_g is continuously enabled.

Transition t_g is prs - With reference to Figure 2b, when t_g is disabled (at point D), its associated age variable a_g is not reset but maintains its constant value until t_g is reenabled and $\iota_g = 1$. At the successive enabling point E, a_g restarts from the previously retained value. When t_g fires, both a_g and ι_g are reset so that the firing time must be resampled at the successive enabling point (γ_2). The memory of t_g is reset only when the transition fires.

Transition t_g is pri - Under this policy (Figure 2c), each time t_g is disabled, its age variable is reset, but ι_g remains equal to 1, and the firing time value γ_1 remains active, so that in

the next enabling period an identical firing will result. In Figure 2c, the same value (γ_1) is maintained over different enabling periods up to the firing of t_g. Only when t_g fires both a_g and ι_g are reset and the firing time is resampled (γ_2). Hence, also in this case, the memory is lost only upon firing of t_g.

If the firing time is exponentially distributed both the *prd* and *prs* policies behave in the same way and can be omitted from specification. However, the *pri* policy does not enjoy the memoryless property [15]. Thus, the marking process of an *SPN* with only exponentially distributed firing times is not a *CTMC* if at least a single non-exclusively enabled transition exists with assigned *pri* policy.

The preemption policies of transitions that can not be preempted before firing do not affect the stochastic behaviour of *SPN*s. There are subclasses of *SPN*s in which none of the transitions can be preempted before firing, e.g. *stochastic decision free Petri nets* [11], *marked graphs* [31], *event graphs* [8], *free choice nets* [59].

If the firing time is deterministic, both the *prd* and *pri* policy behave in the same way (indeed, resampling a deterministic variable provides always an identical value).

The memory of the global marking process is considered as the superposition of the individual memories of the transitions. In general, the marking process $\{\mathcal{M}(t)\}$ underlying a *SPN* is not analytically tractable unless some restrictions are imposed [42, 19]. Note that, a simulation approach for the *prd* and the *prs* cases, based on assumptions very similar to the one stated in the present section, has been described in [65, 66].

In Figure 3, the correspondence between the usual restrictions of *SPN*s and the corresponding classes of analytically tractable stochastic processes is reported. In the following sections, the various entries in Figure 3 are characterized and their solution techniques are briefly summarized.

4 Markovian SPN

When all the random variables γ_k associated with the *PN* transitions are exponentially distributed, and the execution policy is not *pri*, the dynamic behavior of the *PN* is mapped into a *CTMC*, with state space isomorphic to the reachability graph of the untimed *PN*. This restriction is the most popular in the literature [101, 94, 95, 60] and is usually referred to simply as *SPN*. A number of tools are built on this assumption [33, 44, 51, 87]. In order to completely specify the model, the set $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_{nt})$ of the n_t firing rates assigned to the n_t transitions should be given in addition. A usual convention, in the graphical representation, is to indicate transitions with exponentially distributed firing times by means of empty rectangles, and transitions with non-exponentially distributed firing times by means of filled rectangles.

Modeling real systems often involves the presence of activities or actions, whose duration is short, or even negligible, with respect to the time scale of the problem [3]. Hence, it is desirable to associate an exponentially distributed firing time only with those transitions

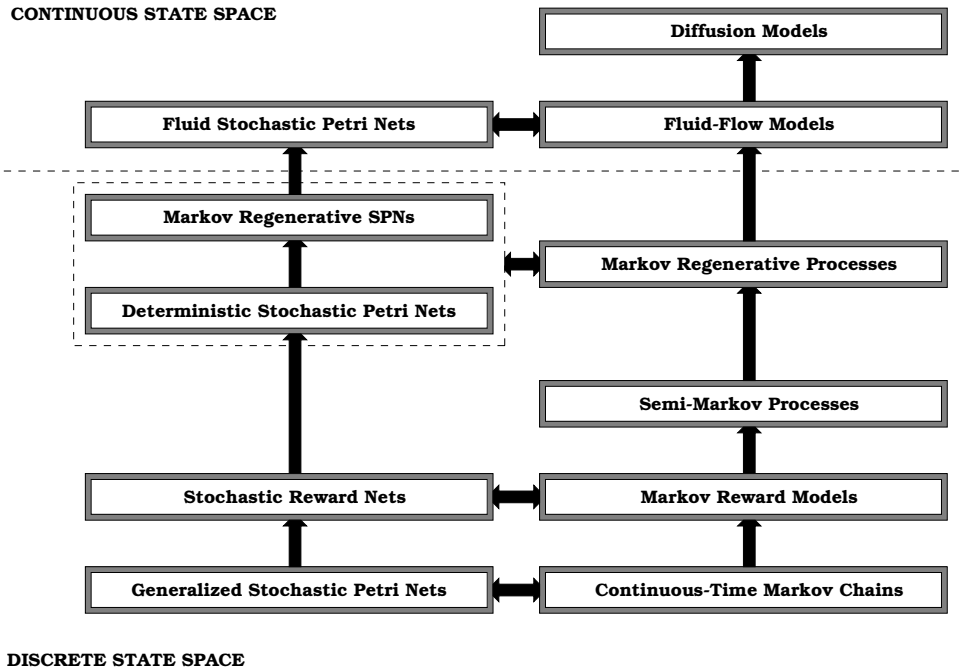

Figure 3: Correspondence between *SPN* models and stochastic processes.

which are believed to have the largest impact on the system operation. The starting assumption in the *GSPN* model [2] is that transitions are partitioned into two different classes: *immediate* transitions and *timed* transitions. Immediate transitions fire in zero time once they are enabled and have priority over timed transitions. Timed transitions fire after an exponentially distributed firing time (these will be called EXP transitions below). In the graphical representation of *GSPN*, immediate transitions are drawn as thin bars.

Markings enabling immediate transitions are passed through in zero time and are called *vanishing* states. Markings enabling no immediate transitions are called *tangible* states. Since the process spends zero time in the vanishing states, they do not contribute to the dynamic behavior of the system so that a procedure can be envisaged to eliminate them from the final Markov chain [2]. With the partition of *PN*-transitions into a timed and an immediate class, a greater flexibility at the modeling level is achieved without increasing the dimensions of the final tangible state space from which the desired measures are computed.

The *Extended Reachability Graph* (ERG) of a *GSPN* comprises both tangible and vanishing states. Elimination of the vanishing states results in a *reduced reachability graph* which is isomorphic to the *CTMC*.

Given a vanishing marking, m_b (directly reachable from a tangible marking m_a), and the set of tangible markings S, reached from m_b passing through a sequence of vanishing markings only, it is possible to evaluate the probability of the next tangible marking after m_b over S. Note that m_a may belong to S. For a discussion about the role of immediate transitions in *GSPN* and the evaluation of this probability see [35].

The vanishing marking m_b and the ones reachable from m_b by the firing of immediate

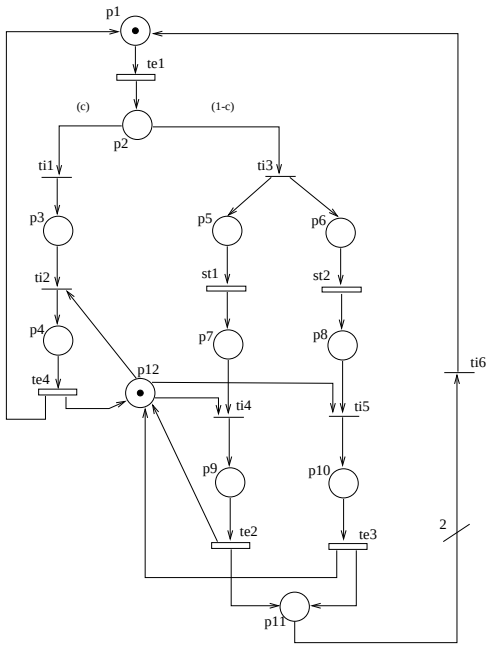

Figure 4: *GSPN* model of a Client-Server system.

transitions can only be eliminated by introducing arcs directly connecting m_a to $m_c \in S, m_c \neq m_a$, and by suitably modifying the firing rate associated with the generic transition t_k enabled in m_a [2]. Let $\mathcal{R}_0$ be the reduced reachability graph of a Markovian *SPN* and N its cardinality. The infinitesimal generator of the underlying *CTMC* is a $N \times N$ matrix $\mathbf{Q} = [Q_{ij}]$.

In [2, 3] matrix equations are provided, for calculating the weights over all the sequences of immediate transitions and for accounting for these weights automatically into the tangible restriction of the reachability graph.

As an example of the generation of the extended and reduced reachability graph, consider a system based on a client-server paradigm, whose *PN* model is shown in Figure 4 [105]. Transitions labeled te_k or st_k are EXP (empty rectangles), and transitions labeled ti_k are immediate (thin bars). The system being examined is made up of a client requesting a service (transition te_1) which can be supplied with probability $(1-c)$ (transition ti_3) by two servers working in parallel, and with probability c (transition ti_1) by accessing a resource (place p_{12}) shared by the two servers. In the case of firing of ti_3, a request forwarded by the client is split (*fork*) into two subrequests each addressed to a different server (places p_5 and p_6). The two servers are characterized by an exponentially distributed service time modeled by transitions st_1 and st_2, respectively. It is assumed, in the definition of the model, that a generic I/O request is concluded when all the servers have served the subrequests they are assigned (fork-join synchronization). When a server has processed its subrequest, it accesses the shared resource to record its processing results (transitions te_2 and te_3). When both servers have accessed the shared resource and the information requested is thus reconstructed and

	$p1$	$p2$	$p3$	$p4$	$p5$	$p6$	$p7$	$p8$	$p9$	$p10$	$p11$	$p12$
$m1$	•											•
$m2*$		•										•
$m3*$			•									•
$m4$				•								
$m5$					•	•						•
$m6*$						•	•					•
$m7*$					•			•				•
$m8$						•			•			
$m9$						•					•	•
$m10$								•	•			
$m11*$								•			•	•
$m12$										•	•	
$m13*$											•	•
$m14$					•					•		
$m15$					•						•	•
$m16*$							•				•	•
$m17$									•		•	
$m18$							•			•		

Table 1: Reachable markings for the *GSPN* of Figure 4

available, a join operation is performed and the processed result is returned to the client (transition ti_6). On the other hand, with probability c the information requested by the client is already available in the shared resource, so that the request is met by accessing the resource, retrieving the data and communicating it to the client (transitions ti_2 and te_4).

