# STOCHASTIC PETRI NETS: AN ELEMENTARY INTRODUCTION

M. Ajmone Marsan
Dipartimento di Scienze dell' Informazione
Università di Milano, Italy

**ABSTRACT** - *Petri nets in which random firing delays are associated with transitions whose firing is an atomic operation are known under the name "stochastic Petri nets". These models are discussed, with the purpose of explaining why they were proposed in the performance evaluation field, why random delays with negative exponential probability density functions are mainly used, and what are their strong and weak points. An effort is made to summarize the lines of research that are currently being pursued, and to explain what new results would be regarded as breakthroughs and have the most impact on the use of this modeling technique in the application field.*

**KEY WORDS** - Stochastic Petri nets, Performance evaluation, Markov chains, Queues.

**CONTENTS**

## 1 INTRODUCTION

Petri nets (PN) [1,2,3], in their various shapes and sizes, have been used for the study of the *qualitative* properties of systems exhibiting concurrency and synchronization characteristics.

The use of PN-based techniques for the *quantitative* analysis of systems requires the introduction of temporal specifications in the basic, untimed models.

This fact has been recognized since a fairly long time, and several different proposals for the introduction of temporal specifications in PN have appeared in the literature. The main alternatives that characterize the different proposals concern

- the PN elements (either places or transitions) with which timing is associated,
- the semantics of the firing in the case of timed transitions (either atomic firing or firing in three phases),
- the nature of the temporal specification (either deterministic or probabilistic).

In this paper we consider PN models that are augmented with a temporal specification by associating a firing delay with transitions. The transition firing is atomic, i.e., tokens are removed from input places and put into output places with a single, indivisible operation. The specification of the firing delay is of probabilistic nature, so that either the probability density function (pdf) or the probability distribution function (PDF) of the delay associated with a transition needs to be specified. In the simplest case we assume that all delays have negative exponential pdf, but we also briefly consider the case of general pdf.

The class of models that we consider are normally referred to with the name Stochastic PN (SPN).

The goal of the paper is to discuss several points related to SPN, including

- why were SPN introduced,
- why exponential pdf are mainly used for the specification of timing,
- what are the strong and weak points of SPN,
- what research efforts are currently in progress,
- what would be the most important breakthrough results in the SPN field.

This paper is addressed to PN experts who are not familiar with the stochastic performance modeling field. For this reason, a brief overview of the classical approach to the performance evaluation of systems in a probabilistic framework is included in Section 2, where some elementary notions about stochastic processes, and queueing theory are summarized.

Section 3 contains the discussion on the various types of SPN models that were presented in the literature, together with some illustrative examples. Comments on the present and future research efforts are also included in this section.

Finally, Section 4 provides the concluding remarks.

In spite of the author's efforts to provide an objective overview of the SPN field, the discussion in the paper is probably biased, due to his particular background and experience. It is thus possible that some classes of SPN that were proposed by the author with his colleagues are presented with more emphasis than they deserve. On the other hand, in a paper like this one, it is necessary to express personal opinions, and to make a selection of the models to be presented. As regards the selection, it naturally favors the models with which the author is more familiar. As regards personal opinions, normally researchers tend to highly value their own work (says an italian proverb: "every cockroach is beautiful to his mother"). Readers interested in the use of SPN models are thus adviced to carefully compare the original proposals in order to develop their own opinions about the suitability of the various modeling approaches to their particular application field.

# 2 PERFORMANCE EVALUATION

The performance evaluation area can be initially subdivided into two subareas. The first one relates to *measuring*, and comprises three distinct fields that can be called

- measurements,
- benchmarks,
- prototypes.

Measurements are performed on a real system under real operating conditions. They provide the *actual* system performance in the particular condition in which the system is observed. However, measurement results have very little generality, since they are heavily dependent upon the detailed characteristics of the measured system, and on the particular workload imposed on the system during the measurement.

When the performances of two systems, say two supercomputers, have to be compared, it is not sufficient to rely upon measurements, since nothing guarantees that the operating conditions under which measurements are performed are equivalent. The comparison would thus be unfair.

In order to overcome this problem, benchmarks were developed. They provide an artificial workload for the system, such that observations can be performed in equivalent operating conditions, and meaningful comparisons can be made.

Both measurements and benchmarks require the availability of the system to be studied, so that it can be observed. In the (many) cases in which the performance study concerns a system that is not available (maybe because it is not yet operational), it is necessary to develop a representative approximation of it, either in hardware or in software. Such approximations, which need to be fairly detailed, are normally called prototypes (the term emulator is also often used when the approximation is implemented in software). Observations are then made on such prototypes, possibly using benchmarks as artificial workloads.

In all three cases, the system performance is obtained by observing the behaviour of the system, or its approximations, in operation, i.e., when loaded by either the actual user requests, or the benchmark.

The study of the performance of a system, however, is not only an important task during and after the system implementation, but also during the early design stages, in order to compare possible alternate architectural choices. This is true in particular when the development of new systems is mainly motivated with the request for ever-increasing performance, like in the computer field.

During the design process, measurements on real systems are obviously not possible, and also prototype implementations present insormountable difficulties due to the necessity of specifying many details that are far from being decided.

The second subarea of performance evaluation thus comes into play: *modeling*. It can be partitioned into two fields:

- simulation models,
- analytical models.

In both cases the performance study is carried out using a description that includes only some "important" characteristics of the system. In the case of simulation models, the description is given by means of a computer program, whereas in the case of analytical models the description is given in mathematical terms.

Models (both simulative and analytical) can be either deterministic or probabilistic. While it is clear that most systems of interest exhibit a deterministic behaviour (we tend to like the fact that by running twice the same program with the same input data we obtain the same results), it may be simpler to describe a very large number of complex, detailed deterministic phenomena by means of macroscopic probabilistic assumptions. This is often done because details are not known, and even when they are, their inclusion may lead to very complex models. Furthermore, the probabilistic approach may be advantageous because it may provide sufficient accuracy, while yielding more general results, and it may permit the study of sensitivity to parameter variations.

It should be noted that a key element in the development of a model is the selection of the *level of abstraction* (also called level of detail). This amounts to selecting the system features to be included in the model. No precise rule exists for this selection, that rests mainly on the experience and ingenuity of the performance analyst. On the other hand, the level of abstraction is the element that differentiates a model from a prototype or an emulator. Simulation lends itself better to the development of more detailed models, whereas analytical models are normally more abstract.

An important characteristic of models concerns the representation of the system behaviour along the time scale. While it is obvious that any instrument for the measurement of time operates according to a discrete time scale, due to its finite precision, and that most interesting modern systems, being digital in nature, intrinsically use a discrete time scale, models often use a *continuous* time scale. The reason for this discrepancy lies in the greater simplicity of continuous time models. Indeed, if the time axis is discrete, the model has to consider the fact that multiple events may occur between two consecutive time marks, and explore the effect of all possible combinations among these events. In the continuous time scale, instead, using appropriate probabilistic assumptions, it is possible to univocally order events, so that it is always possible to take into consideration only one event at a time.

In this paper we deal with models of a probabilistic nature operating with a continuous time scale.

The mathematical framework underlying this class of models, be they simulative or analytical, is the theory of stochastic processes.

## 2.1 Stochastic Processes

Random phenomena are close to our everyday experience, at least due to our familiarity with unpredictable weather changes, equipment failures, and games of chance based on cards or dices.

A stochastic process is a mathematical model useful for the description of phenomena of a probabilistic nature as a function of a parameter that usually has the meaning of time.

Since the definition of a stochastic process is based on the notion of a *random variable*, it is necessary to recall some elementary concepts of probability theory first.

A *random experiment* is an experiment which may have several different outcomes. The set of all possible elementary outcomes is the *sample space* of the experiment. A simple example of a random experiment is provided by the toss of a fair dice. The sample space is in this case comprised of six elementary outcomes. By associating a probability measure to all possible (elementary and complex) outcomes of a random experiment we construct a *probability space*. Continuing with our example, we can associate probability 1/6 with each elementary result of the dice toss, and appropriate probabilities to complex results such as "more than one and less or equal to five, but not equal to three".

A random variable is a real function defined over a probability space; for example, a random variable could associate the value $i\pi$ (where $\pi = 3.1415\ldots$) to the elementary result $i, i = 1, 2, \ldots, 6$. The set of possible values of the function is the *state space* of the random variable.

The probabilistic description of a random variable $X$ is given in terms of its PDF

$$F_X(x) = P\{X \leq x\}$$

which is a real, nonnegative, nondecreasing function of $x$ for which

$$\lim_{x \to -\infty} F_X(x) = 0$$

and

$$\lim_{x \to \infty} F_X(x) = 1 \quad .$$

Alternatively, the random variable $X$ can be described by its pdf

$$f_X(x) = \frac{d}{dx} F_X(x)$$

which is a nonnegative function for which

$$\int_{-\infty}^{\infty} f_X(x) dx = 1 \quad .$$

In the case of random variables assuming values in a discrete set, instead of considering their pdf, which is a generalized function, we consider their probability mass function (pmf)

$$p_X = (p_1, p_2, p_3, \ldots)$$

which is a vector whose entries

$$p_i = P\{X = x_i\} \qquad i = 1, 2, \ldots$$

are the probabilities that the random variable equals one of the admissible values.

The importance of the probabilistic characterization of a random variable is in the fact that it provides the tool for the mathematical formulation of any problem involving the random variable itself.

