PETRI NETS IN PERFORMANCE ANALYSIS: AN INTRODUCTION

M. Ajmone Marsan[1], A. Bobbio[2], and S. Donatelli[2]

[1] Dipartimento di Elettronica
Politecnico di Torino – Italy
ajmone@polito.it

[2] Dipartimento di Informatica
Università di Torino – Italy
bobbio,susi@di.unito.it

Abstract. *In this tutorial paper, the authors discuss the motivations that led to the adoption of Petri nets for performance evaluation, define the class of Petri nets that is most frequently used for performance analysis, and present the subclasses that allow a simpler derivation of performance metrics. Definitions and discussions are paralleled with examples, thus visualizing the strong and weak points of the different alternatives.*

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

1 Introduction

Petri nets (PN) [86, 87, 91] were originally developed and 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 within the basic, untimed models.

This fact was recognized about twenty years ago, and led to several different proposals for the introduction of temporal specifications in PN. The main alternatives that characterize the different proposals concern

- the PN elements associated with timing (normally either places or transitions, but some authors also looked into the possibility of defining timed arcs or tokens),
- the firing semantics in the case of timed transitions (either atomic firing or firing in three phases),

– the nature of the temporal specification (either deterministic or probabilistic),
– the conflict resolution policy.

In this tutorial paper we consider PN models that are augmented with a temporal specification by associating a (possibly null) firing delay with transitions. The transition firing operation is assumed to be atomic, i.e., tokens are removed from input places and put into output places with a single, indivisible operation, after the transition firing delay has elapsed. The specification of the firing delay of timed transitions is of probabilistic nature, so that either the probability density function (pdf) or the cumulative distribution function (Cdf) of the delay associated with a transition needs to be specified. Such functions may be general, or even degenerate, thus allowing the definition of constant (possibly null) delays. We refer to this type of timed Petri nets as Generally Distributed Timed Transitions Stochastic Petri Nets (GDTT_SPN).

Such timed Petri nets are those most frequently used in the field of performance evaluation. Evidence of this fact can be gathered by the observation of the papers presented at the series of International Workshops on Petri Nets and Performance Models, the leading forum for the presentation of novel results in this field.

In this work we review the different classes of GDTT_SPN that have been proposed in the literature, and we shall consider in some details two special subclasses of GDTT_SPN: i) Stochastic Petri Nets (SPN), where all transition firing delays are non-null and have negative exponential pdf, ii) generalized SPN (GSPN), where immediate (null-delay) transitions are freely mixed with timed transitions associated with exponentially distributed non-null random firing delays.

The goal of this paper is to provide a tutorial introduction to GDTT_SPN and their subclasses, discussing several aspects related to their use for performance analysis of complex systems, specially emphasizing their strong and weak points, and the feasibility of their solution.

This tutorial paper is addressed to PN experts who are not familiar with the stochastic performance modelling 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 are summarized. Basic tools are discussed first, with special attention to stochastic processes with Markovian characteristics, and their generalizations. The most common model specification techniques are then presented, starting from queues and queueing networks, and arriving at PN-based approaches; along the path from one modelling paradigm to the other we shall note an increase in modelling power, that is however paid with a reduction in the ease of the model solution.

Section 3 contains the general definition of GDTT_SPN models, and a discussion of their possible use for performance analysis.

Sections 4 and 5 are specifically addressed to the two subclasses of GDTT_SPN that we discuss in more details: SPN, and GSPN, respectively. The presentation of the different modelling paradigms is paralleled with examples of their appli-

cation, thus visualizing the strong and weak points of the different alternatives.

Finally, Section 6 provides some concluding remarks, and comments on the present and future trends of research in this 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 and telecommunications fields.