The reachability graph, generated from the initial token distribution depicted in Figure 4, is represented in Figure 5. Table 1 reports the distribution of the tokens in the reachable markings. It is easily checked from Table 1 that the markings $m2$, $m3$, $m6$, $m7$, $m11$, $m13$ and $m16$ are vanishing (they are shadowed in Figure 5) and can be eliminated. The reduced reachability graph, defined over the tangible markings only, is shown in Figure 6. Once the reduced reachability graph is obtained, the generator matrix for the underlying *CTMC* can be constructed.

Let $\Pi(t)$ be the N-dimensional state probability vector, whose generic entry $\pi_i(t)$ is the probability of being in state i ($i = 1, 2, ..., N$) at time t in the associated *CTMC*. $\Pi(t)$ is the solution of the standard linear differential equation:

$$\frac{d\,\Pi(t)}{d\,t} = \Pi(t)\,\mathbf{Q} \tag{1}$$

with initial condition $\Pi(0) = [1, 0, 0, ..., 0]$. If the steady-state probability vector $\Pi = \lim_{t\to\infty} \Pi(t)$ of the *CTMC* exists, it can be calculated from the equation:

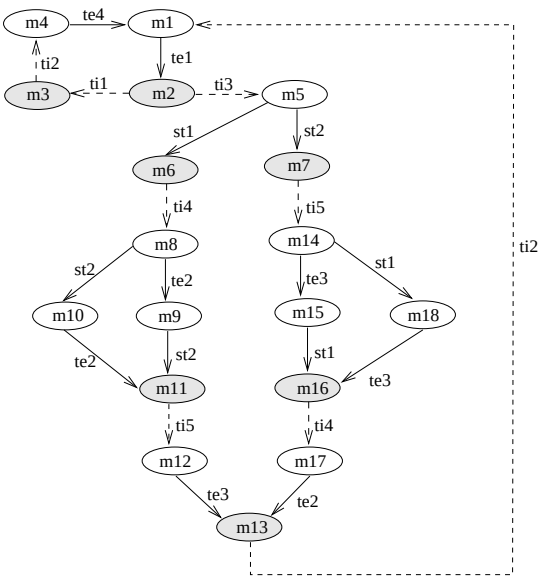

Figure 5: Reachability graph of Client Server system.

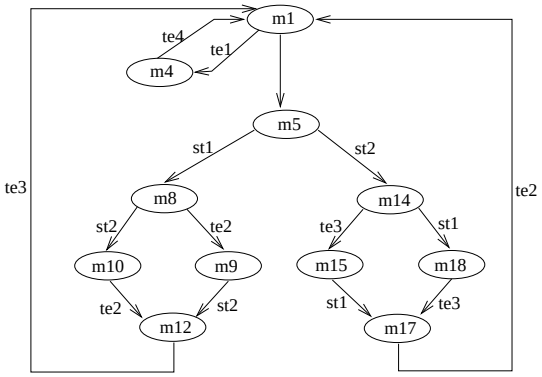

Figure 6: Reduced reachability graph of Client Server system.

$$\Pi \mathbf{Q} = 0 \qquad \text{with} \qquad \sum_{i=1}^{N} \pi_i = 1 \tag{2}$$

The numerical techniques for the solution of Equations (1) and (2) are outside the scope of this paper. Recommended references are [116, 108].

Since some of the output measures depend on the integrals of the probabilities rather than on the probabilities themselves [64], it is necessary to provide the appropriate computation of the integrals of the state probabilities. Let

$$L_i(t) = \int_0^t \pi_i(z) d\, z$$

be the expected time that the *CTMC* stays in state i during the interval $(0, t)$; let $L(t)$ denote the N-dimensional row vector consisting of the elements $L_i(t)$. Integrating both sides of equation (1), the following relation is obtained:

$$\frac{dL(t)}{dt} = L(t)\mathbf{Q} + \Pi(0) \tag{3}$$

Solution of Equation (3) can be obtained utilizing the same techniques available for Equations (1) [108, 109] and with a very little additional overhead if the two set of equations are suitably solved in parallel using the same data structures.

4.1 Measures at the net level

A noticeable property of the time dependent representation of the system behavior through *SPN*s, is that they allow the user to define in a simple and natural way a large number of different measures related to the performance and reliability of the system. In order to exploit this property, the input language must be structured for providing a friendly environment for the specification of the required output measures.

The stochastic behavior of a Markovian-*SPN* is determined by calculating the $\Pi(t)$, Π and $L(t)$ vectors over the reduced reachability set $\mathcal{R}_0$. However, the final output measures should be defined at the *PN* level as a function of its primitive elements. The following subsections provide a practical outline as how to relate the computed probabilities at the *CTMC* level with useful measures at the *PN* level.

4.1.1 Probability of a given condition on the SPN

By means of logical or algebraic functions of the number of tokens in the *PN* places, a particular condition C (e.g., no tokens in a given place) can be specified and the subset of states $S \in \mathcal{R}_0$ can be identified for which the condition is true. The output measure

$$C_S(t) = Prob\,\{condition\ C\ is\ true\ at\ time\ t\,\}$$

is given by:

$$C_S(t) = \sum_{s \in S} \pi_s(t) \tag{4}$$

where $\pi_s(t)$ is the probability of being in state s at time t. For instance, if S is the set of operational states, $C_S(t)$ in (4) is the usual definition of availability.

A very useful case arises when the measure to calculate is the transient probability that the condition is satisfied for the first time. By using a standard device in the analysis of stochastic processes, the states $s \in S$ can be made absorbing, and the requested quantity is evaluated from (4) by stopping the process when entering S. In this way the above equation can be used to calculate the system reliability.

4.1.2 Time spent in a marking

Let $S \in \mathcal{R}_0$ be the subset of markings in which a particular condition is fulfilled. The expected time $\psi_S(t)$ spent in the markings $s \in S$ during the interval $(0, t)$ is given by:

$$\psi_S(t) = \sum_{s \in S} \int_0^t \pi_s(z)\, dz = \sum_{s \in S} L_s(t) \tag{5}$$

Moreover, it is well known from the theory of irreducible Markov chains that as t approaches infinity the proportion of the time spent in states $s \in S$ equals the asymptotic probability:

$$\psi_S = \sum_{s \in S} \pi_s = \lim_{t \to \infty} \frac{\psi_S(t)}{t} \tag{6}$$

$\psi_S(t)/t$ can represent the utilization factor in the interval $(0, t)$, and ψ_S is the expected steady-state utilization factor. For example, if S is the set of states in which a server is idle, $\psi_S(t)/t$ is the fraction of idle time in $(0, t)$, and ψ_S is the expected asymptotic idle time.

4.1.3 Mean first passage time

Given that $C_S(t)$, as calculated in (4), is the probability of having entered subset S before t for the first time, the mean first passage time μ_S, can be calculated as:

$$\mu_S = \int_0^\infty [1 - C_S(z)]\, dz \tag{7}$$

The above formula requires the transient analysis to be extended over long intervals. Of course, in this case, other well known direct techniques for calculating mean first passage times in a *CTMC* can be more effective [44].

4.1.4 Distribution of tokens in a place

The *cdf* of the number of tokens in place p_i of the *SPN* at time t is a staircase function in which the amplitude of the k-th step is obtained by summing up the probabilities of all the states in $\mathcal{R}_0$ containing k tokens ($k = 0, 1, 2, \ldots$) in p_i at time t. The probability mass function $f_i(k,t)$ is the amplitude of the k-th step. The expected value of the number of tokens in place p_i at time t is:

$$\mathbb{E}\left[m_i(t)\right] = \sum_{k=0}^{\infty} k\, f_i(k,t) \tag{8}$$

As an example, if place p_i represents identical units queueing up for a common resource the above quantity gives the expected value of the number of units in the queue versus time. In reliability analysis a very interesting case arises when the tokens in place p_i represent the number of failed components.

4.1.5 Expected number of firings of a PN-transition

Given an interval $(0,t)$, the expected number of firings indicates how many times, on the average, an event modeled by a *PN* transition has occurred in that interval. Let t_k be a generic *PN* transition, and let S be the subset of $\mathcal{R}_0$ which includes all the markings $s \in S$ enabling t_k. The expected number of firings of t_k in $(0,t)$ is given by:

$$\eta_k(t) = \sum_{s \in S} \lambda_k(s) \int_0^t \pi_s(z)\, dz = \sum_{s \in S} \lambda_k(s)\, L_s(t) \tag{9}$$

where $\lambda_k(s)$ is the firing rate of t_k in marking s.
In steady-state, the expected number of firings per unit of time becomes:

$$\eta_k = \sum_{s \in S} \pi_s\, \lambda_k(s) \tag{10}$$

This quantity is very important since it represents the throughput associated with the given transition. If transition t_k represents the completion of a service in a queueing system, $\eta_k(t)$ is the expected number of services completed in time $(0,t)$ and η_k is the expected steady-state throughput.

If transition t_k indicates failure (repair) of a component, $\eta_k(t)$ provides the mean number of failures (repairs) of that component in $(0,t)$.

5 Stochastic Reward Nets

Stochastic Reward Nets (SRN) introduce a new extension into Markovian-*SPN*s consisting in the possibility of associating reward rates to the markings. The reward rates are specified at the *PN* level as a function of its primitives (like the number of tokens in a place or the rate of a transition). The underlying *CTMC* is then transformed into a *Markov reward model* thus

permitting the evaluation of performability measures [107]. The tools, which implements this extension [45, 51], allow the reward structure superimposed on the reachability graph to be generated automatically and easily provide dependability, performance and performability measures.

The reward definition, used in the sequel, is called *rate-based*, to indicate that the system produces reward at rate $r(i)$ for all the time it remains in state $i \in \mathcal{R}_0$. *Impulse-based* reward models [51] can also be implemented: a reward function r_{ij} is associated to each transition from state $i \in \mathcal{R}_0$ to $j \in \mathcal{R}_0$. Each time a transition from i to j occurs, the cumulative reward of the system instantaneously increases by r_{ij}. In general, several combinations of the different reward functions can be specified in the same model.