The probabilistic description of a random vector $\mathbf{X}$ comprising $n$ random variables $X_i$, $i = 1, 2, \ldots, n$ is given by the joint PDF of the individual random variables:

$$F_{\mathbf{X}}(\boldsymbol{x}) = P\{X_1 \le x_1, X_2 \le x_2, \ldots, X_n \le x_n,\}$$

or by their joint pdf

$$f_{\mathbf{X}}(\boldsymbol{x}) = \frac{\partial^n}{\partial x_1 \partial x_2 \ldots \partial x_n} F_{\mathbf{X}}(\boldsymbol{x}) \quad .$$

We are now ready to give the definition of a stochastic process. A stochastic process $\{X(t), t \in T\}$ is a family of random variables defined over the same probability space, taking values in the state space $S$, and indexed by the parameter $t$, which assumes values in the set $T$; normally $T = [0, \infty)$.

A stochastic process can be visualized as a family of functions of time, called *sample paths* of the process. Each sample path defines a particular trajectory over the state space, and corresponds to a possible observed behaviour of the process. Consider for example the stochastic process modeling the state of a door (either closed or open). Each sample path is a function of time made of segments corresponding to the durations of the open and closed times. The observation of the process eliminates the uncertainty on its evolution, and thus yields (at least for the observed time period) a sample path (the same difference is found before and after the toss of a dice). The set of all possible sample paths, together with a probability measure, may provide an alternate description of a stochastic process, which is however normally impractical.

The complete probabilistic characterization of a random process requires the description of any random vector comprising an arbitrary number of random variables extracted from the process at any set of time instants.

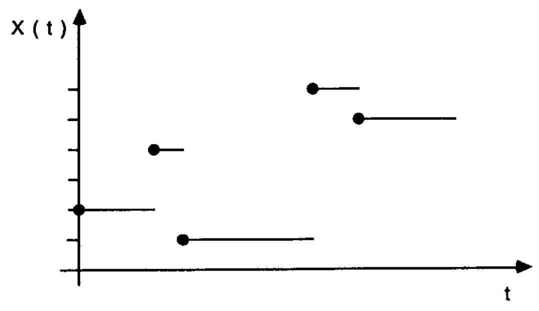


Figure 1: Sample path of a continuous-time Markov chain.

In the general case, the complete probabilistic characterization of a stochastic process is a formidable task. Special classes of stochastic processes for which the probabilistic characterization is simpler are of particular interest.

One such class is comprised of *Markov processes.* A Markov process is a stochastic process that satisfies the *Markovian property*

$$P\{X(t) \leq x | X(t_n) = x_n, X(t_{n-1}) = x_{n-1}, \ldots, X(t_1) = x_1, X(t_0) = x_0, \} =$$

$$= P\{X(t) \leq x | X(t_n) = x_n\}$$

for any $t > t_n > t_{n-1} > \cdots > t_1 > t_0$.

Note that the Markovian property defines a stochastic process for which the behaviour in the future (at some time $t$) depends only on the present situation (at time $t_n$), and not on the history (at times $t_{n-1}, \ldots, t_0$). In other words, a Markov process has no memory of the trajectory followed to reach the present state. This condition is not met by many real life systems, nevertheless Markovian processes are widely used for the construction of stochastic models of discrete event systems. Their main merit lies in their low analysis complexity.

Markov processes with a discrete state space are called *Markov chains.* If the parameter $t$ is discrete, the process is a discrete-time Markov chain. If the parameter $t$ is continuous, the process is a continuous-time Markov chain (CTMC).

## 2.2 Continuous-time Markov chains

The sample path of a CTMC has the appearance depicted in Figure 1. Each horizontal segment represents the sojourn time in a state (black dots denote right-continuity). The Markovian property requires that sojourn times in states be exponentially distributed random variables. Indeed, the negative exponential pdf

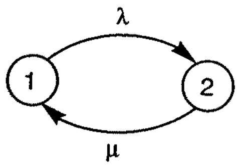
$$f_X(x) = \mu e^{-\mu x} u(x)$$

where $u(x)$ is the unit step function, and $\mu$ is the parameter (or rate) of the pdf, is the only continuous pdf for which the *memoryless property*

$$P\{X \geq x + \alpha | X \geq \alpha\} = P\{X \geq x\}$$

holds. Hence, at any time instant, the residual sojourn time in a state does not depend on the time already spent in the state (i.e., on the history), but only on the present state, as required by the Markovian property.

Note that the average of an exponentially distributed random variable with parameter $\mu$ is $\mu^{-1}$,



$$Q = \begin{bmatrix} -\lambda & \lambda \\ \mu & -\mu \end{bmatrix}$$

Figure 2: State transition rate diagram and infinitesimal generator for a CTMC with two states.

These considerations imply that for the complete probabilistic description of a CTMC it is sufficient to give the pmf over the state space $S$ at the initial time (typically 0), as well as the parameters of the negative exponential pdf describing the sojourn times in all states in $S$, and the probabilities of moving from one state to another.

In practice, a CTMC is described through either a *state transition rate diagram* or a transition rate matrix, also called *infinitesimal generator* and denoted by $\boldsymbol{Q}$. The state transition rate diagram is a labeled directed graph whose vertices are labeled with the CTMC states, and whose arcs are labeled with the rate of the exponential distribution associated with the transition from a state to another. The infinitesimal generator is a matrix whose elements outside the main diagonal are the rates of the exponential distributions associated with the transitions from state to state, while the elements on the main diagonal make the sum of the elements of each row equal to zero.

The two descriptions are shown in Figure 2 for a CTMC with two states (closed and open door in our previous example), for which the average sojourn time in state 1 is $\lambda^{-1}$, and the average sojourn time in state 2 is $\mu^{-1}$.

The solution of a CTMC model consists of the computation of the pmf over the state space $S$ either at any arbitrary time instant $t$ or in equilibrium conditions. When an equilibrium or *steady-state* pmf exists, and is independent of the initial state, the CTMC is said to be *ergodic* [4,5,6].

Denoting by

$$\pi_i(t) = P\{X(t) = i\}$$

the probability that the CTMC is in state $i$ at time $t$, and by

$$\boldsymbol{\pi}(t) = (\pi_1(t), \pi_2(t), \pi_3(t), \ldots)$$

the pmf at time $t$, it can be shown that

$$\boldsymbol{\pi}(t) = \boldsymbol{\pi}(0)e^{\boldsymbol{Q}t} \quad .$$

Furthermore, letting

$$\pi_i = \lim_{t\to\infty} P\{X(t) = i\}$$

in the case of ergodic CTMC, the steady-state pmf

$$\boldsymbol{\pi} = (\pi_1, \pi_2, \pi_3, \ldots)$$

can be obtained as the solution of the system of linear equations

$$\boldsymbol{\pi}\boldsymbol{Q} = \boldsymbol{0}$$

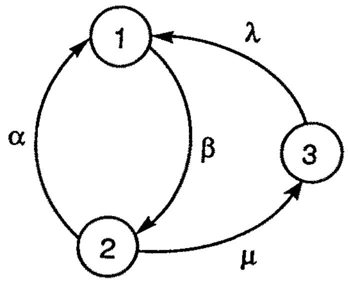


Figure 3: State transition rate diagram for the CTMC describing the behaviour of a lamp.

augmented with the normalization condition

$$\sum_i \pi_i = 1 \quad .$$

From these pmf it is possible to derive many parameters of interest.

As an example, consider a lamp equipped with one lightbulb. The lamp may be turned on and off, and the lightbulb can fail while the lamp is on. Failed bulbs are replaced with new ones, and before the replacement operation is performed, the lamp switch is set in the off position.

We can easily identify three states in our system: 1) off, 2) on, and 3) failed.

The transitions from state to state obey the following rules:

- when the lamp is off, it may be turned on,
- when the lamp is on, either it can be turned off, or the bulb can fail,
- when the lightbulb fails, it is replaced by a new one, after switching off the lamp.

In order to obtain a CTMC model we need to introduce temporal specifications such that the evolution in the future depends only on the present state, not on the history. To this purpose we assume that:

- the time periods during which the lamp is turned off are exponentially distributed with parameter $\beta$,
- the time periods during which the lamp is turned on are exponentially distributed with parameter $\alpha$,
- the lightbulb lifetime (sum of the durations of the on periods before a fault) is exponentially distributed with parameter $\mu$,
- the lamp repair time is exponentially distributed with parameter $\lambda$.

The state transition rate diagram of the resulting CTMC is depicted in Figure 3, and the infinitesimal generator is

$$\boldsymbol{Q} = \begin{bmatrix} -\beta & \beta & 0 \\ \alpha & -(\alpha+\mu) & \mu \\ \lambda & 0 & -\lambda \end{bmatrix} \quad .$$

The CTMC is ergodic, and the steady-state distribution is easily computed by solving the system of linear equations:

$$\begin{aligned} \beta\pi_1 &= \alpha\pi_2 + \lambda\pi_3 \\ (\alpha+\mu)\pi_2 &= \beta\pi_1 \\ \lambda\pi_3 &= \mu\pi_2 \\ \pi_1+\pi_2+\pi_3 &= 1 \end{aligned}$$

obtaining

$$\boldsymbol{\pi} = \frac{1}{\lambda(\alpha+\beta)+\mu(\lambda+\beta)}(\lambda(\alpha+\mu), \lambda\beta, \beta\mu) \quad .$$

Note that the first three equations of the system above can be interpreted as equalities of the flow into and out of a given state, where the probability flow over an arc is the product of the steady-state probability of the state from which the arc originates times the arc label. Thus for example in the case of state 1 we get

$$\text{flow out} = \beta\pi_1$$

$$\text{flow in} = \alpha\pi_2 + \lambda\pi_3$$

This also implies the linear dependency of the equations.

