The Effect of Execution Policies on the Semantics and Analysis of Stochastic Petri Nets

MARCO AJMONE MARSAN, SENIOR MEMBER, IEEE, GIANFRANCO BALBO, ANDREA BOBBIO, GIOVANNI CHIOLA, GIANNI CONTE, MEMBER, IEEE, AND ALDO CUMANI, MEMBER, IEEE

Abstract—**Petri nets in which random delays are associated with atomic transitions are defined in a comprehensive framework that contains most of the models recently proposed in the literature. The inclusion into the model of generally distributed firing times requires to specify the way in which the next transition to fire is chosen, and how the model keeps track of its past history; this set of specifications is called an execution policy. The paper discusses the impact that different execution policies have on the semantics of the model, as well as the characteristics of the stochastic process associated with each of these policies. When the execution policy is completely specified by the transition with the minimum delay (race policy) and the firing distributions are of the phase type, an algorithm is provided that automatically converts the stochastic process into a continuous time homogeneous Markov chain. Finally, an execution policy based on the choice of the next transition to fire independently of the associated delay (preselection policy) is introduced, and its semantics is discussed together with possible implementation strategies.**

Index Terms—**Markov and semi-Markov processes, performance evaluation, Petri nets, phase-type distributions, stochastic models, stochastic Petri nets.**

I. Introduction

PETRI nets (*PN*) [1]–[4] are becoming increasingly popular as a powerful tool for the description and the analysis of systems that exhibit concurrency, synchronization, and conflicts. Although the basic Petri net model includes no explicit notion of time, several researchers have recently devoted their attention to augmented models that include timing and that are therefore named timed Petri nets [5], [6].

Interpreting Petri nets as state/event models, time is naturally associated with activities that induce state changes, hence with the delays incurred before firing transitions. The choice of associating time with transitions is the most frequent in the literature on timed Petri nets, although other possibilities have been explored. Similarly, a common assumption is that the net sojourns in a given marking for a time that is related to the firing delay of the transitions enabled in that marking. Transition firings are in this paper assumed to be atomic operations, and tokens are consumed from input places and put into output places at the same time instant. Alternative approaches are, however, possible. In [7]–[9] the firing process is split in two phases: a start firing in which tokens are removed from the input places, and an end firing in which tokens are put into output places after some time has elapsed. When random variables are used to specify the firing delays of transitions, timed Petri nets are called stochastic Petri nets (*SPN*). Specifications concerning the policy used to select the enabled transition that fires, as well as the way in which memory is kept of the past history of the net, are required for a correct definition of the semantics of the dynamic behavior of these models. We call this set of specifications an *execution policy*.

Stochastic Petri nets were initially proposed [10]–[12] assuming exponentially distributed firing times and a race execution policy, i.e., selecting to fire the transition whose firing delay is statistically minimum among those of the enabled ones. Under these assumptions the authors proved that the dynamic behavior of the net could be represented by a continuous-time homogeneous Markov chain with state space isomorphic to the reachability graph of the Petri net.

In an attempt to extend the class of stochastic processes representable by stochastic Petri nets, Natkin [11], and Bertoni and Torelli [13] proposed a semi-Markov formulation which is, however, not suited to the modeling of parallel activities due to the total lack of memory after every transition firing.

With the aim of extending the modeling power of stochastic Petri nets, generalized stochastic Petri nets (*GSPN*) were proposed in [14], [15], where two classes of transitions are defined: exponentially timed transitions, which are used to model the random delays associated with the execution of activities, and immediate transitions, which are devoted to the representation of logical actions that do not consume time. Immediate transitions allow the introduction of branching probabilities, independently of the timing specifications. The possibility of specifying branching probabilities was also proposed in [16] using a simpler but less powerful formulation (probabilistic arcs).

The first useful results concerning stochastic Petri nets with generally distributed transition delays are due to Du-

Manuscript received November 17, 1986; revised January 29, 1988. This work was supported in part by the Italian Ministry for Education, and by NATO under Research Grants 012.81 and 280.81.

M. Ajmone Marsan is with the Dipartimento di Scienze dell'Informazione, Università di Milano, via Moretto da Brescia 9, 20133 Milano, Italy.

G. Balbo and G. Chiola are with the Dipartimento di Informatica, Università di Torino, corso Svizzera 185, 10149 Torino, Italy.

A. Bobbio and A. Cumani are with the Istituto Elettrotecnico Nazionale G. Ferraris, strada delle Cacce 91, 10135 Torino, Italy.

G. Conte is with the Istituto di Scienze per l'Ingegneria, Università di Parma, Parma, Italy.

IEEE Log Number 8928284.

gan, Trivedi, Geist, and Nicola [16]. In their definition of extended stochastic Petri nets (*ESPN*) they partition transitions into three classes: exclusive, competitive, and concurrent. Provided that the firing delay of all concurrent transitions is exponentially distributed, and that competitive transitions resample a new firing delay whenever they are enabled, *ESPN* can be mapped onto semi-Markov processes. In [16] the authors also suggest a procedure for analyzing acyclic nets in which generally distributed firing delays may be associated with concurrent transitions as well. The technique can however be used only for the transient analysis of models (which are not ergodic due to the acyclicity of the net). Moreover, as the authors recognize, the analysis becomes too complex for even medium size nets.

Using a similar approach, an embedded Markov chain technique was later proposed for the analysis of deterministic and stochastic Petri nets (*DSPN*) [17]. In this case the embedded Markov chain is used for the computation of the steady-state solution of nets in which at most one concurrent transition associated with a constant delay is enabled in any marking.

Previous works however lack a precise discussion of the impact of execution policies on the semantics and analysis of stochastic Petri net models with generally distributed transition firing delays. In this paper we attempt to give a comprehensive definition of stochastic Petri nets with generally distributed transition firing times, and discuss some possible execution policies as well as the problem of specifying such policies at the net level. Phase-type distributions [18] are given special attention due to the possibility of accounting for their impact directly at the state-space level.

Only methods for the analytical solution of the stochastic process underlying the stochastic Petri nets are discussed in this paper, and the reader is referred to [19] for a comprehensive discussion of the tradeoffs between the simulative and the analytical solution of probabilistic models. The analysis of a stochastic Petri net for the computation of performance results requires the determination of the reachable markings, the construction of the stochastic process with a state space isomorphic to the Petri net reachability graph, and the numerical solution of the process.

The paper is organized as follows. In Section II we introduce the basic notation and definitions. The discussion of the possible execution policies and of the related stochastic processes constitutes the rest of the paper. In Section III the race policy is defined, and its semantics is examined under different conditions. In Section IV the stochastic process derived from the race execution policy is investigated. As a further step toward the model solvability, in Section V the probability distributions associated with stochastic Petri net transitions are restricted to be of phase type. Under this restriction, an algorithm is presented that converts the original non-Markov process into a homogeneous Markov chain defined over an expanded state space. The state expansion is driven by the definition of the execution policies and is performed at the state space level, rather than at the Petri net level, as previously proposed in the literature [11], [12]. Once the expanded transition graph has been obtained, either the time-dependent or the steady-state performance indexes related to the original *SPN* model [20] can be evaluated by solving the state probability equations of a Markov chain. An illustrative example is presented in Section V-C. In Section VI an execution policy (called preselection) based on a time independent selection of the next transition to fire is introduced, and its semantics is discussed in several cases. Finally, possible ways of implementing the preselection policy at the process level or at the net level are discussed.

II. Notation and Basic Definitions

We begin by recalling some definitions relating to Petri net models in order to introduce the notation that will be used throughout the paper. It is assumed that the reader is familiar with the basic Petri net concepts.

A *Petri net* with inhibitor arcs is a tuple $N = (P, T, I, O, H)$ where $P = \{p_1, p_2, \cdots, p_{np}\}$ is a set of *places*, $T = \{t_1, t_2, \cdots, t_{nt}\}$ is a set of *transitions*, and I, O, and H denote bags (or multisets) of *input*, *output*, and *inhibition arcs*. A *marking* M of the Petri net N is an np-component vector of nonnegative integers $M = [m_1, m_2, \cdots, m_{np}]$ whose ith entry represents the number of *tokens* contained in place p_i. A *marked Petri net* is defined by the tuple $PN = (N, M_0)$ where M_0 is the *initial marking*.

For each transition $t_k \in T$, it is possible to represent the input, output, and inhibition bags as three vectors $I_k = [i_{k1}, i_{k2}, \cdots, i_{knp}]$, $O_k = [o_{k1}, o_{k2}, \cdots, o_{knp}]$, and $H_k = [h_{k1}, h_{k2}, \cdots, h_{knp}]$, where i_{kj} is the number of input arcs from place p_j to transition t_k, and similarly for o_{kj} and h_{kj}.

A transition t_k is said to be *enabled* in marking M if and only if $M \geq I_k$ (i.e., iff $m_j \geq i_{kj}$ $\forall j = 1, 2, \cdots, np$) and $m_j < h_{kj}$ for all j such that $h_{kj} > 0$. Let $E(M) \subset T$ be the set of transitions enabled in M. A priority level can be associated with each transition t_k, so that a transition t_k in $E(M)$ can fire only if no higher priority transition t_j is enabled in the same marking. In this way, only transitions with the highest priority level, grouped in a subset $E'(M)$ of $E(M)$ are allowed to fire. For all t_k in $E'(M)$ we define the *firing* as follows:

$$M \xrightarrow{t_k} M' \tag{1}$$

where

$$M' = M - I_k + O_k. \tag{2}$$

A marking M_j is said to be *immediately* (*or one-step*) *reachable* from M_i if and only if there exists a transition t_k in $E'(M_i)$ such that $M_i \xrightarrow{t_k} M_j$.