5.1 Measures at the net level

We assume time-independent rate-based reward models, and we show how all the *PN*-based measures, introduced in Section 4.1, can be expressed in a very compact form, just properly particularizing the various reward rates.

Let r_i $(i = 1, 2, ..., N)$ be the reward rate of the process in state i and let $\Pi(t)$, Π and $L(t)$ be the transient, the steady-state and the integral probability vectors calculated from the underlying *CTMC* (Equations 1-3). Let $X(t)$ denote the instantaneous reward rate at time t and $Y(t)$ be the reward accumulated during $(0, t)$. The following measures are conveniently defined.

5.1.1 Expected instantaneous reward rate

The expected instantaneous reward rate at time t is computed as:

$$\mathbb{E}[X(t)] = \sum_{i=1}^{N} r_i \pi_i(t) \tag{11}$$

and in steady-state:

$$\mathbb{E}[X] = \sum_{i=1}^{N} r_i \pi_i \tag{12}$$

The complexity of solving Equations (11) and (12) is the same as that of solving the standard Markov equations (1) and (2). It is easily recognized that Equation (4) can be expressed in the form (11) if an appropriate binary reward rate is assigned to the state space $\mathcal{R}_0$. In particular a reward rate $r_s = 1$ is assigned to states $s \in S$ and a reward rate $r_i = 0$ otherwise.

Furthermore, Equation (8) can be derived directly from (11) by assigning to r_i the value of the number of tokens in place i, and the throughput Equation (10) by assigning to r_i the value of the firing rate of the transition of interest.

5.1.2 Expected accumulated reward

The expected accumulated reward at time t is given by:

$$\mathbb{E}[Y(t)] = \mathbb{E}\left[\int_0^t X(z)dz\right] = \sum_{i=1}^{N} r_i L_i(t) \tag{13}$$

A sometimes useful related measure is the time averaged accumulated reward $\mathbb{E}[W(t)] = \mathbb{E}[Y(t)]/t$, which can also be seen as the average rate according to which the reward is accumulated from 0 to t.

It is easily recognized how Equations (5) and (9) can be expressed in the form (13) by a suitable assignment of the reward rates.

5.1.3 Distribution of cumulative measures

Let $F(t,y) = Prob\{Y(t) \leq y\}$ denote the *cdf* of the reward accumulated in $(0,t)$. The expected value (13) may not give a sufficiently accurate indication about the probability of occurrence of a single event. In [115], several examples are reported revealing a behavior that is not deducible from the mere analysis of the average values. However, computation of $F(t,y)$ is a very complex task [115, 107, 58] and it is not usually available in standard *SPN*-based tools.

6 Dealing with large state spaces

*SPN*s can provide a very compact representation of very large systems. This is reflected in an exponential growth of the reachable markings as a function of the primitive elements in the *SPN* (places and transitions), and as a function of the number of tokens in the initial marking. This exponential growth of the state space has been often recognized [98] as a severe limitation in the use of the *SPN* paradigm to deal with real life applications. Therefore, a large effort has been devoted to overcome, or to alleviate this problem.

Since Markovian-*SPN*s are based on the solution of a *CTMC*, all the techniques that have been explored to handle very large Markov chains can profitably be utilized in connection with *SPN*s. However, original lines of research have been particularly developed in the context of *SPN*s. When dealing with large models, not only that the solution of the system becomes difficult, but the model description and the computer representation also become tedious.

Distributed algorithms - Distributed algorithms have been specifically developed for both the generation of the reachability graph from an *SPN* and for the solution of the underlying *CTMC* [32, 93, 6, 43]. Distributed approaches in the generation phase are typically penalized by the irregularities in the required data structures. Nevertheless, a distributed implementation may achieve a significant speed up in the computational time and a considerable extension of the cardinality of the solvable models [93]. The

distributed steady-state solution of the *CTMC* has proven to outperform standard techniques [6] achieving higher rates of convergence especially when applied to large, stiff problems.

Structured representation - An approach to increase the size of tractable *CTMC*s, is to represent the generator matrix in a compact form as a combination of smaller component matrices, and to exploit this representation in the solution algorithm. A compositional technique based on Kronecker operators proposed in [104], was initially transferred to the *SPN* framework in [54, 55]. Subsequently, efficient techniques have been published for the reachability analysis [79, 80] and for the numerical solution [25, 26, 81] which exploit the specific structure of the generator matrix. To use the structured analysis technique, the *SPN* model has to be described appropriately by means of submodels interacting via synchronizing transitions. Structured schemes for asynchronously interacting submodels have been presented in [24]. Since memory space is often the bottleneck in dealing with large *SPN* models, the structured representation can be very effective.

Hierarchical models including SPNs If an overall system model can be composed from submodels then each submodel is solved separately and results passed to higher level submodel. The hierarchy can be homogeneous where each submodel is of *SPN*-type or heterogeneous. SHARPE sofware package allows models of seven different types to be combined together [110] in such manner. Some of the model types included in SHARPE are: *GSPN*s, product-form queueing networks, Markov chains, fault trees and so on. language of *SPN*s. Other authors have also used such hierarchical models where *GSPN* submodel results are supplied as parameters to a product-form queueing network [12, 13]. A package supporting the view of replacing *GSPN* places by queueing systems has been presented in [14]. Particular classes of *SPN*s, like those originating queueing models with matrix-geometric structure [102], have been considered and a tool has been built for their analysis [70, 71]

Product form SPN - Queueing networks with product-form equilibrium distribution are well established and find application in a variety of fields. With the aim of overcoming the state explosion problem, several proposals have been recently documented to import the product-form concept into the *SPN* arena. In [86], a class of *SPN*s is identified for which a product-form solution can be written from the knowledge of partial balance equations. The generation of the reachability graph is needed to recognize this class of *SPN*s. An extension of this work is presented in [88]. Henderson et al. [74] have developed a product-form criterion based only on the structure of the *SPN*, without the need to generate the reachability graph. A comparative analysis of these two types of appraoches has been reported in [56]. This comparison showed for the first time the possibility of recognizing whether an *SPN* has product-form solution using results from the structural analysis. A complete characterization of this class of models is in [21] and a necessary and sufficient condition for the existence of a positive solution for

the traffic equation is in [22]. Specific algorithms for the computation of product-form solution have been presented in [50, 113]. Mean value analysis for non-product-form *SPN*s has been explored in [112].

PN-driven techniques - These techniques deal with the reduction of both memory requirements and time complexity of the solution algorithms of *SPN*s by using information about the structure of the untimed *PN* models. High Level Stochastic Petri Nets, such as for example Stochastic Well-formed Coloured Nets *(SWN)*, often exhibit behavioral symmetries that can be exploited to reduce the size of the state space, and of the corresponding *CTMC*, by grouping states into *equivalence classes*. The desirable properties of techniques based on this idea are the possibility of automatically discovering the symmetries using only the information contained in the model description at the *PN* level, and the possibility of directly generating the reduced state space (and the lumped Markov chain) without first building the complete reachability graph. A method for the construction of a lumped *CTMC* from and *SWN* model has been presented in [36, 37]. In [67, 68] it has been shown that in some cases it is possible to integrate this method with the decomposition methods based on Kronecker Algebra. In [111], the same idea has been exploited and a method for the construction of the lumped *CTMC* starting from the high-level description is given.

Deterministically Synchronized Sequential Processes (DSSP) are a class of *SPN*s that can be obtained by resorting to simple modular design principles, and for this class a well-established theory exists for the analysis of their qualitative behavior [106, 114]. Net-driven techniques, developed for *DSSP*s, can recognize and extract from the original model a set of simpler auxiliary submodels that are then analyzed through approximate iterative techniques [29] as well as exact solution [30].

CTMC-drive techniques - For a CTMC with a large number of states, only a few states will likely carry most of the probability mass. If we can recognize the states with neglible probability mass in advance of their generation, then such states need not be generated. Such state truncation techniques have been successfully utilized in the context of *SPN* reliability and availability models [78, 99]. Another technique for avoiding large state spaces is to solve a set of CTMC (or SPN) submodels in isolation and pass necessary information from their solution to other submodels. This may need fixed-point iteration among submodels [47, 121]. A time scale decomposition approach (imported from CTMC literature [20]) has been proposed in [77]. This approach requires that the transitions of the *PN* can be classified into two classes: fast and slow transitions.

Performance Bounds - A complementary approach to the development of efficient solution techniques for the computation of performance measures, is the search for bounds. Bounds require less computational effort with respect to the cost of exact solution, since they are estimated based on equations at the *SPN* level, and do not require the generation of the reachability graph. Moreover, the evaluation of the bounds is usually not restricted to Markovian nets. Effort on deriving performance bounds haven been

afoot since the beginning of the research on *SPN* [97, 23]. The evaluation of bounds for the subclass of marked graphs was presented in [27, 29], and in [28] for *SPN*s with a unique consistent firing count vector. A general approach for the computation of bounds has been formulated in [34], based on operational analysis techniques applied at the *SPN* level, with very weak assumptions on their timing semantics. The bounds can be obtained in polynomial time by solving suitable linear programming problems, and depend only on the mean values of the firing times and are insensitive to their distribution. In the case of Markovian *SPN*s, an improved solution technique, based on the randomization algorithm, has been presented in [91].

7 Non-Markovian Stochastic Petri Nets

According to Section 2, in order to define a *SPN* with generally distributed transitions, the following entities must be specified for each transition $t_g \in T$: the *cdf* $(G_g(t))$ of the random firing time γ_g, and the execution policy for determining (a_g, ι_g).

In recent years, several classes of *SPN* models have been developed which incorporate some non-exponential characteristics in their definition, and which adhere to the individual memory semantics discussed in [1]. With the aim of specifying non-Markovian *SPN* models that are analytically tractable, three main lines of research can be envisaged [42, 19]:

- an approach based on *Markov regenerative theory* [49, 83];
- an approach based on the use of *supplementary variables* [52];
- an approach based on *state space expansion* [19].

The first line originated from a particular case of non-Markovian *SPN*, defined in [4], where, in each marking, a single transition is allowed to have associated a deterministic firing time with *prd* policy *(Deterministic and SPN - DSPN)*. Choi et al. [37, 38] have observed that the marking process underlying a *DSPN* is a *Markov Regenerative Process (MRGP)* for which equations for the transition probability matrix in transient and in steady-state can be derived.