From $\boldsymbol{\pi}$ it is possible to compute several steady-state performance indices:

- $\pi_2$ is the fraction of time in which the lamp is on,
- $\pi_3$ is the fraction of time in which the bulb is failed,
- $[\lambda\pi_3]^{-1} = [\mu\pi_2]^{-1}$ is the average time between two consecutive failures,
- $[(\alpha+\mu)\pi_2]^{-1} = [\beta\pi_1]^{-1}$ is the average time between two consecutive instants at which the lamp is turned on.

Another simple example of a CTMC is provided by the *Poisson process*. In this case the state space comprises all nonnegative integers, and transitions are possible only from state $i$ to state $i+1$ for all $i \geq 0$. Sojourn times in states are independent random variables with negative exponential pdf, and mean independent of the state. The parameter of such exponential pdf is the *rate* of the Poisson process. The Poisson process obviously never reaches an equilibrium condition and hence is not ergodic. The pmf at time $t$ of a Poisson process with rate $\lambda$ comprises probabilities

$$\pi_i(t) = \frac{(\lambda t)^i}{i!}e^{-\lambda t} \quad , \qquad i \geq 0 \quad , t \geq 0 \quad ,$$

assuming that

$$\pi_0(0) = 1 \quad .$$

These probabilities form a Poisson pmf.

Constructing models directly at the CTMC level is generally *difficult*, mainly due to the need of choosing an appropriate state definition. For this reason, *more abstract* probabilistic modeling tools were proposed. The main such tools are based on queuing theory.

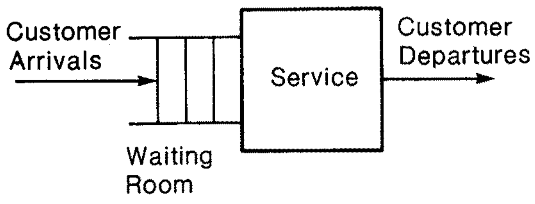


Figure 4: Pictorial representation of a queue.

## 2.3 Queues

A queue [7,8,9,10,11,12,13] is a system to which *customers* arrive to receive service by a *service station*. The service station may comprise one or more *servers*. When all servers are busy, customers are forced to wait in a *waiting room*. At the end of service, customers leave the queue. A pictorial representation of a queue is given in Figure 4.

A queue is a compact description of a probabilistic model in which users (customers) share resources (servers). The probabilistic characterization of the model is comprised of the stochastic process describing the arrival of customers, and the random variables describing the customer service times. Other parameters of a queuing model are:

- the number of servers in the service station,
- the size of the waiting room,
- the size of the customer population,
- the queueing discipline.

The simplest queue is known with the acronym M/M/1. The first symbol M denotes the arrival process as a Markovian one, and precisely as a Poisson process with a fixed rate, say $\lambda$. The second symbol M identifies the service time distribution as being Markovian (negative exponential pdf); an average service time $\mu^{-1}$ is considered. The symbol 1 refers to the presence of only one server in the service station. Furthermore, the size of the waiting room and the customer population are taken to be unlimited, and the first-come-first-served disclipline is used for the selection of the next customer to be served among those in the waiting room.

The CTMC corresponding to the M/M/1 queue has the state transition rate diagram depicted in Figure 5, which corresponds to the infinitesimal generator

$$\boldsymbol{Q} = \begin{bmatrix} -\lambda & \lambda & 0 & 0 & 0 & 0 & \cdots \\ \mu & -(\lambda+\mu) & \lambda & 0 & 0 & 0 & \cdots \\ 0 & \mu & -(\lambda+\mu) & \lambda & 0 & 0 & \cdots \\ 0 & 0 & \mu & -(\lambda+\mu) & \lambda & 0 & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} .$$

The solution of the M/M/1 queue in steady-state can be obtained in closed form when the underlying CTMC is ergodic, that is in the case $\lambda < \mu$:

$$\pi_i = \left(1 - \frac{\lambda}{\mu}\right)\left(\frac{\lambda}{\mu}\right)^i \quad i \geq 0 \quad .$$

Many queuing models exist with more elaborate characteristics for what concerns

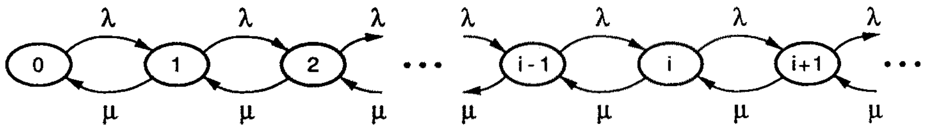


Figure 5: State transition rate diagram of the CTMC generated by the M/M/1 queue.

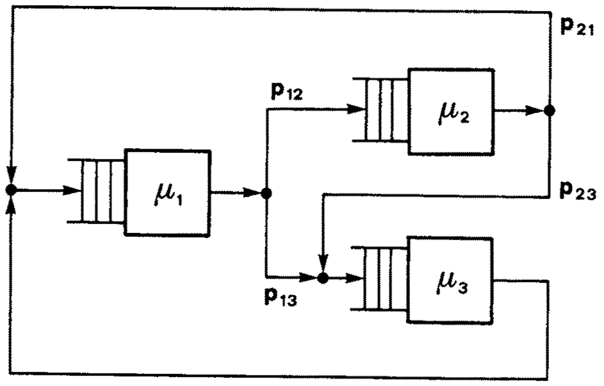


Figure 6: A closed queuing network comprising three queues.

- the customer arrival process,
- the pdf of the customer service time,
- the number of servers,
- the size of the waiting room,
- the size of the customer population,
- the queuing discipline.

In particular, for what concerns the first and the second items, many variations have been considered, trying to capture in the pdf of the interarrival and service times the increased complexity of models of realistic systems. This approach led to very complex service stations with several servers, and load-dependent service rates, whose analysis must be performed with more sophisticated tools than CTMC. The drawback of this tendency lies in the increased solution complexity, and in the hiding of the peculiar system characteristics into one pdf, thus complicating the model construction, and cancelling several aspects of the model which could provide insight into the system behaviour.

To correct this tendency, queuing network models were proposed.

## 2.4 Queuing networks

A queuing network [11,12,13] is a system of interconnected queues in which customers circulate, and possibly arrive from, and leave to, the outside world. When no arrival from and departures to the external world is possible, the queuing network is said to be closed; otherwise it is said to be open.

The path followed by customers in the network is determined by routing probabilities.

As an example, a closed queuing network comprising three queues is depicted in Figure 6.

With queuing networks it is possible to construct models of systems where the sharing of individual resources is represented in more detail than it would be possible if the system model had to be constructed using only one queue.

Queuing networks have become extremely popular in the applied stochastic modeling field for a wide gamut of different application areas, such as telecommunications, computers, manufacturing, and transportation.

The main reason for which queuing networks have become so popular is due to the *product form* solution property that holds for a fairly wide class of these models. This property implies that the steady-state solution of the queuing network can be *factored* in the product of the steady-state solutions of the individual queues, and hence obtained with very limited complexity.

It should be stressed that queuing theory is fairly advanced in the case of continuous-time models. The analysis of discrete-time models is much more complex because of the reasons mentioned in Section 2, and, for example, the product-form characteristic of queueing networks is not retained in the case of discrete-time models, except for some cases.

The shortcomings of queuing-based models are mainly due to their lack of descriptive power in presence of phenomena such as

- synchronization,
- blocking,
- splitting of customers,

and to the fact that these common features of distributed systems destroy the product-form characteristic, so that even a simple queuing model must be translated into its corresponding CTMC for the solution phase.

Stochastic Petri nets come into play in this environment, where they are equivalent to queuing models from the point of view of the solution, but offer a much greater descriptive power.

## 3 STOCHASTIC PETRI NETS

SPN models were proposed by researchers active in the applied stochastic modeling field, with the goal of developing a tool which allowed the integration of formal description, proof of correctness, and performance evaluation. For what concerns the last aspect, the proposals aimed at an equivalence between SPN and CTMC models.

In order to obtain an equivalence between a PN and a CTMC, it is necessary to introduce temporal specifications such that the future evolution of the model, given the present marking, is independent of the marking history. To this purpose, sojourn times in markings must be random variables with negative exponential pdf.

This idea formed the basis of the doctoral dissertations of S. Natkin [14] at the Conservatoire National des Arts et Métiers in Paris, France, and of M. K. Molloy [15] at the University of California at Los Angeles in the United States. These works were performed independently and approximately at the same time, in the late seventies. They led to the definition of almost identical models which even bore the same name: *Stochastic Petri Nets*. It should be mentioned, however, that the idea of associating an exponentially distributed random delay with the PN transitions was already present in the doctoral dissertation of F. J. Symons within the definition of Numerical Petri nets [16,17].

### 3.1 The Basic Model

An SPN is a six-tuple

$$SPN = (P, T, I, O, M_0, \Lambda)$$

where $(P,T,I,O,M_0)$ is the marked untimed PN underlying the SPN, which as usual comprises

- a set of places $P = (p_1, p_2, \ldots, p_m)$,
- a set of transitions $T = (t_1, t_2, \ldots, t_n)$,
- a set of input arcs $I \subset P \times T$,
- a set of output arcs $O \subset T \times P$,
- an initial marking $M_0 = (m_{01}, m_{02}, \ldots, m_{0m})$,

and $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ is an array of (possibly marking dependent) firing rates associated with transitions.

The firing of a transition is an atomic operation, i.e., tokens are removed from its input places and deposited into its output places with one indivisible operation (as opposed to timed PN models which will be mentioned later on, and in which the firing operation is divided into three separate phases).

A firing delay is associated with each transition. It specifies the amount of time that must elapse before the transition can fire. This firing delay is a random variable with negative exponential pdf. The parameter of the pdf associated with transition $t_i$ is the firing rate associated with $t_i$, $\lambda_i$. This firing rate may be marking-dependent, so that it should be written $\lambda_i(M_j)$. The average firing delay of transition $t_i$ in marking $M_j$ is $[\lambda_i(M_j)]^{-1}$.