An *execution sequence* of a marked Petri net is a sequence $ES = (M_{(0)}; (t_{(1)}, M_{(1)}); \cdots; (t_{(j)}, M_{(j)}); \cdots)$ such that $M_{(i-1)} \xrightarrow{t_{(i)}} M(i)$ for any $i = 1, 2, \cdots, j$. Notice that, in an execution sequence ES, the sequence of

transitions together with the first marking uniquely determine the sequence of markings, so that the latter need not be explicitly given. The set of all the execution sequences starting from the initial marking ($M_{(0)} = M_0$) will be denoted by $S(M_0)$.

A marking M_b is said to be *reachable* from M_a if and only if there exists an execution sequence $ES \in S(M_0)$ in which, for some $i < k$, $M_{(i)} = M_a$ and $M_{(k)} = M_b$.

The *reachability set* $R(M_0)$ is the set of all markings reachable from M_0 including M_0 itself. Due to the finiteness of the net, $R(M_0)$ is countable, so that it is possible to write $R(M_0) = \{M_0, M_1, \cdots\}$.

The *reachability graph* $RG(M_0)$ is defined as the labeled directed graph whose vertices are the elements of $R(M_0)$ and such that for each possible transition firing $M_i \xrightarrow{t_k} M_j$ there exists an arc (i, j) labeled k. An execution sequence ES of a marked Petri net can be viewed as an arc progression in $RG(M_0)$.

A *timed execution* TE of a marked PN with initial marking M_0 is an execution sequence ES of $S(M_0)$ augmented by a nondecreasing sequence of real values representing the epochs of firing of each transition such that consecutive transitions $t_{(i)}$, $t_{(i+1)}$ in ES, correspond to ordered epochs $\tau_i \leq \tau_{i+1}$. Thus formally [21]:

$$TE = \big((\tau_0, M_0); (t_{(1)}, \tau_1, M_{(1)}); \cdots (t_{(i)}, \tau_i, M_{(i)}); \cdots\big) \tag{3}$$

The time interval $[\tau_i, \tau_{i+1})$ between consecutive epochs represents the period in which the net sojourns in marking $M_{(i)}$. A timed execution TE truncated at the kth epoch τ_k is called the *history* of the PN up to the kth epoch τ_k and is denoted by $Z(k)$. In the following we always assume (without loss of generality) $\tau_0 = 0$ as initial epoch.

Our main concern lies in the use of PN for the specification of stochastic models of the behavior of systems. We thus focus our attention on the introduction of stochastic timing in PN models. Since we are considering causal systems, we want to be able to describe (at least in a probabilistic sense) the future behavior of a system from the knowledge of the past history. We thus make the following assumption.

Assumption 1: Let $\boldsymbol{Z} = Z(i)$ be a history of the PN up to the ith epoch, and $\boldsymbol{M} = M_{(i)}$ be the marking entered by firing transition $t_{(i)}$. We assume that for all i, $\boldsymbol{Z}$, and $\boldsymbol{M}$, the *firing distribution* $D(x|\boldsymbol{M}, \boldsymbol{Z})$ can be uniquely determined in such a way that its kth component is defined (interpreted) in the following way:

$$D_k(x|\boldsymbol{M}, \boldsymbol{Z}) = \Pr\{t_k \text{ fires, firing delay} \leq x | \boldsymbol{M}, \boldsymbol{Z}\}. \tag{4}$$

The random variable "firing delay" represents the time that elapses from entering $\boldsymbol{M}$ up to the next firing epoch, i.e., the time interval $\tau_{i+1} - \tau_i$. The above distribution must be defined over all transitions t_k in $E'(\boldsymbol{M})$ so that

$$\sum_{k:t_k \in E'(M)} \lim_{x \to \infty} D_k(x|\boldsymbol{M}, \boldsymbol{Z}) = 1. \tag{5}$$

Note that $\boldsymbol{M}$ is known from $\boldsymbol{Z}$, but we have explicitly indicated the dependence on $\boldsymbol{M}$ since in many cases $\boldsymbol{M}$ is the only element that actually influences this distribution function.

The probability $p_k(\boldsymbol{M}, \boldsymbol{Z})$ of selecting t_k to be the next transition to fire is obtained as:

$$p_k(\boldsymbol{M}, \boldsymbol{Z}) = \lim_{x \to \infty} D_k(x|\boldsymbol{M}, \boldsymbol{Z}) = \Pr\{t_k \text{ fires} | \boldsymbol{M}, \boldsymbol{Z}\} \tag{6}$$

and the distribution of the time spent in marking $\boldsymbol{M}$ before the next epoch is found as:

$$F(x|\boldsymbol{M}, \boldsymbol{Z}) = \sum_{k:t_k \in E'(M)} D_k(x|\boldsymbol{M}, \boldsymbol{Z}) = \Pr\{\text{firing delay} \leq x | \boldsymbol{M}, \boldsymbol{Z}\}. \tag{7}$$

We can now give the following definition.

Definition: A stochastic PN (SPN) is a marked PN in which:

D1) With any transition $t_k \in T$ is associated a random variable θ_k modeling the time needed by the activity represented by t_k to complete.

D2) Each random variable θ_k is characterized by the cumulative distribution functions $G_k(x|\boldsymbol{M})$ of the firing time of the transition in isolation.

D3) An execution policy is defined for inferring the probability measures $\{D_k(x|\boldsymbol{M}, \boldsymbol{Z})\}$ on the set of all the timed execution sequences TE. The execution policy consists of two parts: the way in which, for each marking $\boldsymbol{M}$, a transition that belongs to $E'(\boldsymbol{M})$ is selected to fire, and the way in which trace is kept of the past history.

D4) An initial probability distribution on $R(M_0)$ is defined.

With the above definition, the set of possible executions of an SPN, together with the probability measure induced on it by assigning the firing distribution of Assumption 1, constitutes a continuous-time discrete-state stochastic point process.

The state space of the stochastic process is not necessarily isomorphic to the underlying PN reachability graph, since the timing constraints superimposed to the firing rules may alter the set of possible execution sequences. In particular, some of the firing probabilities of (6) might reduce to zero, thus raising some subtle probabilistic problem. In order to avoid these problems we restrict the class of the allowed distribution functions by introducing the following assumption.

Assumption 2: The probabilities $p_k(\boldsymbol{M}, \boldsymbol{Z})$ of (6) satisfy the following condition:

$$p_k(\boldsymbol{M}, \boldsymbol{Z}) > 0 \qquad \forall\, k: t_k \in E'(\boldsymbol{M}). \tag{8}$$

The definition of SPN together with Assumptions 1 and 2 guarantee the isomorphism between the stochastic process state space and the reachability graph of the underlying PN. This isomorphism is a very desirable property, since it permits to study the structural properties of the model (boundedness, reproducibility, deadlock-freeness, invariants, etc.) using classical net theory techniques, in-

dependently of the time specification, that only affects the *SPN* performance measures. It should be observed that under the firing mechanism assumed in [7]–[9] the isomorphism is not maintained.

For what concerns the initial probability distribution, we assume throughout the paper that the system is in state M_0 at $\tau = 0$ with probability one.

III. Race Execution Policy

Recalling the definition of an *SPN*, the most natural way for choosing the next transition to fire, is to select the enabled transition whose associated delay is statistically the minimum. We call this model *race* (or concurrent [22]) model. In Section VI we shall also introduce a second execution policy called *preselection*. Under this latter policy, the choice of the next transition to fire does not depend on the time delay of the associated activity, but is independently assigned by a probability mass function. A combination of race and preselection policies will also be considered as a useful generalization for the practical application domain.

A. Race SPN Definition

When the net enters marking $\boldsymbol{M}$, a random sample is extracted from the joint distribution:

$$\Phi(x_1, x_2, \cdots \mid \boldsymbol{M}, \boldsymbol{Z}) = \Pr\{\theta_1 \le x_1, \theta_2 \le x_2, \cdots \mid \boldsymbol{M}, \boldsymbol{Z}\} \quad (9)$$

where the θ_k are random variables representing, for each transition $t_k \in E'(\boldsymbol{M})$, the time till firing measured from the epoch at which $\boldsymbol{M}$ was entered. The θ_k for which the sampled value is minimum determines which transition will actually fire, and the sojourn time in $\boldsymbol{M}$ equals this minimum sampled value. We only consider the case in which all random variables θ_k are independent, so that (9) is uniquely determined by the marginal distributions:

$$\Phi_k(x \mid \boldsymbol{M}, \boldsymbol{Z}) = \Pr\{\theta_k \le x \mid \boldsymbol{M}, \boldsymbol{Z}\}. \quad (10)$$

In order to satisfy conditions (5) and (8), it is sufficient to assume that these distribution functions are differentiable (at least in the sense of generalized functions), honest distributions whose derivative with respect to x is a probability density function with infinite support $[0, \infty)$.

The kth component of the firing distribution (4) is then computed as:

$$D_k(x \mid \boldsymbol{M}, \boldsymbol{Z}) = \int_0^x \left[\prod_{\substack{j: t_j \in E'(\boldsymbol{M}) \\ j \neq k}} \left[1 - \Phi_j(u \mid \boldsymbol{M}, \boldsymbol{Z})\right] \right] d_u \Phi_k(u \mid \boldsymbol{M}, \boldsymbol{Z}). \quad (11)$$

In this case,

$$p_k(\boldsymbol{M}, \boldsymbol{Z}) = \int_0^\infty \left[\prod_{\substack{j: t_j \in E'(\boldsymbol{M}) \\ j \neq k}} \left[1 - \Phi_j(x \mid \boldsymbol{M}, \boldsymbol{Z})\right] \right] d_x \Phi_k(x \mid \boldsymbol{M}, \boldsymbol{Z}) \quad (12)$$

and

$$F(x \mid \boldsymbol{M}, \boldsymbol{Z}) = 1 - \prod_{k: t_k \in E'(\boldsymbol{M})} \left[1 - \Phi_k(x \mid \boldsymbol{M}, \boldsymbol{Z})\right]. \quad (13)$$

For the solution of the model it is necessary to obtain the distributions $\Phi_k(x \mid \boldsymbol{M}, \boldsymbol{Z})$ for each transition of the *SPN*, given the $G_k(x, \boldsymbol{M})$.