A semantic generalization of the previous formulation, has been proposed in [18], by including the possibility of modeling *prs* transitions and in [15] by including *pri* transitions. The most general framework under which the Markov regenerative theory has been applied is the one in which any regeneration time period is dominated by a single transition (non-overlapping dominant transitions).

The second line resorts to the use of supplementary variables [52]. The method has been, up to now, applied to *prd* execution policies only and with mutually exclusive general transitions. The steady-state solution has been proposed by German and Lindemann in [62, 89, 90], while the possibility of applying the methodology to the transient analysis has been explored in [61, 73].

A comparison of numerical methods for the transient analysis of *MRGP*s applying the Markov regenerative theory and the method of the supplementary variables has been presented in [63].

The third line of research, aimed at affording the solution of non-Markovian *SPN*, is based on the expansion of the reachability graph of the basic *PN*. In this approach, the original non-Markovian marking process is approximated by means of a *CTMC* defined over an augmented state space. According to the definitions given in Section 2, the expansion technique can be realized by assigning to each transition a continuous Phase-type *(PH)* distributed random variable [102, 16].

The merit of this approach is the flexibility in modeling any combination of *prd* and *prs* memory policies and any number of concurrent or conflicting transitions with generally distributed firing times. Moreover, the expansion technique can be easily implemented by a computer program, starting from the basic specification at the *PN* level, so that all the solution steps can be hidden from the modeler [53]. The drawback of this approach is, of course, the explosion of the state space that can be alleviated by resorting to the use Kronecker operators for matrices [69].

A very recent and interesting modification of the expansion technique, resorts to the use of discrete *PH*-type random variables [41, 48], so that the continuous-time marking process is approximated by an expanded discrete-time Markov chain (*DTMC*). However, discrete random variables are not covered by the assumptions stated in Section 2, and their consideration is outside the scope of the present review.

The subclasses of *SPN*s in which none of the transitions can be preempted before firing allow effectiv analysis and simulation methods even with non-exponentially distributed firing times [10, 9].

7.1 Markov Regenerative Stochastic Petri Nets

The formalization of a class of Markov Regenerative Stochastic Petri Nets *(MRSPN)* has been presented in [37]:

Definition 1 *A SPN is called a Markov Regenerative Stochastic Petri Net (MRSPN) if its marking process is a Markov Regenerative Process (MRGP)*[1].

*MRGP*s [83] (or Semi Regenerative Processes [49]) are discrete-state continuous-time stochastic processes with an embedded sequence of *Regenerative Time Points (RTP)* [120], at which the process enjoys the Markov property. The relevance of Definition 1 comes from the fact that *MRSPN*s can be studied by resorting to the techniques available for *MRGP*s [49, 83]. Based on the concept of memory in a *SPN*, RTPs can be defined as follows:

Definition 2 *A regenerative time point (RTP) in the marking process* $\{\mathcal{M}(t)\}$ *underlying an SPN is an instant of time where all the transitions do not have memory; i.e. all the*

[1] *MRSPN*s are referred to as Semi Regenerative *SPN*s in [47].

memory variables a_k and the resampling indicator variables ι_k ($k = 1, 2, \ldots, n_t$) are equal to zero.

EXP transitions with either *prd* or *prs* policy do not have memory, and they do not affect the search for RTPs. Only generally distributed transitions, or EXP transitions with *pri* policy have to be checked to fulfill the requirements of Definition 2. The framework in which an *SPN*, with mixed preemption policies [119], generates a *MRGP* marking process is based on the notion of *non-overlapping dominant transition* [18].

Definition 3 *A transition, for which the beginning and the end of its memory cycle correspond to the initial and final RTPs of a regeneration interval, is said to be dominant over the considered regeneration interval. An SPN with non-overlapping dominant transitions is a MRSPN.*

The evolution of the marking process $\{\mathcal{M}(t)\}$ during a regeneration period between two consecutive RTP's is called the process *subordinated* to the dominant transition. The subordinated process can contain any number of EXT transition firings, but Definition 3, includes the possibility that the memory cycle of one transition is completely contained within the memory cycle of the dominant one, hence allowing simultaneous enabling of different general transitions inside the same subordinated process. However, an analytic derivation is possible if the subordinated processes are restricted to be a *CTMC* or a semi-Markov process *(SMP)*.

If the general transitions are *prd*, a more complex situation has been examined in [105], in which more than one general transition can be enabled at the same time. An example with mixed preemption policies can be found in [119], while an example with simultaneously enabled *prd* transitions is completely developed in Section 8.

7.1.1 Analysis by Markov Regenerative Theory

By the memoryless property of the *MRGP* at the RTPs, the analysis of an *MRSPN* can be split into independent subproblems given by the subordinated processes between any two consecutive RTPs. The probability functions that must be evaluated for the transient analysis of a *MRSPN* are commonly referred to as global and local kernels [49, 83]. The global kernel $\mathbf{K}(t) = [K_{ij}(t)]$ describes the occurrence of the next RTP:

$$K_{ij}(t) = Prob\{M_{(1)} = j\ ,\ \tau_1^* \leq t | \mathcal{M}(0) = i\}$$

where $\mathcal{M}(0) = i$ indicates the initial condition for the marking process, τ_1^* is the next RTP and $M_{(1)}$ is the right continuous state hit by the marking process at the next RTP. The local kernel $\mathbf{E}(t) = [E_{ij}(t)]$ describes the state transition probabilities inside a regeneration period, before the next RTP occurs:

$$E_{ij}(t) = Prob\{\mathcal{M}(t) = j\ ,\ \tau_1^* > t | \mathcal{M}(0) = i\}$$

In the special case where the marking process is a semi-Markov process, all the reachable states must be *RTP*s and the local kernel $\mathbf{E}(t)$ is a diagonal matrix. The conditions under which an *SPN* generates a marking process that is an *SMP* have been studied in [57].

The kernel entries are a function of the execution policy of the single transition dominating the considered regenerative period. For a *prd* dominant transition the analysis is given in [38], for a *prs* dominant transition in [18, 117] and for a *pri* dominant transition in [15].

Let $\mathbf{V}(t) = [V_{ij}(t)]$ denote the transition probability over $(0, t)$, i.e.:

$$V_{ij}(t) = Prob\{\mathcal{M}(t) = j \mid \mathcal{M}(0) = i\}$$

Based on the global and the local kernels the transient analysis can be carried out in the time domain by solving the following generalized Markov renewal equation:

$$V_{ij}(t) = E_{ij}(t) + \sum_k \int_0^t d\,K_{ik}(y)\, V_{kj}(t-y) \tag{14}$$

or in the transform domain:

$$\mathbf{V}^{\sim}(s) = [\mathbf{I} - \mathbf{K}^{\sim}(s)]^{-1}\, \mathbf{E}^{\sim}(s) \tag{15}$$

where the superscript $^{\sim}$ indicates the Laplace-Stieltjes transform and s is the variable corresponding to the time variable t.

A time domain solution for the transition probability matrix $\mathbf{V}(t)$ can be obtained by numerically integrating Equation (14). Alternatively, starting from the Laplace transform Equation (15), a combination of symbolic and numeric computation is needed to obtain measures in the time domain [19]. In both cases, the complexity of the solution limits the applicability of the procedure to an *MRGP* with a small number of states.

For the purpose of the steady-state analysis of an *MRSPN*, the following measures of the subordinated processes are needed:

$$\begin{aligned} \alpha_{ij} &= \int_0^\infty E_{ij}(t)\, dt \\ \phi_{ij} &= Prob\{M_{(1)} = j \mid \mathcal{M}(0) = i\} \end{aligned} \tag{16}$$

α_{ij} is the expected time the subordinated process starting from state i spends in state j, and ϕ_{ij} is the probability that the subordinated process starting from state i is followed by a subordinated process starting from state j. Indeed the matrix $\mathbf{\Phi} = [\phi_{ij}]$ is the transition probability matrix of the *DTMC* embedded at the RTPs. The measures in Equation (16) can be obtained from the global and local kernels either in the time or in the transform domain:

$$\alpha_{ij} = \int_0^\infty E_{ij}(t)\, dt = \lim_{s\to 0} E_{ij}^{\sim}(s)/s \tag{17}$$

$$\phi_{ij} = \lim_{t\to\infty} K_{ij}(t) = \lim_{s\to 0} K_{ij}^{\sim}(s) \tag{18}$$

The evaluation of the measures in (16) is also dependent on the nature of the execution policy associated with the transition dominating the subordinated processes (as it is indicated by Equations (17) and (18)). For a *prd* dominant transition the analysis is given in [4], for a *prs* dominant transition in [118] and for a *pri* dominant transition in [17].
The steady-state analysis of a *MRSPN* requires three steps:

Step 1: Evaluate the $\alpha = [\alpha_{ij}]$ and $\mathbf{\Phi} = [\phi_{ij}]$ matrices based on [4, 118, 17].

Step 2: Evaluate the vector $D = [D_i]$, whose elements are the steady state probabilities of the *DTMC* embedded at the RTPs. D is the unique solution of:

$$D = D\mathbf{\Phi} \; ; \qquad \sum_i D_i \; = \; 1$$

Step 3: The steady-state probabilities of the *MRGP* are given by:

$$v_j \; = \; \lim_{t\to\infty} \; Prob\,\{\mathcal{M}(t) = j\} \; = \; \frac{\sum\limits_k D_k\,\alpha_{kj}}{\sum\limits_k D_k\,\alpha_k} \tag{19}$$

An example of solution of a *MRSPN* with *prs* dominant transitions using Equation (19) has been reported in [118], while in [119] an example with mixed preemption policies has been considered.

7.1.2 Method of Supplementary Variables

The method of supplementary variable has been applied to *MRSPN*s in which, in each (tangible) marking, at most a single enabled transition can have a non-exponential distribution with *prd* policy, with all the other enabled transitions EXP. Let $a(t)$ be the age at time t of the only enabled non-exponential transition, if any. Since only one transition can be enabled at any time, $a(t)$ is the age of the whole model at time t. Under these restrictions, the marking process $\mathcal{M}(t)$ together with the supplementary variable $a(t)$ (i.e., $(\mathcal{M}(t), a(t))$) is a *Markov process* over the state space $\mathcal{R}_0 \times \mathbb{R}$ [52], where $\mathcal{R}_0$ is the set of reachable tangible markings and $\mathbb{R}$ is the set of non-negative real numbers. The joint process can be analyzed by the method of supplementary variables [52] as shown in [62, 61, 73]. Following the concept and the notations of [61] the solution approach is briefly summarized.