Two interpretations of the operations of an SPN model are possible. The first one assumes that whenever a new marking is entered, each enabled transition samples an instance of the random firing delay from the associated pdf. The transition which samples the minimum firing delay is the one whose firing determines the change of marking; hence the sojourn time in the marking equals the minimum sampled delay value. The new marking is obtained with the rules of the underlying untimed PN, and the process is restarted.

Since the minimum of two random variables with negative exponential pdf and parameters $\mu_1$ and $\mu_2$ is a random variable which still is exponentially distributed, with parameter $(\mu_1 + \mu_2)$, the sojourn time in marking $M_j$ is a random variable with negative exponential pdf, with mean

$$\left[ \sum_{i:t_i \in E(M_j)} \lambda_i(M_j) \right]^{-1}$$

where $E(M_j)$ is the set of all enabled transitions in $M_j$.

The fact that all firing delays have exponential pdf permits to write a simple expression for the probability that a given transition, say $t_k$, samples the minimum delay instance, and hence determines the change of marking by firing:

$$P\{t_k | M_j\} = \frac{\lambda_k(M_j)}{\sum_{i:t_i \in E(M_j)} \lambda_i(M_j)} \quad , \qquad t_k \in E(M_j) \quad .$$

The memoryless property of the negative exponential pdf allows a different and more useful interpretation of SPN models. Whenever a change of marking enables a transition that was not previously enabled since its last firing, this transition samples an instance of the firing delay from the associated negative exponential pdf, and sets a timer at the value of the sampled delay instance. While the transition is enabled, the timer is decreased at a (possibly marking-dependent) constant speed. If the transition is disabled by the firing of a conflicting transition, the timer stops, and the decrement resumes (possibly with a different speed) when the transition is enabled again. When the timer reaches the value zero the transition fires.

The greater usefulness of the latter interpretation stems from the fact that we may now associate activities with transitions. These activities can be started and interrupted during the dynamics of

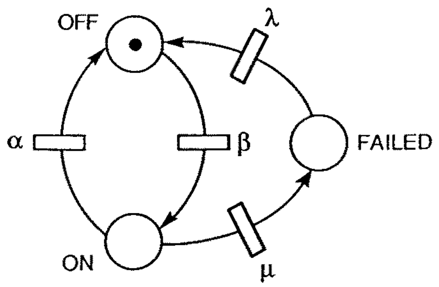


Figure 7: SPN model of the lamp example.

the model, and when they are completed, they induce a change of state. The same association was not possible with the former interpretation, since the resampling would have implied that activities were restarted in each marking and had a different duration at each restart.

Other interpretations can also be shown equivalent to the mentioned two, due to the memoryless property of the exponential pdf. When we shall mention SPN with generally distributed firing delays this equivalence will not hold any more.

It should be noted that the association of a random firing delay with continuous pdf with each transition results in a null probability for two transitions being scheduled to fire at exactly the same instant. An SPN model thus evolves by firing transitions one by one.

The reachability set of an SPN is identical to the one of the underlying untimed PN due to the unlimited support of the firing delay pdf. Indeed, the fact that the firing delay instances are sampled from the set of nonnegative real numbers guarantees that in any reachable marking all enabled transitions have a nonzero probability of firing.

The identity of the reachability sets implies that the structural properties obtained for the PN are still valid also for the SPN.

The state transition rate diagram of the CTMC corresponding to the SPN is obtained by constructing the reachability graph, and by labelling arcs with the firing rate of the transition whose firing produces the marking change.

Hence, obtaining the infinitesimal generator of the CTMC has the same complexity as generating the PN reachability set. The steady-state solution of the model is then obtained by solving the system of linear equations

$$\begin{array}{l} \boldsymbol{\pi Q} = \mathbf{0} \\ \sum_i \pi_i = 1 \end{array} \quad .$$

Note that $\pi_i$ denotes the steady-state probability of marking $M_i$ (and of state $i$ as well, since there is a one-to-one correspondence between markings and states), and $\boldsymbol{\pi}$ is the equilibrium pmf over the reachable markings.

We may now go back to the lamp example discussed in Section 2.2. The SPN model describing the system considered in the example is depicted in Figure 7. The reader is adviced to compare the SPN model and the CTMC state transition rate diagrams, noting the similarity in the topology. This is due to the fact that the PN underlying our SPN model is a marked graph [2].

In many cases this similarity in the topology does not exist.

Consider as a second example the SPN depicted in Figure 8. This is the SPN representation of the M/M/1 queue described in Section 2.3. Hence, the state transition rate diagram it generates is the one depicted in Figure 5, which comprises a denumerable infinity of states, in spite of the extremely simple SPN topology.

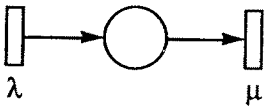


Figure 8: SPN representation of the M/M/1 queue.

## 3.2 Performance Indices

The analysis of an SPN model is usually aimed at the computation of more aggregate performance indices than the probabilities of individual markings. Several kinds of aggregate results are easily obtained from the steady-state distribution over reachable markings. In this section we quote some of the most commonly and easily computed aggregate steady-state performance parameters [18].

- The *probability of an event* defined through place markings (e.g., no token in a subset of places, or at least one token in a place while another one is empty, etc.), can be computed by adding the probabilities of all markings in which the condition corresponding to the event definition holds true. Thus, for example, the steady-state probability of the event $A$ defined through a condition that holds true for the markings $M_i \in \mathcal{M}$ is obtained as:

$$P\{A\} = \sum_{i:M_i \in \mathcal{M}} \pi_i \quad .$$

- The *pmf of the number of tokens in a place*, say $p_i$, can be obtained by computing the individual probabilities in the pmf as probabilities of the event "place $p_i$ contains $k$ tokens".

- The *average number of tokens in a place* can be computed from the pmf of tokens in that place.

- The *frequency of firing of a transition*, i.e., the average number of times the transition fires in unit time, can be computed as the weighted sum of the transition firing rate:

$$f_j = \sum_{i:t_j \in E(M_i)} \lambda_j(M_i)\pi_i$$

  where $f_j$ is the frequency of firing of $t_j$, $E(M_i)$ is the set of transitions enabled in $M_i$, and $\lambda_j(M_i)$ is the firing rate of $t_j$ in $M_i$.

- The *average delay of a token* in traversing a subnet in steady-state conditions can be computed using Little's formula [7,19]

$$E[T] = \frac{E[N]}{E[\gamma]}$$

  where $E[T]$ is the average delay, $E[N]$ is the average number of tokens in the process of traversing the subnet, and $E[\gamma]$ is the average input (or output) rate of tokens into (or out of) the subnet. This procedure can be applied whenever the interesting tokens can be identified inside the subnet (which can also comprise other tokens defining its internal condition, but these must be distinguishable from those whose delay is studied) so that their average number can be computed, and a relation can be established between input and output tokens (e.g., one output token for each input token).

As an example of a performance parameter which in the general case is difficult to compute, we may quote the distribution of the delay incurred by a token in traversing a subnet, or in completing a cycle through a net.

## 3.3 Generalized SPN

The key factor that limits the applicability of SPN models is the complexity of their analysis. This is due to many elements. The possibly very large number of reachable markings is by far the most critical one. Other aspects may however add to the model solution complexity. One of these is due to the presence in one model of activities that take place on a much faster (or slower) time scale than the one relating to the events that play a critical role on the overall performance. This results in systems of linear equations which are *stiff*, i.e., difficult to solve with an acceptable degree of accuracy by means of the usual numerical techniques. On the other hand, neglecting the "fast" (or "slow") activities may result in models which are logically incorrect. It may also happen that in the construction of the topology of an SPN model, the analyst inserts transitions that correspond to purely logical aspects of the system behaviour, so that no timing can be reasonably associated with them.

Generalized SPN (GSPN) were originally proposed by G. Balbo and G. Conte, together with the author, in order to tackle these problems [20]. The definition was later improved by the same authors together with G. Chiola, in order to better exploit the structural properties of the modeling tool [21].

GSPN models comprise two types of transitions:

**timed transitions** , which are associated with random, exponentially distributed firing delays, as in SPN, and

**immediate transitions** , which fire in zero time with priority over timed transitions.

Furthermore, inhibitor arcs are permitted, different priority levels of immediate transitions can be used, and weights are associated with immediate transitions.

A GSPN is thus an eight-tuple

$$GSPN = (P, T, \mathcal{P}, I, O, H, M_0, W)$$

where $(P, T, \mathcal{P}, I, O, H, M_0)$ is the underlying untimed PN model which comprises

- the *basic* underlying PN $(P, T, I, O, M_0)$,
- a set of inhibitor arcs $H \subset P \times T$,
- an assignment of priorities to transitions, $\mathcal{P}$, which associates lowest priority (0) with timed transitions and higher priorities ($\geq 1$) with immediate transitions,

and $W = (w_1, w_2, \ldots, w_n)$ is an array whose entry $w_i$

- is the parameter of the negative exponential pdf of the transition firing delay if $t_i$ is a timed transition,
- is a weight used for the computation of firing probabilities of immediate transitions if $t_i$ is an immediate transition.

In the graphical representation of GSPN, immediate transitions are drawn as segments, and timed transitions as (white or black) rectangular boxes.

The interpretation of a GSPN model is very similar to that of an SPN model, with the changes necessary to account for immediate transitions.

When a marking is entered, it is first necessary to ascertain whether it enables timed transitions only, or at least one immediate transition. Markings of the former type are called *tangible*, whereas markings of the latter type are called *vanishing* (or intangible).