During the design process, measurements on real systems are obviously not possible, and also prototype implementations present insurmountable 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: *modelling*. 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 embedded into the computer program that simulates the system, whereas in the case of analytical models the description is given in mathematical terms. Goal of the analysis is to evaluate a set of "performance indices," like, for example, the percentage of time the system is idle, or the average amount of useful work performed by the system in a fixed period of operation, or the variability of the quality of the service offered to the final (human) user of the system.

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, or that two transfers of the same file produce identical copies), 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 and orderings of 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 on 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 dices or cards (excluding the tricks played by magicians, that, when successful, leave no space for casuality).

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. Many text books on stochastic processes are available, see for example [70].

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 $2\pi i$ (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 characterization of a random variable X is given in terms of its Cdf

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

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

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

and

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

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

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

In the case of random variables assuming values in a discrete set, instead of using their pdf, which is a generalized function, it may be more convenient to use their probability mass function (pmf)

$$\boldsymbol{p}_X = (p_1, p_2, p_3, \ldots)$$

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 probabilistic characterization of a random vector $\boldsymbol{X}$ comprising n random variables $X_i,\ i = 1, 2, \ldots, n$ is given either by the joint Cdf of the n random variables:

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

or by their joint pdf

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

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

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 $T = [0, \infty)$, and t is usually interpreted as "time".

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 modelling the state of a door (either closed or open). Each sample path is a function of time made of steps 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). Figure 1 depicts the sample path of a continuous-time stochastic process: each step represents the sojourn time in a state (black dots denote right-continuity). 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.

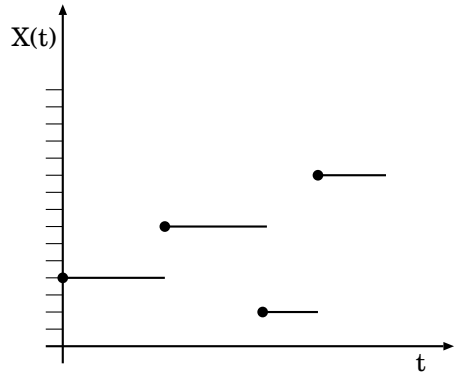

Fig. 1. Sample path of a continuous-time stochastic process.

2.2 Markov Processes and Their Generalizations

The complete probabilistic characterization of a random process requires the characterization of any random vector $\boldsymbol{X}$ comprising an arbitrary number k of random variables $X(t_i)$ extracted from the process at any set of time instants $\{t_i\}_{i=1}^{k}$.

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*

$$P\{X(\tau) \leq x | X(t), t \in [0, \theta]\} = P\{X(\tau) \leq x | X(\theta) = y\}$$

for any $\tau > \theta$.

Note that the Markovian property defines a stochastic process for which the behaviour in the future (at some time τ) depends only on the present situation (at time θ), not on the past history. 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, and in the possibility of coping with the main sources of memory in the system behaviour with an accurate definition of the process state.

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 (DTMC). If the parameter t is continuous, the process is a continuous-time Markov chain (CTMC). The time spent in states of a CTMC is a random variable with negative exponential pdf, as we shall see.

Another class of stochastic processes with discrete state space that allow a simple description and analysis is that of *Semi-Markov Processes* (SMP). Calling

δ_i the instant of time in which the stochastic process $X(t)$ changes state for the ith time, and Y_i the random variable that describes the state of the process between δ_i and δ_{i+1}, then the stochastic process $X(t)$ is Semi-Markov if and only if:

$$P\{Y_{n+1} = j, \delta_{n+1} - \delta_n \leq \tau | (Y_k, \delta_k), k \in [0, n]\} = P\{Y_{n+1} = j, \delta_{n+1} - \delta_n \leq \tau | Y_n = i\}$$

the latter probability is often written as $H_{ij}(\tau)$, to emphasize the dependency of the process future behaviour on the current state and the elapsed sojourn time in that state.

The stochastic sequence $\{Y_n, n \geq 0\}$ is a DTMC, called the Embedded Markov Chain (EMC) of the SMP. Therefore, also in this case, at the instant in which the state of $X(t)$ changes, the future of the stochastic process only depends on the present state, but now the pdf of the time spent in states is no longer exponential.