B. Conditioning on Past History Z

Different ways for keeping track of the past behavior of the net are possible. We consider the following three alternatives:

Resampling: The firing distribution $\{D_k(x \mid \boldsymbol{M}, \boldsymbol{Z})\}$ is independent of $\boldsymbol{Z}$, but it may depend upon the current marking $\boldsymbol{M}$.

Age Memory: The firing distribution depends on the past history $\boldsymbol{Z}$ through the concept of a *work age variable* associated with each transition t_k. The age variable associated with transition t_k accounts for the work performed by the corresponding activity since the time of the last firing of the same transition; the kth component $D_k(x \mid \boldsymbol{M}, \boldsymbol{Z})$ of the firing distribution depends on the distribution of the residual time needed for this activity to complete.

Enabling Memory: The firing distribution depends on the past history $\boldsymbol{Z}$ through the concept of an *enabling age variable* associated with each transition t_k. The age variable associated with transition t_k accounts for the work performed by the corresponding activity since the beginning of the last period during which it has been enabled; the kth component $D_k(x \mid \boldsymbol{M}, \boldsymbol{Z})$ of the firing distribution depends on the distribution of the residual time needed for this activity to complete.

The race model with resampling (we shall refer to this case as *R-R*) can be used to describe the behavior of a set of parallel competing activities (modeled by conflicting transitions) such that the first one to terminate determines a change in the system state. The work performed by those activities that do not complete is lost; the only work that is relevant for the model is that performed by the activity corresponding to the transition that fired leading to a change of state of the system. An example might be the parallel execution of hypotheses tests. The process that terminates first is the one that has verified a hypothesis. Those hypotheses whose verification was not completed need not be remembered; furthermore, it is not important to keep track of the point at which these other tests were interrupted since they are not going to be resumed. This model appears to be interesting only in the case of conflicting transitions. When we use this model in the case of concurrent transitions we easily obtain paradoxical system behaviors [23]. Even if the *R-R* policy is clearly of little interest in practical applications, we considered it because this policy was implicitly assumed in previous attempts to extend the class of *SPN* to general time distributions in the framework of semi-Markov processes [11], [13].

The race model with age memory (*R-A*) may be used to describe the behavior of a set of simultaneous activities

such that the first activity to terminate determines a change in the system state. In this case, however, the work performed by those activities that do not complete is not lost. This implies that simultaneous activities are such that, after the firing of the first transition that completes, they will resume from the point at which they were interrupted, in the first marking that enables them; they may continue in the new state if they are still enabled. An example in this case can be provided by a set of tasks in a multiprocessor system that operates in a message passing fashion. Once a process completes, it issues a message, and thus the system state is changed. The other processes continue to execute after the change of state if the corresponding transitions are still enabled. Otherwise they will resume, in the first marking that enables them, from the point at which they were interrupted. This model appears to be well suited to represent both cases of conflicting and concurrent activities.

The race model with enabling memory (R-E) is again used to describe the behavior of a set of simultaneous activities such that the first activity that terminates determines a change in the system state. In this case the work performed by those activities that do not complete is lost, unless the transitions to which the activities correspond remain enabled in the new marking generated by the change of state. An example in this case can be provided by the behavior of an alternating bit protocol. Consider the activity related to the measure of the timeout between the transmission of a packet and its retransmission if no acknowledgment arrives. This activity starts when a packet is transmitted, and it must continue in all markings that enable it, regardless of any change in system state, unless it is preempted by the arrival of an acknowledgment packet. In this case the activity is stopped, and when restarted (for a new packet) it must be reinitialized. For conflicting transitions R-R and R-E behave identically, however R-E allows the modeling of parallel activities with preemption disciplines.

The conditioning on $\boldsymbol{Z}$ is a property of each single transition of the net and an *SPN* model may contain transitions belonging to all the three types of conditioning.

IV. The Stochastic Process Associated with a Race SPN

The marginal distributions $\Phi_k(x \mid \boldsymbol{M}, \boldsymbol{Z})$ of (10) must be computed from the individual distributions $G_k(x \mid \boldsymbol{M})$ and from the knowledge of the memory policy associated with t_k. For this purpose, we attach an age variable a_k to each transition t_k. The way in which a_k accounts for the work done by the activity corresponding to transition t_k is a function of the memory policy:

- *Resampling:* a_k is reset to zero at any change of state.
- *Age Memory:* a_k accounts for the activity performed by t_k since its last firing.
- *Enabling Memory:* a_k accounts for the activity performed by t_k since the last epoch at which it has become enabled; it is reset to zero at any change of state that disables transition t_k.

A. Marking Dependence

The derivation of the marginal distributions $\Phi_k(x \mid \boldsymbol{M}, \boldsymbol{Z})$ from the individual firing time distributions $G_k(x \mid \boldsymbol{M})$ associated with each transition of the net, must account for the way in which these last distributions depend on the marking of the net. In general the specification of these distributions requires an *a priori* knowledge of the reachability set of the net. Special cases exist in which the description of this marking dependence can be provided at the net level.

In the following we discuss three different cases of marking dependence that we consider useful for the specification of our models.

1) The Individual Distributions Do not Depend on Marking:

$$G_k(x \mid \boldsymbol{M}) = G_k(x). \tag{14}$$

The marginal distributions in (10) are the residual life distributions of t_k conditioned on the transition age a_k:

$$\Phi_k(x \mid \boldsymbol{M}, \boldsymbol{Z}) = G_k^r(x \mid a_k) = \frac{G_k(x + a_k) - G_k(a_k)}{1 - G_k(a_k)}. \tag{15}$$

2) Marking Dependence Through a Scaling Factor: In this case a_k depends both on the time interval spent in each marking in which t_k was enabled without firing and on the distribution $G_k(x \mid \boldsymbol{M})$ of t_k in that particular marking. We consider only the possibility that the dependence is reflected on the distribution by means of a scaling factor $\beta(\boldsymbol{M}) > 0$, so that

$$G_k(x \mid \boldsymbol{M}) = \mathrm{G}_k(\beta(\boldsymbol{M})x). \tag{16}$$

As a possible instance of the process, let us suppose that transition t_k is first enabled in marking M_{i1}, in which it spends x_1 time units; then, without firing, it enters marking M_{i2}. The age a_k of t_k in M_{i2} given it has worked x_1 time units in M_{i1} is computed from the following equation:

$$a_k = x_1 \frac{\beta(M_{i1})}{\beta(M_{i2})} \tag{17}$$

where $x_1\beta(M_{i1})$ can be interpreted as a measure of the quantity of work performed before the change of state in marking M_{i1}.

The marginal distributions in (10) are thus formally expressed by an equation similar to (15) where the residual life distributions of each transition t_k at time x is obtained by introducing the transition age a_k calculated using (17):

$$\Phi_k(x \mid \boldsymbol{M}, \boldsymbol{Z}) = \Phi_k(x \mid x_1, M_{i1}, x_2, M_{i2}, \cdots) = G_k^r(x \mid a_k). \tag{18}$$

3) More General Kind of Marking Dependence: The marking dependence discussed above is well suited to represent the case of a change of speed in the execution of an activity consequent to a change of the net marking. More general cases are not difficult to conceive, but are difficult to include in a single general framework. This

does not mean that they are intractable when the underlying behavior of the net is well understood. For example, consider a transition t_k with a single input place p_i which models the execution of some activity. The net is such that if several tokens reside in p_i each one of them represents an activity that can proceed in parallel with the others (t_k is an "infinite server" transition that models random delays without congestion); the execution policy be *R-A*, and the distribution associated with the activity be an Erlang of order 2. Assume now that one token is in p_i for some time while the net is in marking M_i, and that some other transition fires changing the marking to M_j such that another token enters p_i. The firing distribution of t_k in the newly entered marking must be calculated as the distribution of the minimum between the residual firing time associated with the activity that was already in progress, and the whole activity duration. Thus, while in M_i the firing distribution is Erlang of order 2, in M_j it has a different form, that can be calculated from the knowledge of the model semantics.

B. Characterization of the Stochastic Process

The age variables a_k defined above coincide with the supplementary variables [24] which make the stochastic process Markovian with partly discrete [states in $R(M_0)$] and partly continuous (the Cartesian product of the age variables domains) state space.

In case all transitions of the net are of the *R-R* type, since at any state change all a_k are reset to zero, the associated process becomes semi-Markov, with one-step transition probability matrix $Q(x)$, whose entries are given by:

$$q_{ij}(x) = \begin{cases} D_k(x \mid M_i) & \text{if } M_i \xrightarrow{t_k} M_j \\ 0 & \text{otherwise} \end{cases} \qquad (19)$$

with $D_k(x \mid M_i)$ given by (11).

It should however be emphasized that the numerical calculation of the measures pertinent to the process is quite intractable as long as the individual distributions are of generic form, even in the semi-Markov process of (19) for which closed form expressions exist in the Laplace transform domain [25].