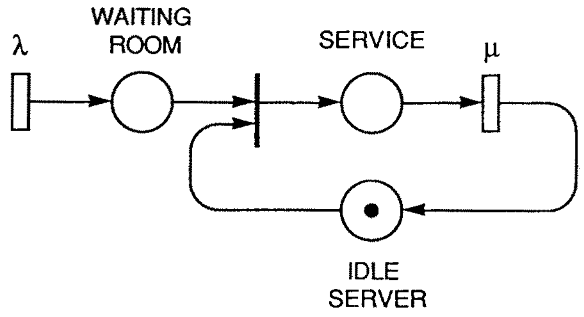


Figure 9: GSPN representation of the M/M/1 queue.

In the case of a tangible marking, the timers of the enabled timed transitions either resume their decrement, or are reset and then decremented, until one timed transition fires, exactly as in the case of SPN.

In the case of a vanishing marking, the selection of which transition to fire cannot be based on the temporal description, since all immediate transitions fire exactly in zero time. The choice is thus based on priorities and weights. The set of transitions with concession at the highest priority level is first found, and if it comprises more than one transition, the further selection, of probabilistic nature, is based on the transition weights according to the expression

$$P\{t_k\} = \frac{w_k}{\sum_{i:t_i \in E(M)} w_i}$$

where $E(M)$ is the set of enabled immediate transitions in marking $M$, i.e., of the transitions with concession at the highest priority level.

Note that the semantics of a GSPN model always assumes that transitions are fired one by one, even in a vanishing marking comprising nonconflicting enabled immediate transitions. The equivalence of this behaviour with the one resulting from the simultaneous firing of some immediate transitions in the model can be exploited to reduce the complexity of the solution algorithms [22].

The presence of inhibitor arcs and priorities reduces the number of reachable markings with respect to the basic underlying PN. The GSPN reachability set is equal to the one of the underlying PN which includes priorities and inhibitor arcs.

The analysis of a GSPN model requires the solution of a system of linear equations comprising as many equations as the number of reachable *tangible* markings. The infinitesimal generator of the CTMC associated with a GSPN model is derived with a contraction of the reachability graph labeled with the rates or weights of the transitions causing the change of marking.

A different approach to the analysis of GSPN models, which also implies a different semantics, is presented in [23].

As a first example of a GSPN model, in Figure 9 we show the GSPN representation of the M/M/1 queue described in Section 2.3. The possibility of using an immediate transition allows the explicit representation of the waiting room, of the service station, and of the idle server.

A somewhat more complex model is the GSPN representation of the M/M/2 queue with finite waiting room (5 positions) and finite user population (10 customers), shown in Figure 10. This queuing system is normally called M/M/2/5/10. The waiting room is represented by place $p_1$; idle servers are represented by tokens in place $p_3$; busy servers are represented by tokens in place $p_2$. Place $p_4$ contains the customers that may arrive to the queue. Place $p_5$ contains tokens that represent free positions in the waiting room, and that constitute the credits for the admission to the queue. The marking of $p_4$ determines the firing rate of the transition modeling customer arrivals, and the marking of $p_2$ determines the firing rate of the transition modeling customer departures.

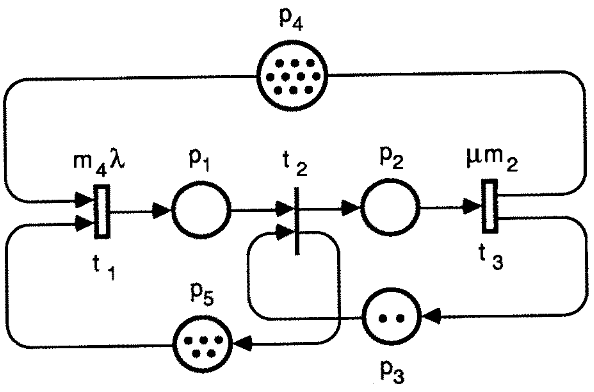


Figure 10: GSPN representation of the M/M/2/5/10 queue.

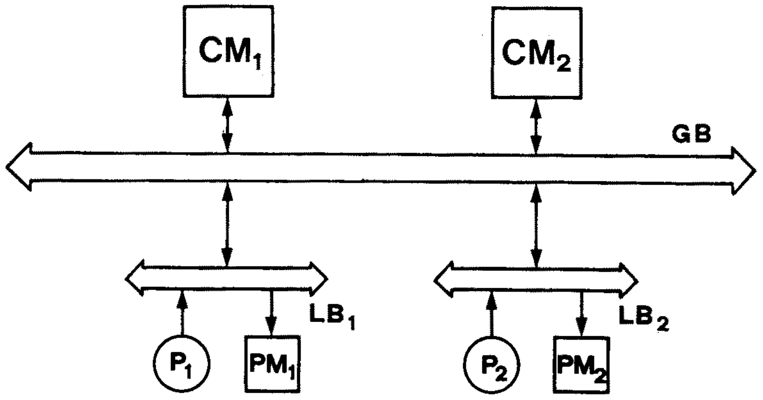


Figure 11: A two-processor system.

Indeed, finite population queuing systems are normally modeled using an arrival process with exponentially distributed interarrival times whose rate is proportional to the number of customers out of the queue.

As a third example of a GSPN model, we analyze the performance of the two-processor system depicted in Figure 11 [18]. Two processing units comprising a processor ($P_1$ and $P_2$) and a private memory ($PM_1$ and $PM_2$) connected by a local bus ($LB_1$ and $LB_2$), access two common memory modules ($CM_1$ and $CM_2$) through a global bus ($GB$). Processors can either be *active*, i.e., working in their private memory, or try and access a common memory module. Requests can be satisfied only if the global bus and the requested common memory are available (note that in this case the availability of the global bus implies the availability of both common memories, so that the model could be simplified). Processor active times and memory access times are assumed to be random variables with negative exponential pdf.

The resulting GSPN model of this two-processor system is shown in Figure 12. Tokens in place $p_1$ represent active processors. Transition $t_1$ represents the issue of a common memory access request. Transitions $t_2$ and $t_3$ model the choice of one of the two common memory modules. Tokens in places $p_5$ and $p_6$ represent the availability of the two common memory modules, respectively. A token in place $p_9$ indicates the availability of the global bus. Transitions $t_4$ and $t_5$ model the beginning of a common memory access, whose duration is represented by the delay associated with transitions $t_6$ and $t_7$, respectively. The firing rates of $t_1$, $t_6$, and $t_7$ are $\lambda$, $\mu$, and $\mu$, respectively.

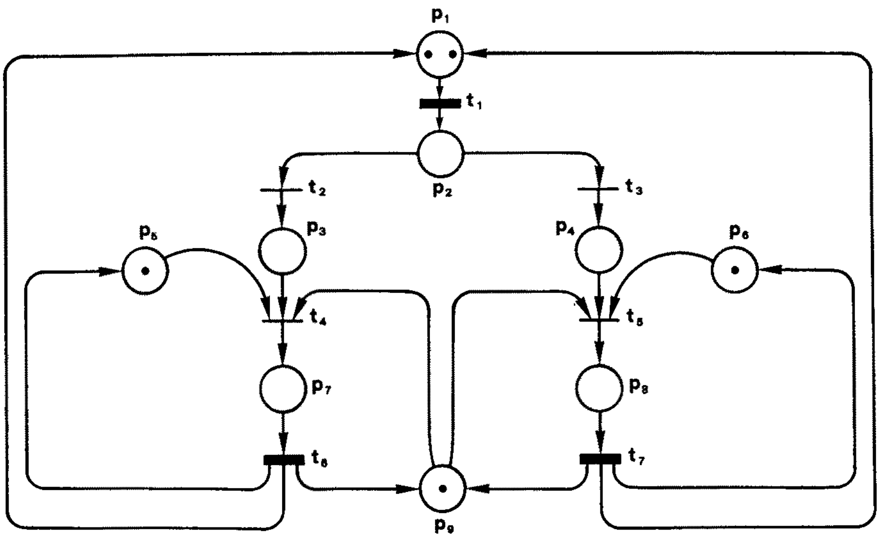


Figure 12: GSPN representation of the two-processor system in Figure 11.

Incorporating in the model the possibility of failure of the system components leads to the GSPN in Figure 13. Heavy solid lines constitute the basic model, depicted in Figure 12. Light solid lines represent the portion of the model corresponding to faults. Dashed lines represent the portion of the model corresponding to repairs. Place $p_{10}$ contains tokens representing failed processing units. Places $p_{11}$ and $p_{12}$ contain tokens representing the two failed common memory modules, respectively. Place $p_{13}$ may contain a token corresponding to the bus failure condition.

Since it would be unreasonable for a processor to request access to a failed common memory module, the two inhibitor arcs from $p_{11}$ to $t_2$, and from $p_{12}$ to $t_3$ may be introduced.

Assuming that the goal of the analysis is the evaluation of the two-processor system processing power, P, defined as the steady-state average number of processors which are active, working in their private memory, it can be obtained by adding the probabilities of all markings containing tokens in place $p_1$, weighted with the number of tokens in that place.

The numerical results are plotted in Figure 14 as a function of the parameter $\rho = \lambda/\mu$, corresponding to the product of the common memory request rate by the average common memory access time.

The five curves correspond to various assumptions. Curve 1 refers to the absence of faults. Curve 2 refers to a system where the bus cannot fail, and failed common memories cannot be requested. Curve 3 refers to a system where the bus can fail, and failed common memories cannot be requested. Curve 4 refers to a system where the bus cannot fail, and failed common memories can be requested. Curve 5 refers to a system where the bus can fail, and failed common memories can be requested.