A further generalization of both CTMC and SMP is obtained with Markov Regenerative Processes (MRP), that allow the identification of a number of instants when the process changes state and "regenerates": at these instant the future evolution of the process only depends on the current state.

A stochastic process $X(t)$ is a MRP if and only if it comprises a renewal sequence of random variables $\{(Y_n, t_n), n \geq 0\}$ such that

$$P\{X(t_n + \tau) = j | X(t), t \in [0, t_n], Y_n = i\} = P\{X(\tau) = j | Y_0 = i\} \qquad (1)$$

The sequence of states at regeneration instants forms a DTMC, but now between two consecutive regeneration points it is possible to have many (possibly infinitely many) changes of states, and the distribution of the time between two consecutive regeneration points needs not be exponential.

2.3 Continuous-time Markov chains

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[1], and μ 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 past history), but only on the present state, as required by the Markovian property.

Note that the negative exponential pdf is characterized by just one parameter, μ, whose inverse μ^{-1} identifies the average value of the random variable.

[1] The unit step function $u(t)$ is such that $u(t) = 0$ for $t < 0$, and $u(t) = 1$ for $t \geq 0$.

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 averages 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 labelled directed graph whose vertices are labelled with the CTMC states, and whose arcs are labelled 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.

In Figure 2 we show the state transition rate diagram for a CTMC with two states (closed and open door in our previous example), for which the average sojourn time in state 1 is λ^{-1}, and the average sojourn time in state 2 is μ^{-1}. The infinitesimal generator for such a CTMC is

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

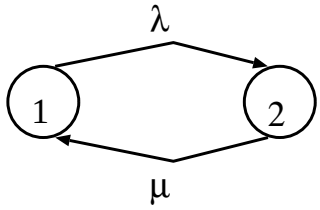

Fig. 2. State transition rate diagram for a CTMC with two states.

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* [42, 53, 64].

Denoting by

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

the probability that the CTMC is in state i at time t, the pmf at time t

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

is defined by the differential equation

$$\frac{d\boldsymbol{\pi}(t)}{dt} = \boldsymbol{\pi}(t)\ \boldsymbol{Q}$$

with initial condition $\boldsymbol{\pi}(0)$, whose solution can be expressed as

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

where $e^{\boldsymbol{Q}t}$ is the matrix exponential defined by

$$e^{\boldsymbol{Q}t} = \sum_{k=0}^{\infty} \frac{(\boldsymbol{Q}t)^k}{k!}$$

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

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

augmented with the normalization condition

$$\sum_i \pi_i = 1$$

The interested reader can find in [96, 99] an in-depth treatment of the problem posed by the solution of the above equations.

A modelling interpretation of the steady-state probabilities is the following: π_i is the probability according to which a random observer finds the system in equilibrium at state i, or equivalently the percentage of time that the system spends in state i when in equilibrium conditions.

The cost of solving the linear system $\boldsymbol{\pi Q} = \mathbf{0}$ is polynomial in the number of states. Iterative techniques are often applied, whose cost per iteration is of the order of the number of nonzero elements in $\boldsymbol{Q}$, i.e., of the order of the number of arcs in the state transition diagram.

From the steady-state pmf it is possible to derive many parameters of interest to quantify the system performance.

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 past history. To this purpose we assume that:

- the time periods during which the lamp is off are exponentially distributed with parameter β,
- the time periods during which the lamp is on are exponentially distributed with parameter α,
- the lightbulb lifetime (sum of the durations of the on periods before a fault) is exponentially distributed with parameter μ,
- the lamp repair time is exponentially distributed with parameter λ.