V. Race SPN with Firing Distributions of Phase Type

When the transition firing distributions are of *phase type* (*PH*) [18], the reachability graph $RG(M_0)$ of the *PN* can be expanded into a discrete-state transition graph over which a continuous-time homogeneous Markov chain, equivalent to the original non-Markov process, can be defined. The measures pertinent to the original process can then be evaluated by solving the expanded Markov chain.

PH distributions are the distributions of the time till absorption of continuous-time homogeneous Markov chains with at least one absorbing state. Their most interesting feature, from our point of view, is the representation as finite-state Markov chains, that allows a natural discretization of the age variable introduced in Section III.

The simplest subclasses of *PH* distributions, like Erlang, hyperexponential (and trivially exponential), are commonly encountered in various areas of applied stochastic modeling. The discussion in this section thus also covers the simplest case in which all the $G_k(x \mid M)$ are exponential. In this case the stochastic process obtained with policies *R-R*, *R-A*, and *R-E* is the same due to the memoryless property of the exponential distribution, and reduces to a continuous time Markov chain [11], [12] isomorphic to the *SPN* reachability graph $RG(M_0)$.

PH distributions have been already mentioned, in connection with *SPN* models, by Natkin [11] and Molloy [12]. The suggestion of these authors was to incorporate the *PH* model into the original *SPN*, by replacing a timed, *PH* distributed, transition with a proper *SPN* subnetwork whose reachability graph yields the Markov graph of the given *PH* distribution.

This approach suffers from at least two drawbacks. First, the complexity of the *SPN* is artificially increased by fictitious nodes (both places and transitions) which do not refer to the operation of the system, but only to the distributions of the firing times. In the resulting reachability set the markings representing physical conditions of the system are dispersed among markings representing the passage of a transition through the succession of exponential stages.

The second drawback is that the proper subnetwork can be easily drawn only in the following cases:

- when the transition to be expanded is never enabled simultaneously to other transitions;
- when the *R-A* and *R-E* policies are used, and the transitions to be expanded never happen to be in conflict with other transitions.

In all other cases the subnetwork expansion would require the introduction of additional interconnections as we shall demonstrate with the following examples.

A. Examples of Expansion of PH Distributions at the Net Level

Consider the *SPN* of Fig. 1, comprising five places and three timed transitions. An Erlang distribution of order 2 is associated with the two conflicting transitions t_1 and t_2, whereas an exponential distribution is associated with transition t_3.

The fact that the *SPN* graph is not connected should not surprise the reader: we wish to provide the simplest possible example in which both conflicts and concurrency are represented. On the other hand, the *SPN* of Fig. 1 can be viewed as a portion of a larger *SPN* model.

The direct expansion at the *SPN* level of the two non-exponential distributions yields the *SPN* of Figs. 2, 3, and 4, for the three policies *R-A*, *R-E*, and *R-R*, respectively.

The *SPN* of Fig. 2 (*R-A* policy) comprises nine places and five transitions. Four places and two transitions were added to the original *SPN* to represent the two Erlang distributions. Transitions t_{k1} and t_{k2} represent the two exponential stages of the Erlang distribution associated with transition t_k. Places p_a and p_d are used to guarantee that

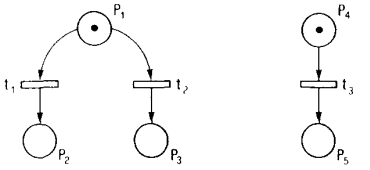


Fig. 1. Simple example of a stochastic Petri net. Delays associated with transitions t_1 and t_2 have Erlang distributions, whereas an exponentially distributed delay is associated with t_3.

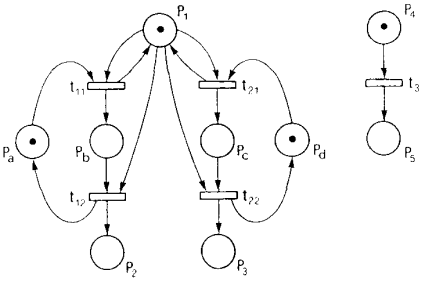


Fig. 2. Expansion of the *SPN* in Fig. 1 in the case of *R-A* policy.

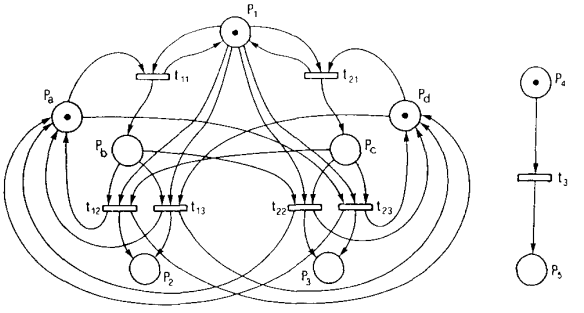


Fig. 3. Expansion of the *SPN* in Fig. 1 in the case of *R-E* policy.

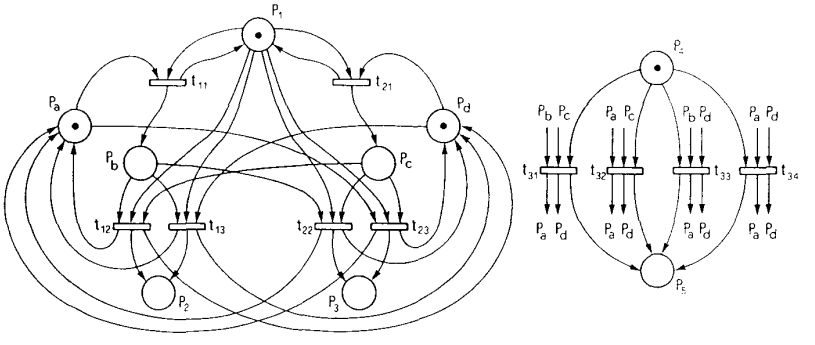


Fig. 4. Expansion of the *SPN* in Fig. 1 in the case of *R-R* policy.

one token at a time enters the subnet that represents the expansion of the Erlang distribution into exponential stages. Note that no connection is in this case necessary with transition t_3.

The *SPN* of Fig. 3 (*R-E* policy) still comprises nine places, but the number of transitions is increased to seven, and many more arcs appear in the net. Transitions t_{k2} and t_{k3} represent the second exponential stage associated with transition t_k. In this case it is necessary to use two different transitions for each second stage since we must reset the age variable associated with the conflicting transition that did not fire. Also in this case no connection is necessary with transition t_3.

The *SPN* of Fig. 4 (*R-R* policy) again comprises nine places, but ten transitions must be used in this case. This is due to the fact that it is now necessary to connect the two separate subnetworks to represent the resampling of transitions t_1 and t_2 whenever transition t_3 fires. For this reason transition t_3 splits into four parallel transitions whose inputs correspond to the four possible conditions of the two Erlang distributions. The firing of any one of these four transitions resets the age variables associated with the two transitions t_1 and t_2.

Obviously, when the individual *PH* distributions of the firing times have order greater than 2 the corresponding *SPN* grows in complexity. When the *R-R* policy is used in the model, the subnetwork expansion is made practically impossible by the necessity of properly interconnecting all the subnetworks referring to concurrent transitions. In a real system concurrency may arise in any

marking in very different locations of the *SPN*. *A priori* we do not know where concurrency arises before having investigated the whole reachability set; thus we are not able to expand the *SPN*, that is to establish the correct connections between the subnetworks in any reachable marking.

B. Expansion of PH Distributions at the State Space Level

When needed, the stage expansion is conveniently performed on the *SPN* reachability graph $RG(M_0)$, knowing for each transition its *PH* model which is itself a labeled directed graph.

The only limitation on the *PH* distributions $G_k(x \mid \boldsymbol{M})$ is that they should be given in a canonical form (always existing, as shown in [26]) in which there is a single initial state (a single state with initial probability equal to one) numbered state 1, and a single final absorbing state numbered $v_k + 1$, where v_k is the order of the *PH* distribution associated with t_k. A transition t_k is said to fire when it reaches the absorbing state $v_k + 1$ in its *PH* model.

Each marking of the original reachability graph $RG(M_0)$ is expanded into a group of submarkings to account for the possible stage of firing reached by all the transitions of the net. Each submarking of M is defined as a pair $(M; W)$ where W is a vector of n_t integers whose kth component $1 \leq w_k \leq v_k$ represents the stage of firing of transition t_k. The pairs $(M; W)$ are the states of an expanded Markov chain equivalent to the original process. With this new notation, the initial state of the process is assumed to be $(M_0; 1, \cdots, 1)$.

The connections among the states resulting from the expansion are made according to the firing policies used in the net. The expansion of the reachability graph can be performed automatically by an enumeration algorithm in which the various firing policies can be accommodated [27]. The equivalence between the Markov chain resulting from the expansion algorithm and the original process defined over the reachability graph $RG(M_0)$ was shown in [28].

An informal description of the expansion algorithm is the following. Assume that a general state $(M; W)$ corresponding to marking M is such that transition t_l ($l = 1, 2, \cdots, nt$) is in stage w_l of its *PH* model. Consider a transition t_k enabled in M, for each α such that there exists an arc $w_k \rightarrow \alpha$ (with rate λ) in the *PH* model of t_k, we generate a new pair $(M; W')$ with $w_k' = \alpha$ and $w_j' = w_j$ for $j \neq k$.

Two cases may arise:

1) $w_k' < v_k + 1$; in this case the pair $(M; W')$ is a state of the process representing the net still being in M, but with t_k in a different stage of firing.