Let T^G be the set of non-exponential timed transitions. The tangible state space $\mathcal{R}_0$ is partitioned into $\#T^G + 1$ disjoint subsets. $\mathcal{R}_0^E$ is the set of states in which no general transition is enabled ($a(t) = 0$ when $\mathcal{M}(t) \in \mathcal{R}_0^E$), and $\mathcal{R}_0^g, g \in T^G$ are the sets of states in which the general transition t_g (the dominant one) is enabled. The superscript ^{E} refers to the states in $\mathcal{R}_0^E$ and the superscript ^{g} (or $^\ell$) refers to the states in $\mathcal{R}_0^g$. The probability of being in state i at time t is $\pi_i(t) = Prob\,\{\mathcal{M}(t) = i\}$. Given that in state i at time t the single dominant transition $t_g \in T^G$ is enabled, with age $a_g(t)$ and *cdf* $G_g(x)$, the so called, *age rate* $h_i(t,x)$ describes the conditional firing rate of t_g in i:

$$h_i(t,x) \; = \; \frac{Prob\,\{\mathcal{M}(t) = i,\, x < a_g(t) \le x + dx\}}{dx} \cdot \frac{1}{1 - G_g(x)}$$

The effect of the firing of a general transition t_g is stored in a branching probability matrix $\mathbf{\Delta} = \Delta_{ij}$ whose generic entry has the following meaning [2, 38]:

$$\Delta_{ij} \; = \; Prob\{\text{next marking is } j \mid \text{current marking is } i \text{ and transition } t_g \text{ fires}\}$$

With the above assumptions and definitions, the *age rate* vector $\mathbf{h}(t,x) = [h_i(t,x)]$ with $i \in \mathcal{R}_0^g$ is described by the following partial differential equation:

$$\frac{\partial}{\partial t}\mathbf{h}^g(t,x) \; + \; \frac{\partial}{\partial x}\mathbf{h}^g(t,x) \; = \; \mathbf{h}^g(t,x)\,\mathbf{Q}^g \tag{20}$$

The transient state probability vector $\Pi(t)$, can be calculated in partitioned form: $\Pi(t) = [\Pi^E(t), \Pi^g(t), \Pi^\ell(t), \ldots]$. The process evolution in $\mathcal{R}_0^E$ is described by the following ordinary differential equation:

$$\begin{aligned} \frac{d}{dt}\Pi^E(t) \;\; = \;\; & \Pi^E(t)\,\mathbf{Q}^E + \\ & \sum_{g \in T^G} \int_0^\infty \mathbf{h}^g(t,x)\,dG_g(x)\,\mathbf{\Delta}^{g,E} + \\ & \sum_{g \in T^G} \Pi^g(t)\,\mathbf{Q}^{g,E} \end{aligned} \tag{21}$$

In (21), the state probabilities inside $\mathcal{R}_0^E$ can change: *i)* - by the firing of an EXP transition which results in a new marking in $\mathcal{R}_0^E$ (1st term); *ii)* - by the firing of a general transition when the reached state is in $\mathcal{R}_0^E$ (2nd term); *iii)* - by the disabling of a general transition when the reached state is in $\mathcal{R}_0^E$ (3rd term).

The boundary condition for Equation (20) is given by:

$$\begin{aligned} \mathbf{h}^g(t,0) \;\; = \;\; & \Pi^E(t)\mathbf{Q}^{E,g} + \\ & \sum_{\ell \in T^G} \int_0^\infty \mathbf{h}^\ell(t,x)\,dG_\ell(x)\,\mathbf{\Delta}^{\ell,g} + \\ & \sum_{\ell \in T^G, \ell \neq g} \Pi^\ell(t)\,\mathbf{Q}^{\ell,g} \end{aligned} \tag{22}$$

In (22), a general transition t_g can be activated: *i)* - by the firing of an EXP transition in $\mathcal{R}_0^E$ leading to a state in which t_g is enabled (1st term); *ii)* - by the firing of a general transition t_ℓ when in the reached state t_g is enabled (or reenabled if $t_g = t_\ell$) (2nd term); *iii)* - by the firing of an EXP transition which disables the active general transition t_ℓ and in the reached marking the general transition t_g is enabled (3rd term).

Once $\mathbf{h}^g(t,x)$ is computed from (20), the transient state probability vector in $\mathcal{R}_0^g$ can be calculated from:

$$\Pi^g(t) = \int_0^\infty \mathbf{h}^g(t,x)\,(1 - G_g(x))\,dx \tag{23}$$

The initial conditions are $\Pi^E(0)$ and $\mathbf{h}^g(0,x) = \Pi^g(0)\,\delta(x)$, where $\delta(x)$ is the Dirac delta function.

An iterative algorithm for solving the above equations, based on a fixed size discretization interval (d) for the continuous variables has been proposed in [63]. The steps of the algorithm are the following:

1. Compute the age rates in the next time instant

$$\mathbf{h}^g(id, jd) = \mathbf{h}^g((i-1)d, (j-1)d)e^{\mathbf{Q}^g\,d}$$

and set $\mathbf{h}^g(id, 0) = 0$

2. Given the age rates $\mathbf{h}^g(id, jd)$, $j = 0, 1, \ldots$, compute the state probabilities $\Pi^g(id)$ from (23)

3. Compute the state probabilities $\Pi^E(id)$ from the ordinary differential Equation (21)

4. Compute the activation rate of the general transitions $\mathbf{h}^g(id, 0)$ from the boundary conditions (22)

5. Check the convergence and go back to step 2 or start with the next time instant $(i+1)d$

An improved numerical procedure, based on the same equations, but with an adaptively varying discretization interval in the integration procedure has been recently described in [73]. The steady-state behavior of the considered class of *MRSPN* can be easily obtained, in the supplementary variable setting, by making the time derivatives equal to 0 in the above set of equations. Lindemann proposed an effective numerical method to evaluate the steady-state probabilities based on this approach [89, 90].

7.2 State space expansion

The technique based on the state space expansion is not restricted to *MRSPN* schemes, but any combination of *prd* and *prs* transition is, in principle, acceptable. The technique consists in approximating the firing times with *PH*-distributed [102] random variables and generating the expanded *CTMC* obtained by combining the reachable states with all the phases of the *PH*-distributions associated with the enabled transitions. An overview of the methods and tools available to estimate the parameters of a *PH*-distribution from a given *cdf* can be found in [16].

The expansion algorithm can be performed automatically by a computer program, and is driven by the execution policy associated with different transitions [53]. The result of the expansion algorithm is that each marking of the original *PN* is blown into a macrostate in the new state space. When the firing times of the original *PN* are already *PH*-distributed, this approach provides exact results. Otherwise, a preliminary step is needed in order to approximate the given distributions by means of a suitable *PH* [16].

The analysis method consists of the following steps:

Step 1: Approximate the firing time distributions by means of *PH cdf*s.

Step 2: Based on the *PN* description, the *PH* distributions assigned to each transition, and their execution policy (among *prd* and *prs*, only) generate the expanded state space and the infinitesimal generator of the expanded *CTMC*.

Step 3: Analyze the expanded *CTMC* by standard techniques (Section 4) and evaluate the final results at the *PN* level by keeping the correspondence between original markings and the macrostates in the expanded state space.

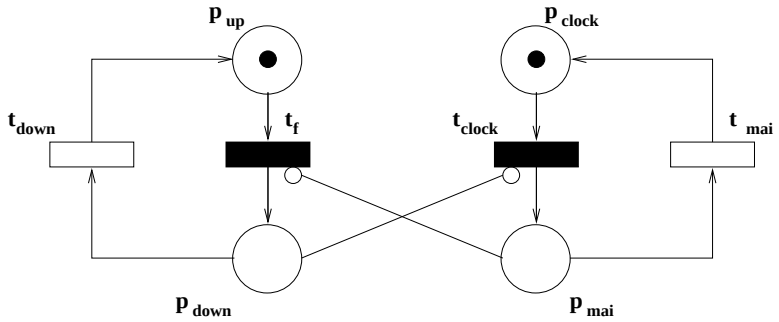

Figure 7: MRSPN model for a preventive maintenance system

Cumani [53] has developed a package, called ESP, which automatically performs Step 2 - 3. A recent attempt to save storage space by algorithmically describing the infinitesimal generator of the expanded *CTMC* by means of Kronecker operators has been documented in [69].

8 Example: a preventive maintenance system

A quantitative example of a preventive maintenance system is developed. Preventive maintenance is considered as one of the key factors to increase system productivity and to reduce production costs. The growing importance of maintenance in industrial applications has led an increased sophistication in the mathematical models required to analyze its impact on the system behavior [122, 85].

The system starts in a working state, but it ages with time and it eventually fails if no preventive maintenance action is done. Once it crashes, a random amount of time is required to bring the system back up and to restart it. Preventive maintenance is performed at fixed intervals from the start (or the last restart) of the system in the working state. The preventive maintenance activity takes an exponentially distributed amount of time and completely regenerates (renews) the system.

Let d be the constant inspection interval. d is a critical design parameter: if d approaches zero, the system is always under maintenance and its availability drops to zero. On the other hand if d becomes too large the beneficial effect of the preventive maintenance action becomes negligible.

The aim of this example is to elaborate a closed-form analytical expression for the steady-state behavior of the system, and to evaluate the optimal value of the maintenance interval that maximizes system availability.

8.1 Petri Net Model

Figure 7 shows the *MRSPN* representation of the system described in the previous paragraph. The working state is modeled by place p_{up}. The generally distributed transition t_f models

the failure distribution whose firing leads the system to place p_{down}. Upon system failure, the preventive maintenance activity is suspended: the inhibitor arc from place p_{down} to transition t_{clock} is used to model this fact.