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

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

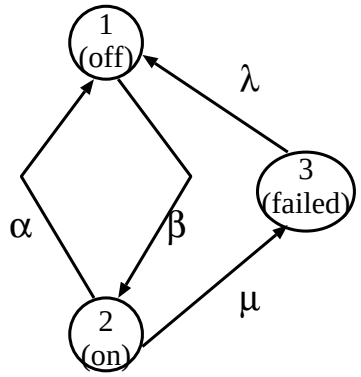

Fig. 3. State transition rate diagram for the CTMC describing the behaviour of a lamp.

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

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

Note that the first three equations of the linear 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; however, the solution is unique when considering the fact that probabilities must sum to 1.

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

- π_2 is the fraction of time in which the lamp is on,
- π_3 is the fraction of time in which the bulb is failed,
- $[\lambda\pi_3] = [\mu\pi_2]$ is the mean number of failures in unit time (the failing throughput),
- $[\lambda\pi_3]^{-1} = [\mu\pi_2]^{-1}$ 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.

From the transient solution $\boldsymbol{\pi}(t)$ we can derive other interesting performance indices:

- $\pi_2(t)$ probability of the lamp being on at time t,
- $\pi_3(t)$ probability of the bulb being failed at time t,

and, if no repair is possible in the system (i.e., the arc between states 3 and 1 in the CTMC model has rate 0, or is removed), we can compute the probability of the bulb not having yet failed at time t as $1 - \pi_3(t)$.

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 λ comprises probabilities

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

assuming that

$$\pi_0(0) = 1$$

These probabilities form a Poisson pmf.

Constructing models of complex systems directly at the CTMC level is generally *difficult*, mainly due to the need of choosing an appropriate state definition, and enumerating all states in the evaluation of transition rates. For this reason, *more abstract* probabilistic modelling tools were proposed. The main such tools are based on queueing theory or Petri nets.

2.4 Queues

A queue [7, 43, 45, 47, 55, 69, 71] 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.

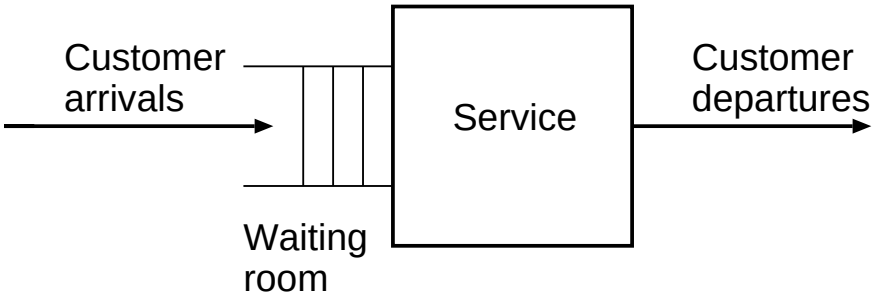

Fig. 4. Pictorial representation of a queue.

A queue is a compact description of a probabilistic (not necessarily Markovian) 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 queueing 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 identifies the arrival process as Markovian, and precisely as a Poisson process with a fixed rate, say λ. The second symbol M identifies the service time as Markovian (hence with negative exponential pdf); an average service time μ^{-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 discipline 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

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

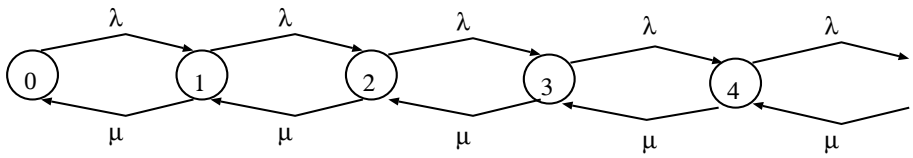

Fig. 5. State transition rate diagram of the CTMC generated by the M/M/1 queue.

The solution of the M/M/1 queue at time t, leads to a rather complicate expression, that can be found, for example, in [69]. When $\lambda < \mu$ the CTMC is ergodic, and the steady-state probabilities can be expressed as:

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

Many queueing models exist with more elaborate characteristics for what concerns 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 queueing discipline.