2) $w_k' = v_k + 1$; this condition represents, by definition, the firing of t_k. Therefore, $(M; W')$ is renamed as $(M'; W'')$ with $M \xrightarrow{t_k} M'$ and W'' dependent upon the assumed memory policy of each transition. If $t_j \notin E'(M)$ we have $w_j'' = w_j'$; $w_k'' = 1$, since, by definition, an activity that ends, restarts from its initial state. Finally, according to Section IV-A, we have three subcases for each $t_j \in E'(M)$, $j \neq k$:

2.1) *resampling:* $w_j'' = 1$, i.e., t_j is reset to its initial state;

2.2) *age memory:* $w_j'' = w_j'$, i.e., t_j keeps track of the stage of firing reached in the former marking;

2.3) *enabling memory:* $w_j'' = 1$ if t_j is no longer enabled in M', $w_j'' = w_j'$ if on the contrary, t_j is still enabled in M'.

We also add an arc with rate λ from the original state to the new one.

By repeating the above procedure for each state, we finally obtain the expanded transition graph.

The exact number of states in the expanded process depends both on the structure of the *PN* and on the execution policy. For the *R–R* policy this number is given by:

$$N_s(R\text{–}R) = \sum_{i=1}^{N} \prod_{k:t_k \in E'(M_i)} v_k. \tag{20}$$

In the cases *R–E* and *R–A* the number of states is not easy to compute without actually building the expanded graph; an upper bound is, however, given by:

$$N_s(R\text{–}E) \leq N_s(R\text{–}A) \leq N \prod_{k:t_k \in T} v_k. \tag{21}$$

This number can be quite large even for small nets; in practical cases the actual number of states can be expected to be much lower than the bound expressed by (21), however, it is not hard to build simple examples in which this upper bound is attained. Note that for both *R–E* and *R–A* cases, the value represented by (20) is a lower bound on the number of states after the expansion.

Once the expanded transition graph has been obtained, either the time-dependent or the steady state performance indices related to the original *SPN* model [20] can be evaluated by solving the state probability equations of a Markov chain.

We finally stress that the use of the procedure outlined in this section requires only the specification of the *SPN*, and of the *PH* models for each transition. The subsequent steps, which consist of:

- the search for the reachability graph;
- the reachability graph expansion;
- the solution of the resulting Markov chain;
- the evaluation of the relevant process measures;

can be completely automated, thus making transparent to the analyst the associated mathematics [27].

C. Example of Construction of the Expanded Markov Graph

As an example, we consider the problem of building the expanded Markov graph for a quite simple net, which however presents both situations of parallelism and conflict (Fig. 5).

This net is obtained from that in Fig. 1, by adding transitions t_4 and t_5; they make the net live, strictly conservative, and ergodic, so that the steady-state analysis can be performed. As before, the firing times of conflicting

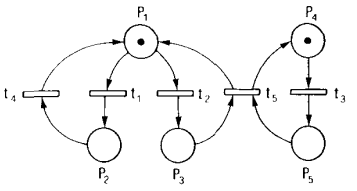


Fig. 5. Simple example of a live, strictly conservative, and ergodic stochastic Petri net. Delays associated with transitions t_1 and t_2 have Erlang distributions, whereas exponentially distributed delays are associated with all other transitions.

transitions t_1 and t_2 are assumed to have Erlang distributions of order 2; the firing times of all other transitions have exponential distributions.

The reachability set of this *SPN* comprises the 6 markings of Table I; Fig. 6 shows the corresponding reachability graph.

Applying the expansion algorithm outlined in Section V-B, under the assumptions of resampling (*R-R*), enabling memory (*R-E*), and age memory (*R-A*), we obtain the Markov chains depicted in Figs. 7, 8, 9, respectively. The nodes of these graphs correspond to states of the expanded process, and are labeled (m; w_1, w_2) where m is the number of the marking and w_1, w_2 represent the stage of firing of t_1 and t_2, respectively (note that the stage of firing of the other transitions need not be reported, since their firing delays have memory-less distributions). Dashed lines encase the groups of Markov states corresponding to the same marking of the net, thus representing the same observable state of the system.

It may be noted that in this example, the graphs for cases (*R-R*) and (*R-E*) comprise the same number of nodes (12) and of arcs (25), but the connections between states are different. The graph for the age memory case (*R-A*) has instead a larger number of both nodes (16) and arcs (30). This is due to the fact that in this case the markings in which t_1 and t_2 are not enabled (i.e., M_2, M_3, M_5, and M_6) are split in order to keep memory of the stage of firing reached by either t_1 or t_2. The number of expanded states is however much less than the upper bound given by (21) that in this case would be 24.

The comparison of the state-transition rate diagrams of these three cases outlines the qualitative difference among the corresponding stochastic processes. The importance of this difference can be stressed by assigning numerical values to the model parameters. Assume, for instance, that a relevant process measure be the sum of the average numbers of tokens in places p_1 and p_2 at steady state. Table II presents this measure computed under the three policies considered when all firing times have the same average value. For the sake of comparison the table also shows the value obtained under the simplifying assumption of exponential (*EXP*) distributions for t_1 and t_2 for which different execution policies lead to the same results.

Even in this extremely simple case, significant differences in the result values can be observed for different policies. Note that the difference between the *R-R* and the

TABLE I
REACHABILITY SET OF THE *SPN* IN FIG. 5

State #	Marking				
	p_1	p_2	p_3	p_4	p_5
1	1	0	0	1	0
2	0	1	0	1	0
3	0	0	1	1	0
4	1	0	0	0	1
5	0	1	0	0	1
6	0	0	1	0	1

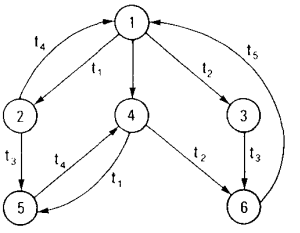


Fig. 6. Reachability graph of the *SPN* in Fig. 5.

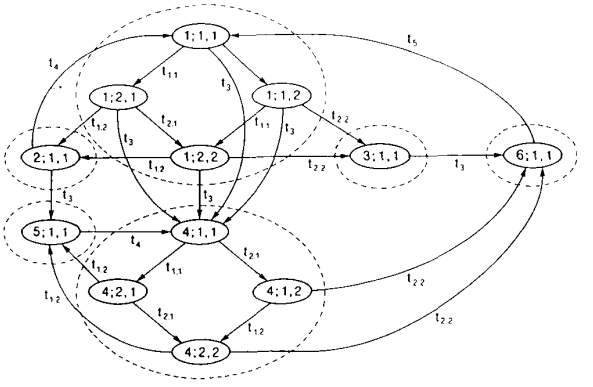


Fig. 7. Markov chain underlying the *SPN* in Fig. 5 in the case of the race with resampling (*R-R*) policy.

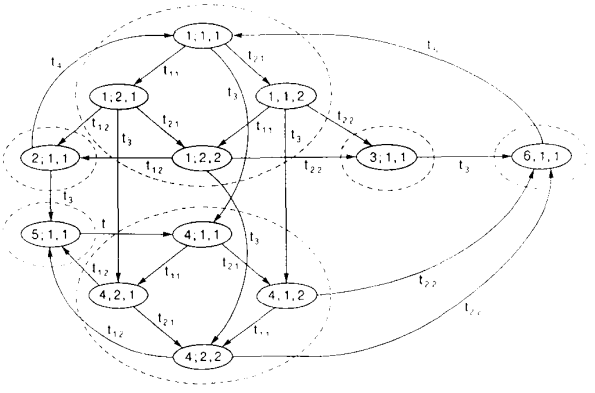


Fig. 8. Markov chain underlying the *SPN* in Fig. 5 in the case of the race with enabling memory (*R-E*) policy.

R-A cases is quite larger than that between *R-A* and the simplified exponential case. This fact is of particular interest since the resampling assumption leads to closed form results [23] that could suggest its use as an approx-

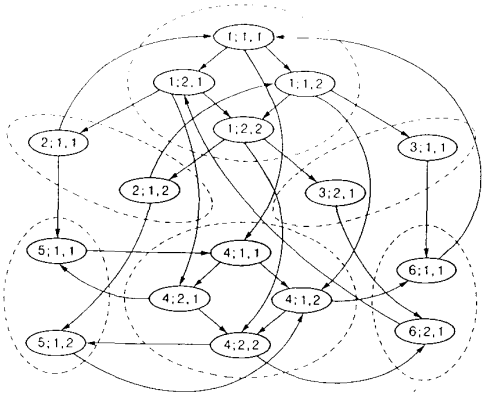


Fig. 9. Markov chain underlying the *SPN* in Fig. 5 in the case of the race with age memory (*R-A*) policy.

TABLE II
SUM OF THE AVERAGE NUMBER OF TOKENS IN PLACES p_1 AND p_2 IN FIG. 5

case	measure
R–R	0.6350
R–E	0.6274
R–A	0.6028
EXP	0.5882

imation for the other policies. Unfortunately, already from this very simple example we can conclude that this is not recommendable. Indeed, should the age memory policy (*R-A*) be the correct representation of the system behavior, using the resampling policy in order to simplify the computations would yield a result much less accurate than the one provided by a crude all-exponential approximation.

VI. Preselection Policies

As already pointed out in the definition of *SPN*, an execution policy is a set of specifications that allow the selection of the transition that will actually fire among those in $E'(\boldsymbol{M})$. The race policy defines a rule in which such a selection is based on transition timing. Other rules can be used for the same purpose. It is indeed possible to perform the selection on the basis of additional specifications that do not depend on the duration of the activities associated with the transitions that are enabled. One possibility is that of defining a probability mass function over the set of enabled transitions $E'(\boldsymbol{M})$, and of using this function to choose the transition that fires next. This policy models situations in which different alternatives exist, and only one of these is chosen on the basis of time-independent information. The semantics of the model is such that the activities with transitions that are enabled, but are not preselected, are not executed. This means that the age variables associated with these transitions do not increase.