In this case it is also possible to use Little's result to compute the average delay from the instant a common memory request is issued until the common memory access terminates. The subnet describing the common memory access comprises the whole GSPN model, with the exception of place $p_1$. Disregarding faults, for the sake of simplicity, the average number of tokens representing processors with pending and served memory requests in the subnet equals 2− P. The average input rate of tokens in the subnet equals P$\lambda$. Thus:

$$E[T] = \frac{2 - \mathrm{P}}{\mathrm{P}\lambda}$$

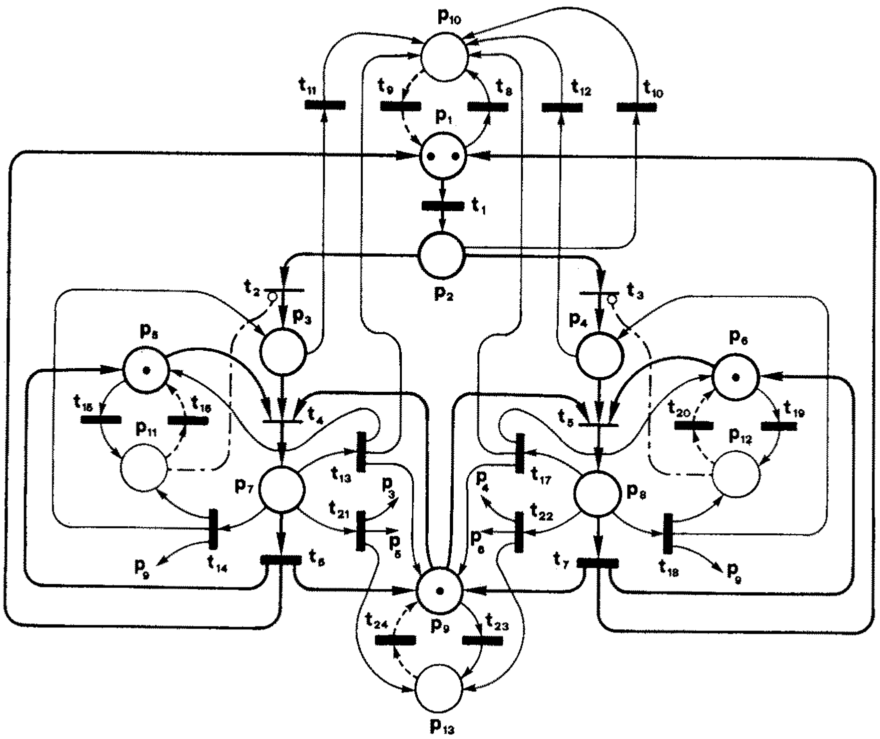


Figure 13: GSPN representation of the two-processor system in Figure 11, including failures and repairs.

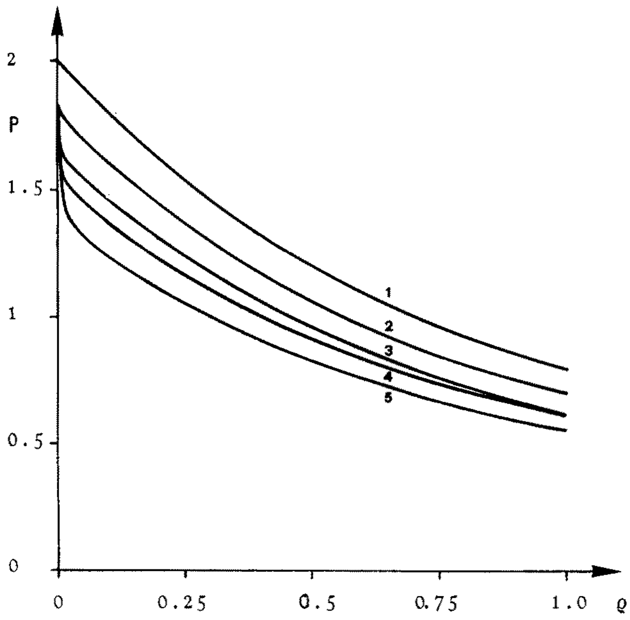


Figure 14: Processing power of the two-processor system in Figure 11.

Note that the tokens representing the common memories and the bus were not taken into account in the computation of $E[N]$.

The application of SPN and GSPN modeling techniques has been very productive in several areas. The factor that has however limited their acceptance as a modeling tool lies in the (graphical and computational) *complexity* of the models of realistic systems. We shall discuss the complexity issue in some detail in a later section. It must be also stressed that the use of SPN and GSPN heavily relies on the availability of adequate software *tools*, without which the model construction and solution is possible only for the smallest toy examples.

## 3.4 Generally Distributed Firing Times

Some research efforts were devoted to attempts to increase the modeling power of SPN and GSPN by allowing firing delays to be random variables with arbitrary pdf.

This resulted in the definition of extended SPN (ESPN) [24], deterministic and stochastic PN (DSPN) [25,26], and SPN with phase-distributed firing delays [27,28].

It is important to note that the existence in these models of random firing delays with pdf other than the negative exponential requires the definition of the SPN execution policy. Indeed, the equivalence among different model interpretations that was described in Section 3.1 rests on the memoryless property of firing delays, i.e., on their negative exponential pdf. A discussion of the effect of firing policies is contained in [27,28], where the usual policy is called *race with age memory*, where "race" indicates that transitions compete for firing (the competition is won by the transition that samples the shortest delay), and "age memory" indicates that the resampling is performed only after the transition firing. In other words, similarly to what was explained in Section 3.1, whenever a change of marking enables a transition that was not previously enabled since its last firing, this transition samples an instance of the firing delay from the associated *general* pdf, and sets a timer at the value of the sampled delay instance. While the transition is enabled, the timer is decreased at a constant speed. If the transition is disabled by the firing of a conflicting transition, the timer stops, and the decrement resumes when the transition is enabled again. When the timer reaches the value zero the transition fires. Other firing policies are also considered in [27,28], such as *race with enabling memory*, where the resampling is performed every time a transition becomes enabled, and *preselection*, where the transition to be fired is selected independently of the firing delays. The difference between the cases of race with age and enabling memory derives from the fact that in the case of general distributions the memoryless property does not hold, hence the residual time before firing when a transition that was disabled becomes enabled again has a different distribution from the firing time sampled from the distribution associated with the transition.

Unfortunately, the increased modeling power of these extended SPN models is paid with an increased complexity in their solution, so that these types of models are often used in the simulation context only.

Some recent results have however shown that in the case of DSPN it is possible to exploit the structure of the underlying PN model to obtain very significant reductions in the solution complexity [29]. Moreover, in a very recent paper [30], Henderson and Lucic define the conditions under which the steady-state distribution over markings depends only on the average firing delay of a transition, not on the distribution type. We shall return to these recent results in a later section.

## 3.5 Relations with Other Models

Several proposals, different from those considered in this paper, for the introduction of temporal specifications of a probabilistic nature into a PN model were published in the technical literature.

Two main streams can be identified:

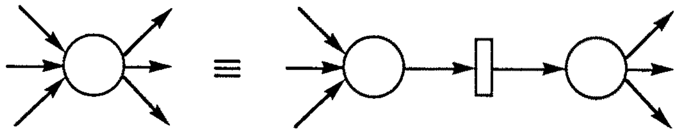


Figure 15: Representation of a timed place with the GSPN formalism.

- PN models in which timing is associated with places [31,32],
- PN models in which timed transitions fire in three phases [33,34,35,36].

Without the presumption of claiming the superiority of any one of the timed PN models with respect to others, we describe how the semantics of those two cases can be rendered with the GSPN notation.

When timing is associated with places, at the firing of an enabled transition tokens are removed from input places and deposited into output places with a single, indivisible operation. However, the tokens deposited into the transition output places are not immediately available for the enabling of other transitions. They become available only after the delay associated with the place has elapsed.

It is thus possible to translate a PN with timing associated with places into the GSPN formalism by replacing every timed place with a standard place, a timed transition and another standard place. The input arcs of the timed place are linked to the first untimed place, and the output arcs of the timed place depart from the second untimed place, as shown in Figure 15.

Note that the aging of tokens deposited in a timed place proceeds in parallel, so that the translation shown in Figure 15 requires that the firing rate of the timed transition be proportional to the number of tokens in its input place (borrowing the queuing jargon, the transition is said to be of *infinite server* type).

When timing is associated with transitions that fire in three phases, an enabled transition immediately starts firing, and removes tokens from its input places. Tokens are however not deposited into the transition output places until the transition delay has elapsed. They are "kept in the transition" for the whole transition delay.

In other words, the firing of a transition consists of three phases

1. a *start firing* phase, in which tokens are removed from input places,
2. a *firing in progress* phase, with which is associated the firing delay,
3. an *end firing* phase, in which tokens are deposited into output places.

The translation of a timed PN with transitions firing in three phases into the GSPN formalism is easily performed by substituting for each three-phase transition an immediate transition, a place, and a timed transition. The input arcs of the three-phase transition are connected to the immediate transition, and the output arcs of the three-phase transition depart from the timed transition, as shown in Figure 16.

Normally, three-phase timed transitions can initiate a new firing while another one is in progress. The timed transition in Figure 16 is thus again of infinite server type. When the three-phase timed transition can start a new firing only when the previous one is completed, the firing rate of the timed transition in Figure 16 is constant (the transition is said to be of *single server* type). An inhibitor arc can be added from the place to the immediate transition if the tokens must be kept in the input places of the three-phase timed transition during the transition delay.