Nevertheless, a single queue may not be adequate to describe complex system behaviours, where customers may require the service of many different servers, in different orders. For this reason, a more flexible formalism was introduced, and it is still one of the most popular in performance evaluation: queueing networks.

2.5 Queuing networks

A queueing network [7, 55, 71] is a system of interconnected queues in which customers circulate, and possibly arrive from, and leave to, the outside world. When no arrivals from, and departures to, the external world are possible, the queueing 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, an open queueing network comprising three queues is depicted in Figure 6.

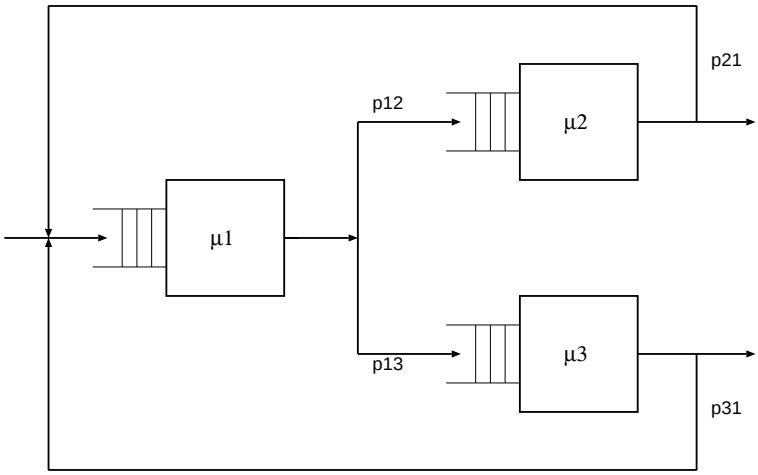

Fig. 6. An open queueing network comprising three queues.

With queueing 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 modelling field for a wide gamut of different application areas, such as telecommunications, computers, manufacturing, and transportation. For example, Figure 6 can be considered as a description of a very simple manufacturing process: servers at queues represent machines, and customers are products that are being manufactured. Each product is first worked at machine 1, and it is then moved to either machine 2 or 3. After being worked at machine 2 or 3 the product is either considered finished, and it leaves the system, or it may require to repeat the sequence of operations on machine 1 followed by machine 2 or 3.

The main reason for which queueing 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 queueing network can be *factored* in the product of the steady-state solutions of the individual queues, and hence obtained with very limited complexity (typically polynomial in the number of queues and customers).

It should be stressed that queueing 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 of limited practical interest.

The shortcomings of queueing-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 most of these features, quite common in distributed systems, generally destroy the product-form characteristic, so that

even a simple queueing model must be translated into its corresponding CTMC for the solution phase.

To cope with the lack of modelling power of queueing networks, many authors have introduced special "queues", that allow the description of the phenomena mentioned above, but there is no commonly agreed language for extended queueing networks, and this special queues usually do not have a formally defined semantics.

2.6 Timed Petri nets

When blocking and synchronization phenomena are important characteristics of the system to be modelled, their description with queueing networks is not natural, and the model solution must be obtained (with few exceptions) from the CTMC translation of the queueing model.

The use of Petri nets (PN) for performance analysis comes into play in this environment, where they are basically equivalent to extended queueing models from the point of view of the model solution, since both require a translation into the underlying CTMC, but PN offer a language in which synchronization, blocking, and splitting are native in the formalism, and in its semantics. Moreover, PN models benefit from the availability of a wide range of qualitative results derived in a number of years of lively research; these qualitative results allow for example to check for deadlocks, livelocks, etc.

The use of PN-based techniques for performance analysis requires the introduction of temporal specifications within the basic, untimed models, thus generating the modelling paradigms that are usually named Timed Petri Nets (TPN).