A. Global Preselection Policy

The idea of preselecting the enabled transition that is going to fire next, if applied to all the markings of the net, leads to what we call a *global preselection* policy. In this case all the activities represented in the net are serialized, allowing in any marking $\boldsymbol{M}$ only one of the enabled activities to execute.

Equation (4), which defines the probability of firing next for a certain transition (say transition t_k), can be rewritten in the following form:

$$D_k(x \mid \boldsymbol{M}, \boldsymbol{Z}) = p_k(\boldsymbol{M}, \boldsymbol{Z}) F_k(x \mid \boldsymbol{M}, \boldsymbol{Z}) \qquad (22)$$

where $F_k(x \mid \boldsymbol{M}, \boldsymbol{Z})$ are honest distributions for all transitions t_k in $E'(\boldsymbol{M})$ and represent the distributions of the firing delays conditioned on $\boldsymbol{M}$, $\boldsymbol{Z}$, and on the fact that t_k is the transition that will actually fire. In the global preselection policy, (22) is interpreted as follows: when the *SPN* enters marking $\boldsymbol{M}$, a transition is selected among those in $E'(\boldsymbol{M})$ according to the probability mass function $\{p_k(\boldsymbol{M}, \boldsymbol{Z})\}$; the selected transition, say t_k, will then fire after a random delay with distribution $F_k(x \mid \boldsymbol{M}, \boldsymbol{Z})$. Thus the selection of the transition that actually fires does not depend upon the associated delay; conversely, once a transition is selected, the sojourn time in $\boldsymbol{M}$ does not depend upon the delays associated with other transitions.

For the solution of a model which uses this policy it is necessary to know for each transition t_k both the probability $p_k(\boldsymbol{M}, \boldsymbol{Z})$ and the distribution $F_k(x \mid \boldsymbol{M}, \boldsymbol{Z})$. The way in which the conditioning on the past history $\boldsymbol{Z}$ is accounted for in the analytical formulation of the process assumes that in this context the probability measures $p_k(\boldsymbol{M}, \boldsymbol{Z})$ and $F_k(x \mid \boldsymbol{M}, \boldsymbol{Z})$ are independent of $\boldsymbol{Z}$, but they may depend on the current marking $\boldsymbol{M}$; thus in (13) $F_k(x \mid \boldsymbol{M}, \boldsymbol{Z}) = F_k(x \mid \boldsymbol{M})$ and $p_k(\boldsymbol{M}, \boldsymbol{Z}) = p_k(\boldsymbol{M})$. Following the classification introduced in Section III, the conditioning upon the past history is in this case implicitly assumed of resampling type.

By definition $F_k(x \mid \boldsymbol{M}) = G_k(x \mid \boldsymbol{M})$; it thus follows that the transition probability matrix among markings can be expressed as:

$$\Pr\{M_i \to M_j \text{ occurs before } x \mid M_i \text{ at } 0\} = \begin{cases} p_k(M_i) G_k(x \mid M_i) & \text{if } M_i \xrightarrow{t_k} M_j \\ 0 & \text{otherwise.} \end{cases} \qquad (23)$$

The stochastic process associated with an *SPN* with global preselection policy is semi-Markov with one-step transition probability matrix as in (23).

The association of memory with the distribution of the age variable (either of type A or E) seems not to extend the power of the model [23].

The global preselection policy can be used to model a set of conflicting activities that cannot be performed in parallel. When the system enters a new state, it must choose which one of the conflicting activities will be executed next. Once the choice is made, the selected activity is performed until completion, and then the system state changes. An example can be provided by a processor that must sequentially execute a set of jobs. Once a process is completed, it is necessary to choose which process will be executed next. This process is then run until completion.

B. Local Preselection Policy

The global preselection policy discussed before, implies a complete serialization of all the activities described by transitions. No parallelism is thus naturally representable in the model. Moreover, the specification of the policy requires that the probability mass function $p_k(\boldsymbol{M})$ be given for every possible reachable marking of the net. Except for special cases where the mass functions $p_k(\boldsymbol{M})$ are obtained by default assumptions with doubtful modeling power (e.g., assuming that all enabled transitions may be selected with equal probability), the specification of the model is very cumbersome and requires the *a priori* knowledge of the whole reachability set $R(M_0)$ of the net.

A limited use of the preselection policy in conjunction with the race policy, may yield models that are easier to construct, and that allow a certain degree of parallelism to be explicitly represented in the net. For this purpose we can assume that the transitions of the net are grouped in not necessarily disjoint sets within which a preselection policy is applied. We call these sets *local preselection sets*. In a marking that enables transitions belonging to these sets, the next transition to fire is identified by selecting first, with time independent criteria, an enabled transition (if there exists one) from each of the sets, and then by choosing among the preselected transitions the one whose firing delay is minimal. The age variables associated with the preselected transitions can be managed with any of the policies described for the race case.

As a possible example of application of such a policy, we may consider a multiprocessor system where several processing units execute tasks submitted from the outside world. Submitted tasks that may belong to classes are allocated to the processing units, so that each one of them has as many input queues as there are classes of tasks. Each processing unit selects the task to be executed next from one of its input queues according to a predefined rule. Arrivals of new tasks as well as selections of the task to be executed and completions of old tasks, modify the processing units input queues and hence the system state.

Such a system may be represented by several submodels, one for each processing unit, comprising one transition for each class of tasks. As an example, in the simple case of only two different classes of tasks, each processing unit can be described by a subnet as in Fig. 10, where a preselection policy is applied between transitions t_1 and t_2. It is conceivable to assume that a modification of the state of a subnet has no influence on the behavior of a different subnet. The mutual behavior of the transitions preselected in each subnet is that specified by the race policy.

Formally, the local preselection policy can be described using the following notation. Let A_I, $(I = 1, 2, \cdots, R)$ be the Ith local preselection set of transitions such that

$$\bigcup_{I=1}^{R} A_I = T. \tag{24}$$

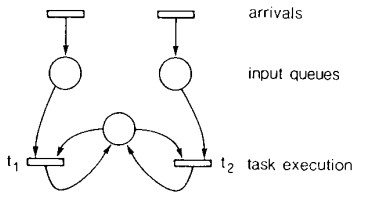


Fig. 10. Representation of a processing unit used by two classes of tasks.

When the *SPN* enters marking $\boldsymbol{M}$ we can identify certain subsets $a_I(\boldsymbol{M})$, $(I = 1, 2, \cdots, R)$ such that

$$a_I(\boldsymbol{M}) \subseteq A_I,\ I = 1, 2, \cdots, R \quad \text{and}$$

$$\bigcup_{I=1}^{R} a_I(\boldsymbol{M}) = E'(\boldsymbol{M}). \tag{25}$$

We can assume that for each set a local preselection probability distribution has been defined, so that $q_{jJ}(\boldsymbol{M}, \boldsymbol{Z})$ represents the probability of selecting transition j within set $a_J(\boldsymbol{M})$ (the jth component of the probability distribution associated with the Jth set of transitions). Then, assuming that transition t_i belongs to set I, (4) can be rewritten in the following way:

$$D_i(x|\boldsymbol{M}, \boldsymbol{Z}) = \int_0^x \left[\prod_{J \neq I} \sum_{j:t_j \in a_J(\boldsymbol{M})} q_{jJ}(\boldsymbol{M}, \boldsymbol{Z})\left[1 - \Phi_j(u|\boldsymbol{M}, \boldsymbol{Z})\right] \right] \cdot q_{iI}(\boldsymbol{M}, \boldsymbol{Z})\, d_u\Phi_i(u|\boldsymbol{M}, \boldsymbol{Z}). \tag{26}$$

For the solution of the model under this policy it is necessary to know the distributions $\Phi_k(x|\boldsymbol{M}, \boldsymbol{Z})$ for each transition of the *SPN*, and the probability mass functions $q_{jJ}(\boldsymbol{M}, \boldsymbol{Z})$ for each set of transitions.

The dependence of the distributions $\Phi_k(x|\boldsymbol{M}, \boldsymbol{Z})$ on the past history of the *SPN* is similar to that described in the case of the race policy. The difference is that only the history of the local preselection sets to which a given transition belongs is relevant to the behavior of the transition itself.

The dependence of the probability mass functions $q_{jJ}(\boldsymbol{M}, \boldsymbol{Z})$ on the past history of the *SPN* may be related to the states of the local preselection sets through boolean *choice variables* associated with all transitions of the net. The modification of these variables can be accomplished following different rules. As an example we discuss two possible policies named *variable* and *persistent*.

Variable: The choice variable associated with transition t_k accounts for the selection performed when the set to which this transition belongs was enabled for the last time. Usually all the transitions belonging to a given set have their choice variable set to "false." When the enabling conditions of the transitions belonging to the local preselection sets change, the choice variables are reset and, if some of the transitions are still enabled, one of these transitions is selected as a candidate to fire. Its choice variable is turned to "true," and remains such until either it succeeds in firing or the enabling conditions change again.

Persistent: The choice variables are reset, and the selection is repeated only when the selected transition is disabled. If the changes in the local state do not disable the transition that is currently selected, the selection is retained.

Referring to the multiprocessor system example, the variable policy corresponds to performing a new choice every time a new task joins the input queue of a processing unit or a task is completed. The activity performed while executing a task that is interrupted may be lost or not, depending on the work memory policy associated with the corresponding transition. The persistent policy, instead, implies that a selected task is executed until completion independently of possible modifications of the processing units input queues. Note that the age variable and the choice variable are independent of each other, since they refer to two distinct aspects of the execution policy. Several combinations are thus possible and meaningful.