The deterministic transition t_{clock} models the constant inspection interval. It is competitively enabled with t_f so that the one that fires first disables the other one. Once t_{clock} fires, a token moves in place p_{mai} and the activity related with the preventive maintenance (transition t_{mai}) starts.

During the preventive maintenance phase, the system is switched off and cannot fail (inhibitor arc from place p_{mai} to transition t_f). The completion of the maintenance (firing of t_{mai}) re-initializes the system in an *as good as new* condition; hence t_f is assigned a *prd* policy. Since upon failure and repair a complete d interval must elapse before the successive preventive maintenance takes place, t_{clock} also must be assigned a *prd* policy.

As can be observed from Figure 7, t_f and t_{clock} are conflicting *prd* general transitions that fit into the framework elaborated in [105].

8.2 Model solution

Since there are no immediate transitions in the *PN*, all the markings are tangible. Starting from the initial marking m_1 represented in Figure 7 , the token distribution in the reachable markings (assuming the following order for the places: p_{up}, p_{clock}, p_{down}, p_{mai}) is given by :

$$m_1 = (1,1,0,0)\,, \qquad m_2 = (0,1,1,0)\,, \qquad m_3 = (1,0,0,1)$$

From marking m_1 both t_f and t_{clock} may fire leading to m_2 and m_3, respectively. From m_2 only t_{down} can fire leading to m_1 and, finally, from m_3 only t_{mai} can fire leading to m_1. As a consequence, the kernel matrices $\mathbf{K}(\mathrm{t})$ and $\mathbf{E}(\mathrm{t})$ have the following structure:

$$\mathbf{E}(t) = \begin{pmatrix} E_{11}(t) & 0 & 0 \\ 0 & E_{22}(t) & 0 \\ 0 & 0 & E_{33}(t) \end{pmatrix}$$

$$\mathbf{K}(t) = \begin{pmatrix} 0 & K_{12}(t) & K_{13}(t) \\ K_{21}(t) & 0 & 0 \\ K_{31}(t) & 0 & 0 \end{pmatrix}$$

Since $\mathbf{E}(t)$ is a diagonal matrix, the marking process is an*SMP*. Let $G_f(t)$ be the *cdf* of the firing time associated with transition t_f, and d be the deterministic maintenance interval associated with t_{clock}. Furthermore, let λ_1 and λ_2 be the firing rates associated with transitions t_{down} and t_{mai}, respectively. The non-zero kernel entries are:

$$K_{12}(t) \;=\; \begin{cases} G_f(t) & 0 \le t < d \\ \\ G_f(d) & t \ge d \end{cases} \tag{24}$$

$$K_{13}(t) = \begin{cases} 0 & 0 \le t < d \\ 1 - G_f(d) & t \ge d \end{cases} \tag{25}$$

$$K_{21}(t) = 1 - e^{-\lambda_1 t} \quad ; \quad K_{31}(t) = 1 - e^{-\lambda_2 t} \tag{26}$$

$$E_{11}(t) = \begin{cases} 1 - G_f(t) & 0 \le t < d \\ 0 & t \ge d \end{cases} \tag{27}$$

$$E_{22}(t) = e^{-\lambda_1 t} \quad ; \quad E_{33}(t) = e^{-\lambda_2 t} \tag{28}$$

To obtain the steady-state solution, we follow the procedure described in Section 7.1.1.

Step 1:

$$\alpha = \begin{pmatrix} \alpha_{11} = \int_0^d [1 - G_f(t)]\, dt & 0 & 0 \\ 0 & \alpha_{22} = \frac{1}{\lambda_1} & 0 \\ 0 & 0 & \alpha_{33} = \frac{1}{\lambda_2} \end{pmatrix}$$

$$\mathbf{\Phi} = \begin{pmatrix} 0 & G_f(d) & 1 - G_f(d) \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Step 2:

$$D = \left[\frac{1}{2}, \ \frac{1}{2\,G_f(d)}, \ \frac{1}{2\,(1 - G_f(d))} \right]$$

Step 3:

$$v = \left[\frac{1}{A 2 \alpha_{11}}, \ \frac{1}{A 2 \alpha_{22} G_f(d)}, \ \frac{1}{A 2 \alpha_{33} (1 - G_f(d))} \right] \tag{29}$$

where $A = \frac{1}{2\alpha_{11}} + \frac{1}{2\alpha_{22} G_f(d)} + \frac{1}{2\alpha_{33}(1 - G_f(d))}$.

8.3 Results

The steady-state availability is given by the probability of being in state m_1 (entry v_1 in 29). The effect of the length of the preventive maintenance interval d on the system availability can now be examined.

The numerical computations are performed assuming the following values:

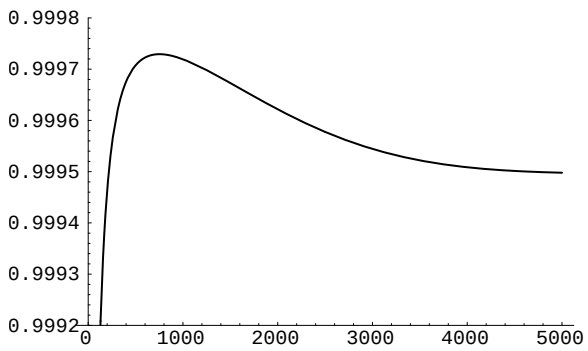

Figure 8: steady-state availability versus maintenance interval

i) - Transition t_f is distributed according to a Weibull *cdf* $G_f(t) = 1 - e^{-ct^\beta}$, where β is the shape parameter and c is the scale parameter, respectively. We assume $\beta = 2.0$ (increasing failure rate) and $c = 2 \cdot 10^{-7}$. With the above value for c the expected value of the Weibull *cdf* is $\mathbb{E}(\gamma_f) = 1981.66$ h.

ii) - $\lambda_1 = 0.1\ h^{-1}$ and $\lambda_2 = 1\ h^{-1}$ for the firing rates of transitions t_{down} and t_{mai}, respectively.

iii) - The preventive maintenance interval d varying from 0 to 5000 h.

Figure 8 plots the system availability v_1 versus the maintenance interval d. If $d = 0$, the system is always under maintenance, and is completely unavailable. As d increases, the steady-state availability increases as well. However, for large d the effect of the preventive maintenance is overshadowed by the downtime due to failure, and in the limit $d \to \infty$, the availability approaches the value when there is no preventive maintenance. The optimal maintenance interval is $d = 752\ h$, at which the availability achieves its maximum value $v_1 = 0.999727$.

9 Fluid Stochastic Petri Nets

Recognizing the increasing use of stochastic fluid flow models in performance analysis, Trivedi and Kulkarni introduced the class of Fluid Stochastic Petri Nets (*FSPN*) [82]. This class extends the traditional integer token concept by introducing the possibility for the tokens to be real (positive) entities assigned to special continuous places. For a discussion about continuous and hybrid *PN* models see also [5].

The places are partitioned into a set of discrete places P_d containing an integer number of tokens and a set of fluid (or continuous) places P_c containing a real fluid level. The state

space of an *FSPN* is partially discrete and partially continuous. The discrete part is an integer vector accounting for the number of tokens in the discrete places. The continuous part is a vector of real numbers accounting for the fluid levels in the continuous places.

In [82], the continuous part of a marking does not affect the discrete-state stochastic process defined over the discrete places (which is a homogeneous *CTMC*). Let S be the set of reachable discrete markings and $\mathbf{Q}$ be the infinitesimal generator of the underlying *CTMC*. The evolution of the continuous part of the marking is governed by flow rate functions which depend only on the discrete part. Let $r_i(n)$ be the flow rate out of a fluid place $i \in P_c$ given that $n \in S$ is the current discrete state. The fluid level $X_i(t)$ of the continuous place $i \in P_c$ given $n \in S$ and t is characterized by the following equation:

$$\frac{dX_i(t)}{dt} = \begin{cases} r_i(n) & \text{if} \quad X_i(t) > 0 \\ max\{r_i(n), 0\} & \text{if} \quad X_i(t) = 0 \end{cases} \tag{30}$$

Define the row vector $\vec{H}(t, \vec{x}) = [H_n(t, \vec{x})]$ whose entries $H_n(t, \vec{x})$ (with $n \in S$) are the transient distribution functions:

$$H_n(t, \vec{x}) = Prob\{X_i(t) \leq x_i, i \in P_c, n \in S\}$$

In accordance with the result of stochastic fluid models [7] $\vec{H}(t, \vec{x})$ satisfies [82]:

$$\frac{\partial \vec{H}(t, \vec{x})}{\partial t} + \sum_{i \in p_c} \frac{\partial \vec{H}(t, \vec{x})}{\partial x_i} \mathbf{R}_i = \vec{H}(t, \vec{x})\mathbf{Q}, \quad \vec{x} > 0 \tag{31}$$

with the boundary condition $H_n(t, \vec{x}) = 0$ if $x_i = 0$ and $r_i(n) > 0$. In the above expression $\mathbf{R}_i = diag(r_i(n))$.

The steady-state behavior of an *FSPN* is obtained by eliminating the time dependent derivative from (31). The analytical evaluation of the transient as well as the steady-state behavior of an *FSPN* with multiple fluid places is very hard. Numerical techniques are being explored. An *FSPN* with a single fluid place results in a traditional stochastic fluid flow model for which the steady-state analysis has been investigated in [7, 84], based on the spectral decomposition of $\mathbf{Q}$.

An extension of the original *FSPN* model was investigated by Horton at al. [76]. In the extended *FSPN* class, mutual interactions of the continuous over the discrete part and viceversa are allowed. Considerations on the numerical analysis of this class of *FSPN*s with a single fluid place can be found in [76]. A discrete event simulation method has been investigated in [46].

10 Conclusions

Markovian-*SPN*s and their most popular variant, the *GSPN*s, have become a well known modeling technique in industrial and academic environments. The availability of well established and user-friendly tools based on this paradigm has largely contributed to the success of *PN*s as a general purpose, flexible and effective modeling and analysis language.

The research lines devoted to exploit specific properties and structures either at the *PN* or at the reachability graph level, are rapidly increasing the size of problems that can be effectively handled. Moving the frontiers of *PN*-based models to deal with very large state spaces has attracted a relevant efforts in the *PN* community as evidenced by the number of papers dealing with these topics, and is still an open research area.