Of the numerous TPN proposals that appeared in the literature, we consider only the case in which timing is associated with transitions that keep the atomic firing semantics typical of the "untimed" PN world. We thus neglect the approaches based on timed places, timed arcs, timed tokens, as well as timed transitions that operate in three phases, removing tokens from input places as soon as they become enabled, then letting the transition delay elapse, and finally generating tokens into output places. Although preferences about modelling paradigms are very personal, it may be fair to say that the class of TPN that we consider in this paper is the one that gained widespread acceptance among researchers in the field.

Consistently with the stochastic modelling technique that characterizes queueing approaches, delays associated with TPN transitions will be assumed to be of probabilistic nature.

In summary, we focus on TPN models where a random delay is associated with transitions whose firing is assumed to be atomic, i.e., tokens are removed from input places and put into output places with a single indivisible operation, after the transition firing delay has elapsed. The specification of the firing delay of timed transitions is of probabilistic nature, so that either the probability density function (pdf) or the cumulative distribution function (Cdf) of the delay

associated with a transition needs to be specified. Such functions may be general, or even degenerate, thus allowing the definition of constant (possibly null) delays. We refer to this type of timed Petri nets as Generally Distributed Timed Transitions Stochastic Petri Nets (GDTT_SPN).

The class of TPN that we consider is however too wide to allow a simple solution of any GDTT_SPN model; for this reason we shall pay special attention to two special subclasses of GDTT_SPN, that have the nice property of permitting a reasonably simple evaluation of performance metrics:

- Stochastic Petri Nets (SPN), where all transition firing delays are non-null and have negative exponential pdf,
- generalized SPN (GSPN), where immediate (null-delay) transitions are freely mixed with timed transitions associated with exponentially distributed non-null random firing delays.

When a GDTT_SPN model of a system has been developed, this is normally translated into its underlying stochastic process, that is analyzed either in steady-state or in transient conditions. The computation of the corresponding probabilities can be translated into performance metrics that have a net-level semantics; for example, a net-level result may be the distribution of the number of tokens into a particular place at steady-state, or the average number of times a transition fires during a specified time interval. The mapping between the system features and their description within the GDTT_SPN model allows the translation of such net-level performance parameters into the system-level performance metrics of interest.

3 Generally Distributed Timed Transitions Stochastic Petri Nets

For the translation of a GDTT_SPN model into its underlying stochastic process, it is necessary to associate with the model an *execution policy*, comprising two specifications: a rule to choose the next transition to fire in any marking (the *firing policy*), and a criterion to account for the past history of the model whenever a transition fires (the *memory policy*)[2].

As regards the firing policy, two alternatives are basically possible: either use the delays associated with transitions to decide which one will fire next, or add a specific metrics for this purpose. The GDTT_SPN formalisms that we consider in this paper adopt the first option, that corresponds to the *race policy*: the transition with the minimum remaining delay is the one that fires first.

As regards the memory policy, again two basic alternatives are possible at every change of marking:

[2] In (untimed) Petri nets, the next transition to fire is chosen non deterministically, and there is no need to record the "past history," which is captured by the current state (an intrinsically Markovian assumption!)

continue: the timers[3] associated with transitions hold their present values and will continue being decremented later on;

restart: the timers associated with transitions are restarted, i.e., their present values are discarded, and new values will be generated when needed.

The memory policy is implemented whenever a transition fires. The memory policy thus affects transitions that fire as well as transitions that lose their enabling due to the change of marking, and transitions that keep their enabling in the new marking. The memory of transitions that fire is irrelevant, since in this case a new delay instance must always be generated. The memory of transitions that do not fire is often assumed to be of the following types:

Resampling - The timer of the transition is reset to a new value at any change of marking. The new value is sampled from the pdf of the delay associated with the transition.

Enabling memory - If in the new marking the transition is still enabled, the value of the timer is kept; it is instead reset to a new value if the transition is not enabled.

Age memory - The timer value is kept, even if the transition is not enabled in the new marking.