1) Special Cases: The local preselection policy is obtained by using the concepts of preselection and race in conjunction. By properly defining the local preselection sets it is possible to let the local preselection policy degenerate into either global preselection or pure race.

If all the transitions in the *SPN* are grouped into a single local preselection set, the preselection obviously becomes global. Note that the choice variables in this case become useless since nothing can happen in the net before a preselected transition actually fires. The behavior is thus implicitly that specified by the persistent policy. For what concerns the work age variable, it was already noticed in Section VI-A that global preselection implies the resampling policy.

If instead there is a one to one correspondence between transitions and local preselection sets, the *SPN* executes as specified by the race policy. Indeed, all enabled transitions are selected since any one of them is the only member of the corresponding local preselection set, and then race with each other. Also in this case the choice variables become useless, and the behavior is implicitly the one specified by the persistent policy. For what concerns the modification of the work age variables when each local preselection set comprises only one transition, the age memory and enabling memory policies correspond exactly to those specified in the case of pure race. The resampling policy is instead not equivalent to the corresponding policy specified in the case of pure race. Indeed, when the *R–R* policy is defined, any change in the *SPN* marking induces a resampling of the distributions of the delays associated with all the enabled transitions. Instead, when the resampling policy is defined over an *SPN* with local preselection, a change in the marking induces a resampling only for those transitions whose enabling conditions have changed (i.e., the markings of their input places have either increased or decreased), but not for those transitions whose input places were left untouched by the marking change. This new type of behavior may actually turn out to be more useful from a modeling point of view, since it allows interactions only among activities that modify each other's state.

2) Definition of the Local Preselection Sets: General rules for the definition of local preselection sets appear difficult to establish. The parallelism of the system can however serve as a guideline for the specification of such sets. Indeed, transitions representing activities that are performed in parallel should never belong to the same local preselection set, whereas transitions corresponding to activities that are either in conflict or that must be executed in sequence can belong to the same local preselection set.

Moreover, if the *SPN* model is constructed using a hierarchical approach, local preselection sets should naturally be confined to subnets representing refinements of objects (places and/or transitions) described in nets of higher levels. Indeed, the local preselection policy appears to be a very appropriate execution rule for models comprising several subnets. Using this policy it is possible to guarantee that the changes of state that do not involve a given subnet, should not influence its behavior.

C. Description of the Preselection at the Net Level

The idea of representing the preselection at the net level by separating the selection process from the activity execution has been proposed in the definition of *ESPN* [16]. In these models the construct of the *probabilistic arc* allows the implementation of local preselection among conflicting transitions. The probability distribution, that is defined on the set of the output arcs of a transition, can be obtained from the *SPN* structure.

A different approach for the specification of global and local preselection policies at the *SPN* level is the representation of the probability mass functions according to which the preselection is made (the probabilities $\{p_k(\boldsymbol{M})\}$ in the case of global preselection and the probabilities $\{q_{jJ}(\boldsymbol{M})\}$ in the case of local preselection) as a branching process that occurs in zero time, separated from the activity execution part that consumes time.

The behavior of an *SPN* of this type is thus characterized by intermediate states where the preselection takes place and that are traversed in zero time. The explicit representation of this branching process at the net level was already proposed in the literature in the definition of generalized *SPN* (*GSPN*) [14], [15] models.

Even if these models can be used to describe systems with preselection, an explicit definition of the local preselection sets might be preferable because it provides a powerful construct that is completely defined at the net level.

VII. Concluding Remarks

A formal definition of stochastic Petri nets with generally distributed firing times has been presented, and different semantics of the model have been discussed in conjunction with different execution policies. The models presented in this paper are a generalization of the various

SPN models that were previously proposed in the literature.

Many interesting extensions to the work presented in this paper may further increase the *SPN* modeling capabilities. For example, one might consider the coloring of tokens [29], [30] and the association of timing with other network elements besides transitions. Once tokens are labeled, it is also possible to define orderings of tokens inside places [31], and study their impact on the *SPN* behavior.

The use of *SPN* for performance analysis relies heavily on the availability of tools automating the solution techniques described in this paper. In particular, these tools should give the user the possibility of describing the net graphically, and of defining both the model parameters and the interesting performance indices during the net specification. All details concerning the construction of the reachability graph, the solution of the Markov chain, and the computation of the performance results should be transparent to the user who only needs to construct the model, validate it, and then request its solution.

Some concluding comments about the use of the models introduced in this paper, and about the implementation of the different execution policies in practical applications are in order.

A. Execution Policies and Model Construction

The definitions of execution policies introduced in this paper are very general and the construction of an *SPN* model in this general context may require the specification of a large quantity of additional information besides the *SPN* topology.

In particular, in the construction of a race *SPN* model the distributions $\Phi_k(x \mid \boldsymbol{M}, \boldsymbol{Z})$ should be specified for each transition and each possible history, while in the construction of a preselection *SPN* model both the probabilities $p_k(\boldsymbol{M}, \boldsymbol{Z})$ and the distributions $F_k(x \mid \boldsymbol{M}, \boldsymbol{Z})$ should be specified for each transition and each possible history. Even if we have already restricted the possible ways in which the process may be conditioned on its past history $\boldsymbol{Z}$, the conditioning on $\boldsymbol{M}$ might still be interpreted very extensively. In the case of a race *SPN*, for example, in principle it is possible to define a totally new distribution $\Phi_k(x \mid \boldsymbol{M}, \boldsymbol{Z})$ for each marking $\boldsymbol{M}$. Similarly, in the case of a preselection policy, different choices may be performed within identical sets of transitions that are enabled by different markings so that both new probabilities $p_k(\boldsymbol{M}, \boldsymbol{Z})$, and distributions $F_k(x \mid \boldsymbol{M}, \boldsymbol{Z})$, may be defined for each of these markings. In practice, exploiting the generality of the definitions to this extent appears not to be desirable. Indeed, the main advantage of using *SPN* instead of more traditional Markovian models is the possibility of describing the behavior of a real system by choosing a proper topology and by introducing additional specifications without becoming directly involved with the details of the underlying stochastic process.

This implies that the modeler should be able to specify all the stochastic parameters of the model *before* (*and possibly without*) knowing the reachability set. Moreover the stochastic behavior of the model should be consistent with the qualitative behavior of the underlying *PN*, as it can be obtained by structural analysis, e.g., local preselection sets should be restricted to conflicting transitions.

This observation leads to a somewhat restricted use of the execution policies defined before. In the case of the race policy, the dependence on the marking must be such that only a relatively small set of places may be recognized as relevant for the definition of each component of the firing distribution. The fact that the number of tokens in each of these places is not known *a priori* should not hamper the definition of the firing distribution. Indeed, these components should all be well defined for each possible submarking pertaining to these places. The *PN* boundedness properties as well as a preliminary study of other *PN* structural properties can be of great help in this process. In the case of the preselection policy, the definition of a valid set of probabilities $p_k(\boldsymbol{M}, \boldsymbol{Z})$ requires the knowledge of the transitions in $E'(\boldsymbol{M})$. Whenever the *SPN* model is not trivially small, identifying the set of transitions that are simultaneously enabled in each marking is a difficult task. For this reason, a local preselection policy is normally used. In this case, the user identifies a set of transitions that may be in conflict, and defines for this set some marking dependent firing probabilities $q_{il}(\boldsymbol{M})$. The dependence on the marking should be restricted to the number of tokens in the places connected to the transitions in the local preselection set.

For what concerns the definition of the distributions $F_k(x \mid \boldsymbol{M}, \boldsymbol{Z})$ in the case of the preselection policy, we may observe that comments similar to those made with respect to the dependence of the firing distribution on the marking in the case of the race policy apply also in this case.

Drawing on these considerations, a convenient use of *SPN* models should require the user to specify only the distributions associated with each activity of the system to be modeled, and for each transition some capacity function that allows the representation of the parallel execution of identical activities with either "multiple server" or "generalized processor sharing" transitions.

B. The Use of General Distributions

The use of general distributions requires a careful analysis of the system behavior in order to correctly build the graphical model. The use of exponential distributions, instead, due to their memoryless property, may forgive the designer some imperfection in the model construction. Indeed, the memoryless property often renders slightly different models equivalent, making the distinction among the various firing policies unessential. However, even if an approximate solution of the model is obtained first by using only exponential distributions, it is not easy to infer from the comparison between the results with exponential and general distributions any information on the correctness of the graphical model. In general it can be even difficult to predict whether the performance index to be

computed should either increase or decrease when general distributions are introduced in the model: both behaviors are possible depending on the nature of the distributions and of the chosen execution policies.

The use of *PH* distributions requires the expansion of the *SPN* reachability set in the construction of the state space of the stochastic process. The number of states generated in this way may be very large; this can be a cause of nonnegligible difficulties, since it worsens the problem of the exponential growth of the state space of these models.

C. The Use of Execution Policies

The race policy provides the means for defining the model by simply specifying the distribution function of the activity associated with each *PN* transition considered in isolation. The Markovian model can then be automatically constructed from this specification.

The implementation of the preselection policy requires more caution. The preselection probability functions are meaningfully defined on the reachability set and no simple but general tool has yet been devised for introducing such functions at the net level. Further research in this direction is needed and we hope that classical net theory can provide useful results for this purpose.

REFERENCES

[1] C. A. Petri, "Communication with automata," Rome Air Dev. Center, New York, NY, Tech. Rep. RADC-TR-65-377, 1966.

[2] T. Agerwala, "Putting Petri nets to work," *Computer*, pp. 85-94, Dec. 1979.

[3] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[4] W. Reisig, *Petri Nets: An Introduction*. New York: Springer-Verlag, 1985.

[5] *Proc. Int. Workshop Timed Petri Nets*, Torino, Italy. Washington, DC: IEEE Computer Society Press, July 1985.