New challenging and promising results have been recently obtained in the attempt to overcome the exponential assumption. Combination of deterministic and stochastic timings in the same model represents a realistic goal at the present state of the art, particularly for what concerns the steady-state analysis. Numerical techniques in the transient domain are still in the infant stage but a variety of methods and algorithms have been developed.

Fluid models are of extreme interest per se, and as a continuous approximation to situations where an enormous number of discrete objects has to be considered. The preliminary results are encouraging but it is hard to forecast the success that these models will encounter in applications.

Acknowledgements

This work has been partially supported by the Italian CNR under Grant No. 96.01939.CT12 and Hungarian OTKA under Grant No. T-16637. We like thank Ricardo Fricks for his help with Figure 3.

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, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, New York, 1995.

[4] 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.

[5] H. Alla and R. David. Continuous and hybrid Petri nets. *Journal of Circuits, Systems and Computers*, This same issue, 1997.

[6] S.C. Allmaier, M. Kowarschik, and G. Horton. State space construction and stedy-state solution of GSPNs on a shared-memory multiprocessor. In *7-th International*

Conference on Petri Nets and Performance Models - PNPM97, pages 112–121. IEEE Computer Society, 1997.

[7] D. Anick, D. Mitra, and M.M. Sondhi. Stochastic theory of data handling systems with multiple sources. *Bell System Technical Journal*, 61:1871–1884, 1982.

[8] F. Bacelli. Ergodic theory of stochastic Petri networks. *Annal. Prob.*, vol. 20:375–396, 1992.

[9] F. Bacelli and M. Canales. Paralell simulation of stochastic Petri nets using recurrence equations. *ACM Transactions on Modelling and Computer Simulation*, 3:20–41, 1993.

[10] F. Bacelli, B. Gaujal and S. Foss. Structural, themporal and stochastic properties of unbounded free-choice Petri nets. *INRIA technical report*, No. 2411, 1994.

[11] F. Bacelli and Z. Liu. Comparison properties of stochastic decision free Petri nets. *IEEE Transactions on Automatic Control*, 37:1905–1930, 1992.

[12] G. Balbo, S.C. Bruell, and S. Ghanta. Combining queueing network and generalized stochastic Petri net models for the analysis of some software blocking phenomena. *IEEE Transactions on Software Engineering*, 12:561–576, 1986.

[13] G. Balbo, S.C. Bruell, and S. Ghanta. Combining queueing network and generalized stochastic Petri nets for the solution of complex models of system behavior. *IEEE Transactions on Computers*, 37:1251–1268, 1988.

[14] F. Bause. Queueing Petri nets: a formalism for the combined qualitative and quantitative analysis of systems. In *Proceedings 5th International Workshop on Petri Nets and Performance Models - PNPM93*, pages 14–23. IEEE Computer Society, 1993.

[15] A. Bobbio, V.G. Kulkarni, A. Puliafito, M. Telek, and K. Trivedi. Preemptive repeat identical transitions in Markov Regenerative Stochastic Petri Nets. In *6-th International Conference on Petri Nets and Performance Models - PNPM95*, pages 113–122. IEEE Computer Society, 1995.

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

[17] A. Bobbio and M. Telek. Combined preemption policies in MRSPN. In Ravi Mittal et al., editor, *Fault Tolerant Systems and Software*, pages 92–98. Narosa Pub. House, New Dehli - India, 1995.

[18] 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 Computer Society Press, 1995.

[19] A. Bobbio and M. Telek. Non-exponential stochastic Petri nets: an overview of methods and techniques. In *To be published in: Computer Systems Science & Engineering*, 1997.

[20] A. Bobbio and K.S. Trivedi. An aggregation technique for the transient analysis of stiff Markov chains. *IEEE Transactions on Computers*, C-35:803–814, 1986.

[21] R.J. Boucherie. A characterization of independence for competing Markov chain with applications to stochastic Petri nets. In *Proceedings 5th International Workshop on Petri Nets and Performance Models - PNPM93*, pages 117–126. IEEE Computer Society, 1993.

[22] R.J. Boucherie and M. Sereno. On the traffic equations for batch routing queueing networks and stochastic Petri nets. *To appear on: Advances in Applied Probability3*, 1997.

[23] S.C. Bruell and S. Ghanta. Throughput bounds for generalized stochastic Petri ne modelst. In *International Workshop Timed Petri Nets*, pages 250–261, Torino (Italy), 1985. IEEE Computer Society Press No. 674.

[24] P. Buchholz. A hierarchical view of GCSPNs and its impact on qualitative and quantitative analysis. *Journal of Parallel and Distributed Computation*, 15:207–224, 1992.

[25] P. Buchholz. Aggregation and reduction techniques for hierarchical GCSPNs. In *Proceedings International Workshop on Petri Nets and Performance Models - PNPM93*, pages 216–225. IEEE Computer Society, 1993.

[26] P. Buchholz. Hierarchical structuring of superposed GSPNs. In *7-th International Conference on Petri Nets and Performance Models - PNPM97*, pages 81–90. IEEE Computer Society, 1997.

[27] J. Campos, G. Chiola, J.M. Colom, and M. Silva. Properties and performance bounds for timed marked graphs. *IEEE Transactions on Circuits and Systems - I: Fundamental Theory and Applications*, 39:386–401, 1992.

[28] J. Campos, G. Chiola, and M. Silva. Ergodicity and throughput bounds of Petri nets with unique consistent firing count vector. *IEEE Transactions on Software Engineering*, 17:117–125, 1991.

[29] J. Campos, J.M. Colom, H. Jungnitz, and M. Silva. Approximate throughput computation of stochastic marked graphs. *IEEE Transactions on Software Engineering*, 20:526–535, 1994.

[30] J. Campos, M. Silva, and S. Donatelli. Structures solution of stochastic DSSP systems. In *7-th International Conference on Petri Nets and Performance Models - PNPM97*, pages 91–100. IEEE Computer Society, 1997.

[31] F. Comonner, A. W. Holt, S. Even and A. Pnuelli. Marked directed graphs. *J. Comput. Syst. Sci.*, 5:511–523, 1971.

[32] S. Caselli, G. Conte, and P. Marenzoni. Massively parallel analysis GSPN models. In *Proceedings International Conference on Application and Theory of Petri Nets*, pages 181–200. Springer Verlag - LNCS, Vol 935, 1995.

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

[34] G. Chiola, C. Anglano, J. Campos, J.M. Colom, and M. Silva. Operational analysis of timed Petri nets and application to the computation of performance bounds. In *Proceedings 5th International Workshop on Petri Nets and Performance Models - PNPM93*, pages 128–137. IEEE Computer Society, 1993.

[35] G. Chiola, S. Donatelli, and G. Franceschinis. GSPNs versus SPNs: what is the actual role of immediate transitions? In *Proceedings 4th International Workshop on Petri Nets and Performance Models - PNPM91*. IEEE Computer Society, 1991.

[36] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic Well-Formed coloured nets and multiprocessor modelling applications. In K. Jensen and G. Rozenberg, editors, *High-Level Petri Nets. Theory and Application*. Springer Verlag, 1991.

[37] 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.

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

[39] G. Ciardo. Toward a definition of modeling power for stochastic Petri net models. In *Proceedings International Workshop on Petri Nets and Performance Models*, pages 54–62, Madison, 1987. IEEE Computer Society Press no. 796.

[40] G. Ciardo. Petri nets with marking-dependent arc cardinality: properties and analysis. In *Proceedings of the 15-th International Conference on Application and Theory of Petri Nets*, pages 179–198. Lectures Notes in Computer Science #815 - Springer Verlag, 1994.

[41] G. Ciardo. Discrete-time markovian stochastic Petri nets. In *Proceedings of the 2-nd International Workshop on Numerical Solution of Markov Chains*, pages 339–358, 1995.

[42] 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.

[43] G. Ciardo, J. Gluckman, and D. Nicol. Distributed state-space generation of discrete state stochastic models. *ORSA J. Comp. - To appear*, 1997.

[44] G. Ciardo, J. Muppala, and K.S. Trivedi. SPNP: stochastic Petri net package. In *Proceedings 3-rd International Workshop on Petri Nets and Performance Models - PNPM89*, pages 142–151. IEEE Computer Society, 1989.

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

[46] G. Ciardo, D. Nicol, and K. Trivedi. Discrete-event simulation of Fluid Stochastic Petri Nets. In *7-th International Conference on Petri Nets and Performance Models - PNPM97*, pages 217–226. IEEE Computer Society, 1997.

[47] G. Ciardo and K.S. Trivedi. A decomposition approach for stochastic reward net models. *Performance Evaluation*, 18:37–59, 1993.

[48] G. Ciardo and R. Zijal. Well-defined stochastic Petri nets. In *Proceedings of the 4-th International Workshop on Modeling Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'96)*, pages 278–284. IEEE Computer Society Press, 1996.

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

[50] J.L. Coleman, W. Henderson, and P.G. Taylor. Product form equilibrium distributions and an algorithm for classes of batch movement queueing networks and stochastic Petri nets. *Performance Evaluation*, 26:159–180, 1996.

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

[52] 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.

[53] 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.

[54] S. Donatelli. Superposed stochastic automata: a class of stochastic Petri nets amenable to parallel solution. *Performance Evaluation*, 18:21–36, 1993.

[55] S. Donatelli. Superposed generalized stochastic Petri nets: definition and efficient solution. In *Proceedings International Conference on Application and Theory of Petri Nets*, pages 258–277. Springer Verlag - LNCS, Vol 815, 1994.

[56] S. Donatelli and M. Sereno. On the product form solution for stochastic Petri nets. In *Proceedings International Conference on Application and Theory of Petri Nets*, pages 154–172. Springer Verlag - LNCS, Vol 616, 1993.

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

[58] E. De Souza e Silva and H.R. Gail. Calculating availability and performability measures of repairable computer systems using randomization. *Journal of the ACM*, 36:171–193, 1989.

[59] L. Esparza and M. Silva. On the analysis and synthesis of free choice systems. *Advances in Petri nets*, LNCS vol. 483:243–286, 1990.

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

[61] R. German. New results for the analysis of deterministic and stochastic Petri nets. In *International Computer Performance and Dependability Symposium - IPDS95*, pages 114–123. IEEE CS Press, 1995.