Table 3 summarizes the possible memory combinations, depending on the transition enabling in the new marking.

	transition remains enabled	transition loses enabling
Resampling	restart	restart
Enabling memory	continue	restart
Age memory	continue	continue

Table 1. Summary of the memory mechanisms.

A GDTT_SPN is a seven-tuple

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

where (P, T, I, O, H, M_0) is the underlying PN system, which as usual comprises

- a set of places $P = (p_1, p_2, \ldots, p_m)$,
- a set of transitions $T = (t_1, t_2, \ldots, t_n)$,
- the input, output, and inhibitor functions $I, O, H : T \to N$,
- an initial marking $M_0 = (m_{01}, m_{02}, \ldots, m_{0m})$,

[3] We can describe the evolution of a GDTT_SPN by associating a timer with each transition: timers are decremented at constant speed while transitions are enabled, and when a timer runs down to zero the corresponding transition fires.

to which it is necessary to add

- a distribution function $\mathcal{W} : T \rightarrow \{\text{pdf}\}$, that assigns to each transition a random variable with a specified pdf,
- an execution policy function $\mathcal{E} : T \rightarrow \{\text{resampling, enabling, age}\}$ that assigns an execution policy to each transition.

The stochastic process that corresponds to the evolution of the GDTT_SPN over its state space (or reachability set) is called the *marking process*.

It should be noted that if $\mathcal{W}$ identifies only continuous functions, then the probability that two transitions are scheduled to fire at exactly the same instant is zero. In such case, a GDTT_SPN model evolves by firing transitions *one by one*.

A GDTT_SPN model can correspond to quite complex stochastic processes, whose definition, not to mention their solution, is not at all a trivial task. Researchers and practitioners have defined subclasses of GDTT_SPN corresponding to simpler classes of stochastic processes, that will be discussed in the sequel of this paper.

Stochastic Petri Nets (SPN)

When all the firing delays associated with transitions are exponentially distributed random variables, GDTT_SPN are called Stochastic Petri Nets (SPN).

The marking process generated by a SPN is a CTMC, with state space isomorphic to the reachability set.

This type of GDTT_SPN models is the most popular in the literature [1, 54, 80, 81, 83, 84], and a number of software tools are available for them [18, 36, 41, 46, 56, 73, 76].

We keep, in this context, the name that was initially assigned to this class of GDTT_SPN, although a more appropriate (less ambiguous) name could be Exponential Petri Nets. We discuss SPN at length in section 4.

Generalized Stochastic Petri Nets (GSPN)

Generalized SPN (GSPN) were originally proposed in [6], with the aim of allowing the simple modelling of complex state changes induced by the firing of a transition, as well as the representation within one model of activities that consume a significant amount of time and activities that require a negligible amount of time. GSPN models comprise therefore 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.

We discuss GSPN at length in section 5.

Semi-Markov SPN

When all transitions in a GDTT_SPN are assigned a resampling policy, the marking process becomes a semi-Markov process, independently of the adopted $\mathcal{W}$ function.

This case was studied in [19, 84], but is of little interest in applications, since it is difficult to find systems where the firing of any transition of the corresponding GDTT_SPN has the effect of forcing the reset of the timers associated with all other transitions, even of those that are concurrently enabled.

A more interesting semi-Markov SPN model was defined in [52]. In this case, transitions are partitioned into three classes: exclusive, competitive and concurrent. Provided that the firing delays associated with all concurrent transitions are exponentially distributed, and that non-exponential competitive transitions are resampled whenever they become enabled, the associated marking process becomes a semi-Markov process.

Phase Type SPN (PHSPN)

A PHSPN is a GDTT_SPN in which:

- the function $\mathcal{W}$ associates with transitions PH (phase type) distributions [85] with a single initial stage and a single final stage,
- any timed transition is assigned a memory policy among the three defined alternatives: resampling, enabling or age memory.