[6] *Proc. Int. Workshop Petri Nets and Performance Models*, Madison, WI. Washington, DC: IEEE Computer Society Press, Aug. 1987.

[7] W. M. Zuberek, "Timed Petri nets and preliminary performance evaluation," in *Proc. 7th Annu. Symp. Computer Architecture*, La Baule, France, May 1980, pp. 88-96.

[8] R. R. Razouk and C. V. Phelps, "Performance analysis using timed Petri Nets," in *Proc. Int. Conf. Parallel Processing*, Aug. 1984, pp. 126-129.

[9] M. A. Holliday and M. K. Vernon, "A generalized timed Petri net model for performance analysis," in *Proc. Int. Workshop Timed Petri Nets*, Torino, Italy. Washington, DC: IEEE Computer Society Press, July 1985, pp. 181-190.

[10] F. J. W. Symons, "Introduction to numerical Petri nets, a general graphical model of concurrent processing systems," *A.T.R.*, vol. 14, no. 1, pp. 28-33, Jan. 1980.

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

[12] M. K. Molloy, "On the integration of delay and throughput measures in distributed processing models," Ph.D. dissertation, UCLA, Los Angeles, CA, 1981.

[13] A. Bertoni and M. Torelli, "Probabilistic Petri nets and semi-Markov processes," in *Proc. 2nd European Workshop on Petri Nets*, Bad Honnef, West Germany, Sept. 1981.

[14] M. Ajmone Marsan, G. Balbo, and G. Conte, "A class of generalized stochastic Petri nets for the performance analysis of multiprocessor systems," *ACM Trans. Computer, Syst.*, vol. 2, no. 1, May 1984.

[15] M. Ajmone Marsan, G. Balbo, G. Chiola, and G. Conte, "Generalized stochastic Petri nets revisited: Random switches and priorities," in *Proc. Int. Workshop Petri Nets and Performance Models*, Madison, WI. Washington, DC: IEEE Computer Society Press, Aug. 1987, pp. 44-53.

[16] J. B. Dugan, K. S. Trivedi, R. M. Geist, and V. F. Nicola, "Extended stochastic Petri nets: Applications and analysis," in *Proc. Performance '84*, Paris, France, Dec. 1984.

[17] M. Ajmone Marsan and G. Chiola, "On Petri nets with deterministic and exponential transition firing times," in *Proc. 7th European Workshop Application and Theory of Petri Nets*, Oxford, England, June 1986.

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

[19] P. Heidelberger and S. S. Lavenberg, "Computer performance evaluation methodology," *IEEE Trans. Comput.*, vol. C-33, no. 12, pp. 1195-1220, Dec. 1984.

[20] M. Ajmone Marsan, A. Bobbio, G. Conte, and A. Cumani, "Performance analysis of degradable multiprocessor systems using generalized stochastic Petri nets," *IEEE Comput. Soc. Distributed Processing T-C Newslett.*, vol. SI-1, no. 6, 1984.

[21] G. Florin and S. Natkin, "Les reseaux de Petri stochastiques," *TSI*, vol. 4, no. 1, pp. 143-160, Feb. 1985.

[22] R. A. Howard, *Dynamic Probabilistic Systems*. New York: Wiley, 1971.

[23] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani, "On Petri nets with stochastic timing," IENGF, Torino, Italy, Internal Rep. 314, Apr. 1985.

[24] D. R. Cox, "The analysis of non-Markovian stochastic processes by the inclusion of supplementary variables," *Proc. Cambridge Phil. Soc.* vol. 51, pp. 433-440, 1955.

[25] R. Pyke, "Markov renewal processes with finitely many states," *Ann. Math. Stat.*, vol. 32, pp. 1243-1259, 1961.

[26] A. Cumani, "On the canonical representation of homogeneous Markov processes modelling failure time distributions," *Microelectron. Rel.*, vol. 22, pp. 583-602, 1982.

[27] A. Cumani, "ESP—A package for the evaluation of stochastic Petri nets with phase-type distributed transition times," in *Proc. Int. Workshop Timed Petri Nets*, Torino, Italy. Washington, DC: IEEE Computer Society Press, July 1985, pp. 144-151.

[28] A. Bobbio and A. Cumani, "Discrete state stochastic systems with phase type distributed transition times," in *Proc. ASME Int. Conf. Modelling and Simulation*, Athens, Greece, 1984, pp. 173-192.

[29] A. Zenie, "Colored stochastic Petri nets," in *Proc. Int. Workshop Timed Petri Nets*, Torino, Italy. Washington, DC: IEEE Computer Society Press, July 1985, pp. 262-271.

[30] C. Marinescu and Chuang Lin, "On stochastic high level Petri nets," in *Proc. Int. Workshop Petri Nets and Performance Models*, Madison, WI. Washington, DC: IEEE Computer Society Press, Aug. 1987, pp. 34-43.

[31] G. Memmi and A. Finkel, "An introduction to FIFO nets," *Theoret. Comput. Sci.*, vol. 35, pp. 191-214, 1985.

Marco Ajmone Marsan (S'76-M'76-SM'86) was born in Torino, Italy, in 1951. He received the Dr. Ing. degree in electronic engineering from Politecnico di Torino, Italy, and the Master of Science degree from the University of California, Los Angeles.

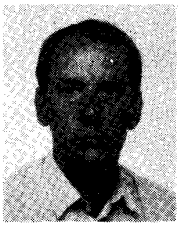
He is currently a Full Professor with the Department of Computer Science of the University of Milan, Italy. From November 1975 to October 1987 he was at the Department of Electronics of Politecnico di Torino, first as a Researcher, then as an Associate Professor. During the Summers of 1980 and 1981 he was with the Research in Distributed Processing Group, Department of Computer Science, UCLA. His current interests are in the fields of performance evaluation of data communication and computer systems, communication networks, and queueing theory.

Dr. Ajmone Marsan is a member of AEI, the Italian Electrotechnics Association.

Gianfranco Balbo was born in Piossasco, Italy, in 1946. He received the Doctor degree in physics from the University of Torino, Italy, in 1970, and the M.S. and Ph.D. degrees from Purdue University, West Lafayette, IN, in 1975 and 1979, respectively.

Since 1978 he has been with the Department of Computer Science of the University of Torino where he is now a Full Professor in charge of two introductory courses on operating system theory and performance evaluation of computer systems. His current research interests are in the area of performance evaluation of computer systems, queueing network models, stochastic Petri net models, and queueing theory. He is coauthor with S. C. Bruell of the book *Computational Algorithms for Closed Queueing Networks*, and with M. Ajmone Marsan and G. Conte of the book *Performance Models of Multiprocessor Systems*.

Dr. Balbo is a member of the Association for Computing Machinery.



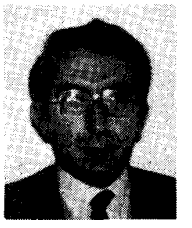
Andrea Bobbio was born in Torino, Italy, on February 24, 1946. He received the degree in nuclear engineering from Politecnico di Torino in 1969.

In 1971 he joined the Istituto Elettrotecnico Nazionale Galileo Ferraris, where he was involved in research on degradation mechanisms in electronic devices with particular emphasis on electromigration failures in aluminum metallizations. In 1974 he was assigned to the staff for setting up an Italian Reliability Data Bank on electronic components in cooperation with the Italian electronic industries. In particular, he was involved in the development of procedures for the collection and the statistical analysis of failure data. In 1978, he turned his attention to the area of system reliability, modeling, quantitative analysis, optimization, and Petri nets. During 1984, 1986, and 1987 he was a Visiting Scientist in the Department of Computer Science at the Duke University, Durham, NC. He is the author of several reliability-oriented papers.

Giovanni Chiola was born in Torino, Italy, in March 1958, and received the Doctor in Electronics Engineering degree from the Politecnico di Torino in 1983.

He worked at the Department of Electronics of the Politecnico di Torino as a consultant until the end of 1985, developing a computer package for performance modeling and analysis based on stochastic Petri nets, and studying performance models for various multiprocessor architectures. In March 1986, he joined the Faculty of the Department of Computer Science of the University of Torino as a Researcher. He (co)authored about 20 papers on performance modeling of computer architectures and stochastic Petri nets. His research interests include distributed systems, graphical interfaces, and system software, as well as performance modeling.

Dr. Chiola is a member of the IEEE Computer Society, the Association for Computing Machinery, and SIGMETRICS.



Gianni Conte (M'82) was born in Torino, Italy, in 1946. He received the Doctor degree in electronic engineering from the Politecnico di Torino in 1970.

From 1972 to 1986 he has been with the Department of Electronics of the Politecnico di Torino. From January 1987 to October 1988 he was at the University of Salerno as a Full Professor of Computer Science. In November 1988 he joined the Faculty of Engineering at the University of Parma. His current research activity includes performance evaluation of distributed systems, timed Petri nets, and special purpose parallel architectures. He has (co)authored more than 50 papers on digital electronics, multiprocessor architectures, and performance evaluation of computer systems.

Dr. Conte is a member of the Associazione Elettrotecnica Italiana, the Associazione Italiana per il Calcolo Automatico, and the IEEE Computer Society.

Aldo Cumani (PM'80) was born in Torino, Italy, in 1948. He received the degree in electronic engineering from Politecnico di Torino in 1972.

In 1972 he joined the Istituto Elettrotecnico Nazionale Galileo Ferraris, where he was involved in research on signal processing and pattern recognition. From 1978 to 1984, he turned his attention to the area of stochastic modeling with application to system reliability. His current interests are mainly in the field of computer vision and its applications to monitoring systems and robotics.