It should be noted that in the case of timed transitions firing in three phases, conflicts are restricted to those transitions that become enabled at the same time instant. Indeed, it is not

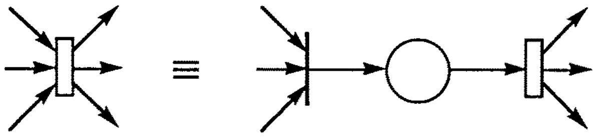


Figure 16: Representation of a three-phase timed transition with the GSPN formalism.

possible that a transition that becomes enabled at a later time disables by firing a previously enabled transitions. In order to incorporate this feature, which may in some cases be useful for the modeling of systems, into timed PN models in which transition fire in three phases, special arcs that "unfire" a transition were proposed [37].

## 3.6 Structural Analysis of SPN

As we already mentioned, SPN were initally proposed by researchers active in the applied stochastic modeling field as a convenient graphical notation for the abstract definition of Markovian models.

As a consequence, the basic definitions of SPN (and of their variations as well) were originally more concerned with the characteristics of the underlying stochastic process, rather than with the structure of the underlying PN model.

Thus, for example, the possibility of defining marking-dependent firing rates in SPN assumed that the analyst be aware of all reachable markings in order to correctly specify the marking dependency. Even more so was the initial definition of GSPN [20], in which the choice of the immediate transition to be fired in a vanishing marking was based on a user-defined switching distribution, which could be correctly defined only by knowing the GSPN reachability set.

With the revised GSPN definition [21], an effort was made to integrate the structural and the stochastic characteristics of the model, so that the model definition be possible at the net level, without requiring the generation of the reachability set.

One of the main advantages of the basic SPN model, besides its great simplicity and the resulting ease of use, is the identity of its reachability graph with the one of the underlying PN. This means that all the results available for the structural analysis of place/transition nets can be readily applied to the SPN environment.

In the case of GSPN, instead, the reachability set is identical to the one of the underlying place/transition net with inhibitor arcs and priorities, an untimed model that received very little attention by net theoreticians. Fortunately, however, the presence of inhibitor arcs and priorities only restricts the reachability set with respect to the one of the basic underlying net which is obtained by removing timing, inhibitor arcs and priorities. This means that some of the structural properties valid for the basic underlying PN are retained by the GSPN.

Indeed, the restriction of the reachability set guarantees that all place invariants found with the study of the basic underlying PN still hold for the GSPN; in principle there might exist other place invariants which hold for the GSPN, but are not valid for the basic underlying PN. Experience has shown that this normally does not happen. For what concerns transition invariants, the presence of priorities and inhibitor arcs may render unfeasible an invariant which is firable in the basic underlying PN.

In the case in which general firing delay pdf are allowed, the structural analysis problem may be much more difficult, since it may happen that the probabilistic characteristics of the model have an impact on the qualitative behaviour in the sense that, although two transitions may be simultaneously enabled, one of them cannot fire due to timing constraints. This means that the structural properties of the net cannot be completely studied even by taking into account possible priorities and inhibitor arcs; timing must be considered as a structural component.

A sufficient condition to avoid this undesirable influence of the temporal specification on the model qualitative behaviour is that all firing delay pdf have unlimited support [27,28], as we already noted. This restriction, however, does not permit the use of constant firing delays, which are very important from the modeling point of view.

It should however be stressed that the structural properties of SPN models are today used to either ease the model definition, or compute very partial results [38,39]. Nothing is yet known on the role that the structural properties of the model can play for the reduction of the complexity of the computation of the pmf over the reachable markings.

## 3.7 Current Research Activities

Several groups of researchers are presently active in the field of SPN. We mention here some of the current research efforts, aiming at a unitary view of the research field rather than at a comprehensive list of isolated activities.

Much of the current research work in the SPN field is devoted to the application of SPN and related models to performance and reliability studies of a very diverse gamut of systems, including

- distributed computing systems architectures,
- distributed software,
- communication protocols,
- manufacturing systems,
- VLSI,
- data base,
- inventory and logistics,
- object-oriented systems,
- real-time systems.

In some of these application studies the performance analysis aspect is integrated with the formal proof of correctness of the system under investigation, exploiting the formal system description obtained with the PN formalism (see for example [40]).

As we already mentioned, the main problem in the use of SPN techniques for the analysis of real-life systems originates from the complexity in the model solution. It is often the case that models comprise such monstruously large state spaces that the generation of the reachability set is too costly (both in time and in space) to be performed. Several research efforts are thus devoted to attempts to reduce the solution cost.

A possible approach is to resort to simulation techniques, rather than trying to numerically solve the model. By so doing, the problems originating from the space complexity are removed, since the generation of the reachability set is not necessary any more, but the burdens inherent to the time complexity remain, as always in the case of simulation. On the other hand, Haas and Shedler have shown [41,42,43] that the modeling power of SPN in the simulation framework is remarkable, since they are equivalent to generalized semi-Markov processes.

Other approaches that can be followed for the reduction of the model solution complexity concern the utilization of the structural properties of the underlying PN models to devise efficient solution algorithms. Place invariants can be very useful to improve the space efficiency of the generation of the reachability set [44]. Causal connection and mutual exclusion, together with invariants, can be exploited for the identification of submodels that in some special cases can be separately studied, providing results whose subsequent combination yields the desired global solution [29].

The effort to use structural properties of the underlying PN models in the various phases of a performance study is demonstrated by the use of place and transition invariants for the study of the model ergodicity, as well as for the model debugging process; by the use of place invariants, causal connection, mutual exclusion, and confusion to verify that a model can be classified in a special class; by the use of transition invariants to identify bottlenecks; by the use of conflicts to ease the definition of a GSPN model.

The graphical complexity of SPN models can often be reduced if a high level description is used. The definition of high-level and colored SPN has already appeared in the literature [45,46,47], but works are still in progress in order to obtain the most adequate balance between modeling power and simplicity of use.

Finally, some researchers are studying the possibility of combining SPN with queues in order to obtain an integrated performance evaluation methodology that enjoys the nice properties of both approaches [48,49].

## 3.8 Open Problems

Once more, the main open problems have to do with the reduction of the solution complexity. The most important questions could be phrased as follows:

1. how can an SPN model be decomposed into submodels which can be studied in isolation, and whose results can later be combined to obtain (either exactly or approximately) the solution of the global model?

2. how can the structural characteristics of the underlying PN be exploited in the model solution phase?

3. how can other performance measures (such as delay distributions) be computed?

The answer to the first question implies the definition of a technique for the modular costruction of SPN models, i.e., rules for the decomposition of a large model into submodels, and for the composition of submodels into one model, as well as the identification of classes of SPN which allow some sort of factorization of their solution.

The answer to the second question requires the discovery of the relations between the structure of the underlying PN model and the structure of the stochastic process, and the design of techniques that exploit this structure in the model solution.

The answer to the third question is probably possible only for special SPN classes; solutions for trivial cases are indeed easy to obtain. It is however necessary to identify the nontrivial SPN for which these parameters can be computed, and to define the algorithms that lead to the desired result.

Some results related to questions 1 and 2 were obtained by Florin and Natkin [50,51,52] and Robertazzi and Lazar [53,54,55], and others have been derived very recently by Henderson, Taylor, and Lucic [56]. It should be emphasized that a comprehensive solution to the first question would most likely have the same impact on the SPN field as the BCMP theorem [57] had on queuing theory. The BCMP theorem identified the classes of queuing networks which admit a product form solution, it provided the algorithms for their computation, and it gave a tremendous impulse to the application of queuing techniques in the performance evaluation field.

## 4 CONCLUSIONS

This paper is meant to be an introduction to stochastic PN for Petri net experts who are not familiar with the stochastic performance modeling field. For this reason, a brief overview of the performance analysis field, of Markovian processes, and of queuing models was provided. The characteristics of SPN models were then described together with the motivations that led to their definition. Some examples were also used to illustrate the possible application of the SPN methodology to the performance analysis of systems.

A concise summary of the ongoing research activities and of the most important open problems has been provided, with the hope that some of the readers will become interested in the SPN field, and obtain new results that will expand the applicability of the SPN methodology.

## 5 ACKNOWLEDGMENTS

This work was financially supported in part by the Italian Ministry for Education, and by the NATO Research Grant n. 012.81.

The author is grateful to G. Balbo, G. Chiola, G. Conte, F. De Cindio, A. Fumagalli, C. Simone, and the three anonymous reviewers who carefully read the early versions of this paper and provided useful comments for their improvement.

## References

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

[2] **W. Reisig**, *Petri Nets: an Introduction*, Springer Verlag, 1985.

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

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

[5] **W. Feller**, *An Introduction to Probability Theory and Its Applications*, John Wiley, New York, NY, 1966.

[6] **R.A. Howard**, *Dynamic Probabilistic Systems*, John Wiley, New York, NY, 1971.

[7] **L. Kleinrock**, *Queueing Systems Volume I: Theory*, John Wiley, New York, NY, 1975.

[8] **J.W. Cohen**, *The Single Server Queue*, American Elsevier, New York, NY, 1969.

[9] **D.R. Cox and W.L. Smith** *Queues*, John Wiley, New York, NY, 1961.

[10] **R.B. Cooper**, *Introduction to Queueing Theory*, Mac Millan New York, NY, 1972.

[11] **S.S. Lavenberg**, *Computer Performance Modeling Handbook*, Academic Press, New York, NY, 1983.

[12] **E. Gelenbe and I. Mitrani**, *Analysis and Synthesis of Computer Systems*, Academic Press, New York, NY, 1980.

[13] **M. Ajmone Marsan, G. Balbo and G. Conte**, *Performance Models of Multiprocessor Systems*, The MIT Press, Cambridge, Massachusetts, 1986.