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

[63] R. German, D. Logothetis, and K. Trivedi. Transient analysis of Markov Regenerative Stochastic Petri Nets: a comparison of approaches. In *6-th International Conference on Petri Nets and Performance Models - PNPM95*, pages 103–112. IEEE Computer Society, 1995.

[64] A. Goyal, S. Lavenberg, and K.S. Trivedi. Probabilistic modeling of computer system availability. *Annals of Operations Research*, 8:285–306, 1987.

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

[66] 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.

[67] S. Haddad and P. Moreaux. Evaluation of high level Petri nets by means of aggregation and decomposition. In *6-th International Conference on Petri Nets and Performance Models - PNPM95*, pages 11–20. IEEE Computer Society, 1995.

[68] S. Haddad and P. Moreaux. Asynchronous composition of high level petri nets: a quantitative approach. In *Proceedings of the 17-th International Conference on Application and Theory of Petri Nets, ICATPN '96*, pages 192–211, Osaka, Japan, june 1996. Springer Verlag - LNCS, Vol 1091.

[69] S. Haddad, P. Moreaux, and G. Chiola. Efficient handling of phase-type distributions in generalized stochastic Petri nets. In P. Azéma and G. Balbo, editors, *Application and Theory of Petri Nets (18-th International Conference), Lecture Notes in Computer Science*, volume 1248, pages 175–194. Springer Verlag, 1997.

[70] B. Haverkort. Matrix-geometric solution of infinite stochastic Petri nets. In *International Computer Performance and Dependability Symposium - IPDS95*, pages 72–81. IEEE Computer Society Press, 1995.

[71] B. Haverkort and A. Ost. Steady-state analysis of infinite stochastic Petri nets: a comparing between the spectral expansion and the matrix-geometric method. In *7-th International Conference on Petri Nets and Performance Models - PNPM97*, pages 36–45. IEEE Computer Society, 1997.

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

[73] A. Heindl and R. German. A fourth-order algorithm with automatic stepsize control for the transient analysis of DSPNs. In *7-th International Conference on Petri Nets and Performance Models - PNPM97*, pages 60–69. IEEE Computer Society, 1997.

[74] W. Henderson, D. Lucic, and P.G. Taylor. A net level performance analysis of stochastic Petri nets. *Journal of Australian Mathematical Soc. Ser. B*, 31:176–187, 1989.

[75] M.A. Holliday and M.K. Vernon. A generalized timed Petri net model for performance analysis. *IEEE Transactions on Software Engineering*, SE-13:1297–1310, 1987.

[76] G. Horton, V. Kulkarni, D. Nicol, and K. Trivedi. Fluid stochastic Petri nets: Theory, application and solution. In *European Journal of Operational Research* - To appear, 1997.

[77] S.M.R Islam and H.H. Ammar. On bounds for token probabilities in a class of generalized stochastic Petri nets. In *Proceedings 3rd International Workshop on Petri Nets and Performance Models - PNPM89*, pages 221–227. IEEE Computer Society, 1989.

[78] H. Kantz and K. S. Trivedi. Reliability modeling of the MARS system: A case study in the use of different tools and techniques. In: Proceedings of the Fourth International Workshop on Petri Nets and Performance Models. Los Alamitos, CA: IEEE Computer Society Press, Dec. 1991.

[79] P. Kemper. Numerical analysis of superposed GSPNs. *IEEE Transactions on Software Engineering*, 22, 1996.

[80] P. Kemper. Reachability analysis based on structured representation. In *Proceedings International Conference on Application and Theory of Petri Nets*, pages 269–288. Springer Verlag - LNCS, Vol 1091, 1996.

[81] P. Kemper. Transient analysis of superposed GSPNs. In *7-th International Conference on Petri Nets and Performance Models - PNPM97*, pages 101–110. IEEE Computer Society, 1997.

[82] K.Trivedi and V. Kulkarni. FSPNs: fluid stochastic Petri nets. In *Proceedings 14-th International Conference on Application and Theory of Petri Nets*, pages 24–31, Chicago, 1993.

[83] V.G. Kulkarni. *Modeling and Analysis of Stochastic Systems*. Chapman Hall, 1995.

[84] V.G. Kulkarni. Fluid models for single buffer systems. In *Frontiers in Queueing: Models, Methods and Problems*. To appear - CRC Press, 1997.

[85] C. Lam and R. Yeh. Optimal maintenance policies for deteriorating systems under various maintenance strategies. *IEEE Transactions on Reliability*, 43, 1994.

[86] A.A. Lazar and T.G. Robertazzi. Markovian Petri net protocols with product form solution. *Performance Evaluation*, 12:67–77, 1991.

[87] 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, 1991.

[88] M. Li and N.D. Georganas. Parametric analysis of stochastic Petri nets. In G. Balbo and G. Serazzi, editors, *Computer Performance Evaluation*. Elsevier Science Publishers, 1992.

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

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

[91] Z. Liu. iperformance bounds for stochastic timed Petri nets. In G. De Michelis and M. Diaz, editors, *Application and Theory of Petri Nets (16-th International Conference), Lecture Notes in Computer Science*, volume 935, pages 316–334. Springer Verlag, 1995.

[92] D. Logothetis, K. Trivedi, and A. Puliafito. Markov regenerative models. In *International Computer Performance and Dependability Symposium - IPDS95*, pages 134–143. IEEE Computer Society Press, 1995.

[93] P. Marenzoni, S. Caselli, and G. Conte. Analysis of large GSPN models: a distributed solution tool. In *7-th International Conference on Petri Nets and Performance Models - PNPM97*, pages 122–131. IEEE Computer Society, 1997.

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

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

[96] M.K. Molloy. Discrete time stochastic Petri nets. *IEEE Transactions on Software Engineering*, SE-11:417–423, 1985.

[97] M.K. Molloy. Fast bounds for stochastic Petri nets. In *International Workshop Timed Petri Nets*, pages 244–249, Torino (Italy), 1985. IEEE Computer Society Press No. 674.

[98] M.K. Molloy. Petri net modelling, the past, the present, the future. In *Proceedings 3rd International Workshop on Petri Nets and Performance Models - PNPM89*. IEEE Computer Society, 1989.

[99] Muppala, J. K. et al.: Dependability modeling of a heterogenous VAXcluster system using stochastic reward nets. In: Avresky, D. R. (ed.) Hardware and Software Fault Tolerance in Parallel Computing Systems. Ellis Horwood Ltd., 1992, pp. 33–59.

[100] T. Murata. Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, 77:541–580, 1989.

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

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

[103] J.L. Peterson. *Petri net theory and the modeling of systems*. Prentice Hall, Englewood Cliffs, 1981.

[104] B.D. Plateau. On the stochastic structure of parallelism and synchronization models for distributed algorithms. In *Proceedings ACM/SIGMETRICS Conference*, pages 147–154, Austin, 1985.

[105] A. Puliafito, M. Scarpa, and K.S. Trivedi. Petri nets with k simultaneously enabled generally distributed timed transitions. In *To appear in: Performance Evaluation*, 1997.

[106] L. Recalde, E. Teruel, and M. Silva. On well-formedness analysis: the case of Deterministic Systems of Sequential Processes. In J. Desel, editor, *Structures in Concurrency Theory*, pages 279–293. Springer Verlag, 1995.

[107] 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.

[108] A. Reibman and K.S. Trivedi. Numerical transient analysis of Markov models. *Computers and Operations Research*, 15:19–36, 1988.

[109] A. Reibman and K.S. Trivedi. Transient analysis of cumulative measures of Markov chain behavior. *Stochastic Models*, 5:683–710, 1989.

[110] R. Sahner, K.S. Trivedi, and A. Puliafito. *Performance and Reliability Analysis of Computer Systems: An Example-based Approach Using the SHARPE Software Package*. Kluwer Academic Publisher, 1995.

[111] W.H. Sanders and J.F. Meyer. Reduced base model construction methods for stochastic activity networks. *IEEE Journal on Selected Areas in Communications*, 9(1):25–36, January 1991.

[112] M. Sereno. Approximate mean value analysis for stochastic marked graphs. *IEEE Transactions on Software Engineering*, 22:654–664, 1996.

[113] M. Sereno and G. Balbo. Computational algorithms for product form solution stochastic Petri nets. In *Proceedings 5th International Workshop on Petri Nets and Performance Models - PNPM93*. IEEE Computer Society, 1993.

[114] M. Silva, L. Recalde, and E. Teruel. Linear algebraic technique for the analysis of liveness of Petri nets. *Journal of Circuits, Systems and Computers*, This same issue, 1997.

[115] R. Smith, K. Trivedi, and A.V. Ramesh. Performability analysis: Measures, an algorithm and a case study. *IEEE Transactions on Computers*, C-37:406–417, 1988.

[116] W.J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.

[117] M. Telek and A. Bobbio. Markov regenerative stochastic Petri nets with age type general transitions. In G. De Michelis and M. Diaz, editors, *Application and Theory of Petri Nets (16-th International Conference), Lecture Notes in Computer Science*, volume 935, pages 471–489. Springer Verlag, 1995.

[118] M. Telek, A. Bobbio, L. Jereb, A. Puliafito, and K. Trivedi. Steady state analysis of Markov regenerative SPN with age memory policy. In H. Beilner and F. Bause, editors, *8-th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation, Lecture Notes in Computer Science*, volume 977, pages 165–179. Springer Verlag, 1995.

[119] M. Telek, A. Bobbio, and A. Puliafito. Steady state solution of MRSPN with mixed preemption policies. In *International Computer Performance and Dependability Symposium - IPDS96*, pages 106–115. IEEE Computer Society Press, 1996.

[120] M. Telek. *Some advanced reliability modelling techniques.* CSc/Phd Thesis, Hungarian Academy of Science, 1994.

[121] Tomek, L. A., Trivedi, K. S.: Fixed point iteration in availability modeling. In: Cin, M. D., Hohl, W. (eds.) Proceedings of the 5th International GI/ITG/GMA Conference on Fault-Tolerant Computing Systems. Berlin: Springer-Verlag, September 1991, pp. 229–240.

[122] C. Valdez-Flores and R.M. Feldman. A survey of preventive maintenance models for stochastically deteriorating single-unit systems. *Naval Research Logistic Quarterly*, 36:419–446, 1989.