[14] **S. Natkin**, "Les Reseaux de Petri Stochastiques et leur Application a l'Evaluation des Systèmes Informatiques", *Thèse de Docteur Ingegneur*, CNAM, Paris, France, 1980.

[15] **M.K. Molloy**, "On the Integration of Delay and Throughput Measures in Distributed Processing Models", *Ph.D. Thesis*, UCLA, Los Angeles, CA, 1981.

[16] **F.J.W. Symons**, "Introduction to Numerical Petri Nets, a General Graphical Model of Concurrent Processing Systems", *Australian Telecommunications Research*, Vol. 14, n. 1, pp. 28-33, January 1980.

[17] **F.J.W. Symons**, "The Description and Definition of Queueing Systems by Numerical Petri Nets", *Australian Telecommunications Research*, Vol. 13, pp. 20-31, 1980.

[18] **M. Ajmone Marsan, A. Bobbio, G. Conte, A. Cumani** "Performance Analysis of Degradable Multiprocessor Systems using Generalized Stochastic Petri Nets", *Distributed Processing T-C Newsletters*, IEEE Computer Society, n. 6, SI-1, 1984, pp. 47-54.

[19] **J. D. C. Little**, "A Proof of the Queueing Formula $L = \lambda W$", *Operations Research*, Vol. 9, 1961, pp. 383-387.

[20] **M. Ajmone Marsan, G. Balbo, G. Conte**, "A Class of Generalized Stochastic Petri Nets for the Performance Analysis of Multiprocessor Systems", *ACM Transactions on Computer Systems*, Vol. 2, n. 1, May 1984, pp. 93-122.

[21] **M. Ajmone Marsan, G. Balbo, G. Chiola, G. Conte**, "Generalized Stochastic Petri Nets Revisited: Random Switches and Priorities", *Proceedings of the International Workshop on Petri Nets and Performance Models*, Madison, WI, USA, August 1987.

[22] **G. Balbo, G. Chiola, G. Franceschinis, G. Molinar Roet**, "On the Efficient Construction of the Tangible Reachability Graph of Generalized Stochastic Petri Nets", *Proceedings of the International Workshop on Petri Nets and Performance Models*, Madison, WI, USA, August 1987.

[23] **H. H. Ammar, R. W. Liu**, "Analysis of the Generalized Stochastic Petri Nets by State Aggregation", *Proceedings of the International Workshop on Timed Petri Nets*, Torino,Italy, July 1985.

[24] **J.B. Dugan, K.S. Trivedi, R.M. Geist, V.F. Nicola**, "Extended Stochastic Petri Nets: Applications and Analysis", *Proceedings of PERFORMANCE '84*, Paris, France, December 1984.

[25] **M. Ajmone Marsan, G. Chiola**, "On Petri Nets with Deterministic and Exponential Transition Firing Times", *Proceedings of the 7-th European Workshop on Application and Theory of Petri Nets*, Oxford, England, June 1986.

[26] **M. Ajmone Marsan, G. Chiola**, "On Petri Nets with Deterministic and Exponentially Distributed Firing Times", in: G.Rozenberg (editor), *Advances in Petri Nets 1987, Lecture Notes on Computer Science*, n. 266, Springer Verlag, 1987.

[27] **M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, A. Cumani**, "On Petri Nets with Stochastic Timing", *Proceedings of the International Workshop on Timed Petri Nets*, Torino,Italy, July 1985.

[28] **M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, A. Cumani**, "The Effect of Execution Policies on the Semantics and Analysis of Stochastic Petri Nets", *IEEE Transactions on Software Engineering*, Vol. SE-15, n. 7, July 1989.

[29] **M. Ajmone Marsan, G. Chiola, A. Fumagalli**, "Improving the Efficiency of the Analysis of DSPN Models", *Proceedings of the 9-th European Workshop on Application and Theory of Petri Nets*, Venezia, Italy, June 1988.

[30] **W. Henderson, D. Lucic**, "Application of Generalized Semi Markov Processes to Stochastic Petri Nets", *Proceedings of the International Seminar on Performance of Distributed and Parallel Systems*, Kyoto, Japan, December 1988.

[31] **J. Sifakis**, "Perfomance Evaluation of Systems Using Petri Nets", in *Net Theory and Applications*, edited by G. Goos and J. Hartmanis, Springer Verlag, New York, NY, 1979.

[32] **C.Y. Wong, T.S. Dillon, K.E. Forward**, "Timed Places Petri Nets with Stochastic Representation of Place Time", *Proceedings of the International Workshop on Timed Petri Nets*, Torino, Italy, July 1985.

[33] **R.R. Razouk, C.V. Phelps**, "Performance Analysis using Timed Petri Nets", *Proceedings of the International Conference on Parallel Processing*, August 1984.

[34] **M.A. Holliday, M.K. Vernon**, "A Generalized Timed Petri Net Model for Performance Analysis", *Proceedings of the International Workshop on Timed Petri Nets*, Torino, Italy, July 1985.

[35] **W.M. Zuberek**, "Timed Petri Nets and Preliminary Performance Evaluation", *Proceedings of the 7-th Annual Symposium on Computer Architecture*, La Baule, France, May 1980.

[36] **W.M. Zuberek**, "Performance Evaluation using Timed Petri Nets", *Proceedings of the International Workshop on Timed Petri Nets*, Torino, Italy, July 1985.

[37] **W.M. Zuberek**, "M-Timed Petri Nets, Priorities, Preemptions, and Performance Evaluation of Systems", in *Advances on Petri Nets '85* edited by G. Rozenberg, LNCS 222, Springer Verlag, 1986.

[38] **M.K. Molloy**, "Fast Bounds for Stochastic Petri Nets", *Proceedings of the International Workshop on Timed Petri Nets*, Torino, Italy, July 1985.

[39] **M.K. Molloy**, "Structurally Bounded Stochastic Petri Nets", *Proceedings of the International Workshop on Petri Nets and Performance Models*, Madison, WI, USA, August 1987.

[40] **G. Balbo, S. C. Bruell, G. Chiola, P. Chen**, "An Example of Validation and Evaluation of a Concurrent Program: Lamport's Fast Mutual Exclusion Algorithm", submitted for publication.

[41] **P.J. Haas, G.S. Shedler**, "Regenerative Simulation of Stochastic Petri Nets", *Proceedings of the International Workshop on Timed Petri Nets*, Torino, Italy, July 1985.

[42] **P.J. Haas, G.S. Shedler**, "Regenerative Stochastic Petri Nets", *Performance Evaluation*, Vol. 6, n. 3, September 1986, pp. 189-204.

[43] **P.J. Haas, G.S. Shedler**, "Stochastic Petri Nets with Timed and Immediate Transitions", *Stochastic Models*, to appear.

[44] **G. Chiola**, "Compiling Techniques for the Analysis of Stochastic Petri Nets", *Proceedings of the 4-th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation*, Palma de Mallorca, Spain, September 1988.

[45] **A. Zenie**, "Colored Stochastic Petri Nets", *Proceedings of the International Workshop on Timed Petri Nets*, Torino, Italy, July 1985.

[46] **C. Marinescu, Chuang Lin**, "On Stochastic High Level Petri Nets", *Proceedings of the International Workshop on Petri Nets and Performance Models*, Madison, WI, USA, August 1987.

[47] **G. Chiola, G. Bruno, T. Demaria**, "Introducing a Color Formalism into Generalized Stochastic Petri Nets", *Proceedings of the 9-th European Workshop on Application and Theory of Petri Nets*, Venezia, Italy, June 1988.

[48] **G. Balbo, S. C. Bruell, S. Ghanta**, "Combining Queueing Network and Generalized Stochastic Petri Net Models for the Analysis of a Software Blocking Phenomenon", *Proceedings of the International Workshop on Timed Petri Nets*, Torino, Italy, July 1985.

[49] **G. Balbo, S. C. Bruell, S. Ghanta**, "Combining Queueing Network and Generalized Stochastic Petri Net Models for the Analysis of Some Software Blocking Phenomena", *IEEE Transactions on Software Engineering*, Vol. SE-12, n. 4, April 1986, pp. 561-576.

[50] **G. Florin, S. Natkin**, "On Open Synchronized Queuing Networks", *Proceedings of the International Workshop on Timed Petri Nets*, Torino, Italy, July 1985.

[51] **G. Florin, S. Natkin**, "Les Reseaux de Petri Stochastiques", *Technique et Science Informatiques*, Vol. 4, n. 1, February 1985, pp. 143-160.

[52] **G. Florin, S. Natkin**, "A Necessary and Sufficient Saturation Condition for Open Synchronized Queueing Networks", *Proceedings of the International Workshop on Petri Nets and Performance Models*, Madison, WI, USA, August 1987.

[53] **A. A. Lazar, T. G. Robertazzi**, "Markovian Petri Net Protocol Models with Product Form Solution", *Proceedings of INFOCOM 87*, San Francisco, CA, USA, March 1987.

[54] **A. A. Lazar, T. G. Robertazzi**, "The Algebraic and Geometric Structure of Markovian Petri Network Lattices", *Proceedings of the 24-th Annual Allerton Conference on Communications, Control and Computing*, Urbana-Champaigne, Illinois, USA, 1986.

[55] **I. Y. Wang, T. G. Robertazzi**, "Service Stage Petri Net Protocols with Product Form", submitted for publication.

[56] **W. Henderson, P. Taylor, D. Lucic**, "A Net Level Performance Analysis of Stochastic Petri Nets with Conflict Sets", submitted for publication.

[57] **F. Baskett, K. M. Chandy, R. R. Muntz, F. Palacios**, "Open, Closed and Mixed Networks of Queues with Different Classes of Customers", *Journal of the ACM*, Vol. 22, n. 2, April 1975, pp. 248-260.