The distinguishing feature of PHSPN is that it is possible to design a completely automated tool for their solution. The non-Markovian process generated by a PHSPN over the reachability set $\mathcal{R}(M_0)$ is converted into a CTMC defined over an expanded state space. The measures pertinent to the original process are defined at the PN level and can be evaluated by solving the expanded CTMC.

A program package that can solve this type of nets is ESP [48].

Deterministic SPN (DSPN)

Deterministic and Stochastic PN were defined in [10], with the aim of providing a technique for considering GDTT_SPN models in which not all transition firing delays are forced to be exponentially distributed.

In [10] only the steady-state solution of DSPN was studied. An improved algorithm for the evaluation of the steady-state probabilities was successively presented in [75, 76], and some structural extensions were proposed in [40].

A DSPN [10] is a GDTT_SPN in which:

- the function $\mathcal{W}(t)$ associates either exponentially distributed or deterministic firing times with timed transitions,
- at most one transition with deterministic delay is enabled in each marking,
- the execution policy for all deterministic transitions is *enabling memory*.

As a consequence of this definition, during the firing of a transition with a deterministic delay, the marking process can undergo state changes only due to exponentially timed transitions, thus describing a CTMC called the subordinated process.

The steady-state solution technique originally proposed in [10] is based on the evaluation of the subordinated CTMC at the firing time of the deterministic transition.

Tools currently supporting the steady-state analysis of DSPN models are DSPNexpress [76], UltraSAN [46] and TimeNET [56], this last one also supports transient analysis.

Choi et al. [37] proved that the marking process associated with a DSPN is a Markov regenerative process (MRP), for which steady-state and transient solution equations are available [42].

Markov Regenerative SPN (MRSPN)

A natural extension of DSPN was proposed in [38], where the deterministic distribution is substituted by a general one; this model is referred to by the authors as MRSPN, since the underlying model is again an MRP.

Similar restrictions as for DSPN apply also to MRSPN, that is to say:

- $\mathcal{W}(t)$ is either exponential or general,
- at most one transition with general pdf can be enabled in each marking,
- the only allowed execution policy for generally distributed transitions is *enabling memory*.

During the firing delay of a transition with general distribution, only exponential transitions can concurrently fire: the process subordinated to a generally distributed transition is thus a CTMC.

With the aim of extending the modelling power of MRSPN by including generally distributed transitions with age memory policy, Bobbio and Telek [20, 21] investigated a class of models characterized by the fact that the subordinated process between two consecutive regeneration epochs is a Semi Markov Reward Process [90].

GDTT_SPN Taxonomy

The relationships among the various subclasses of GDTT_SPN are depicted in Figure 7.

It should be noted that some of the definitions of the GDTT_SPN subclasses do not provide a structural characterization of the class. Indeed, the net structure is sufficient to decide whether a GDTT_SPN belongs to the SPN, PHSPN or GSPN classes (it is sufficient to check the $\mathcal{W}$ function). Instead, it may not be possible to determine from the GDTT_SPN structure, without building the reachability set, whether a GDTT_SPN is a DSPN or a MRSPN, since it is necessary to determine the set of effectively conflicting transitions. As a consequence, from the structure of the GDTT_SPN it is only possible to obtain

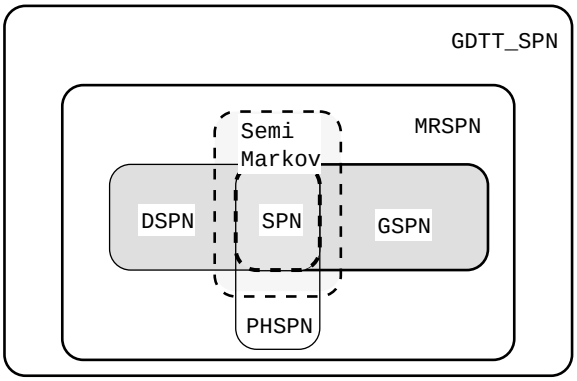

Fig. 7. Subclasses of GDTT_SPN.

sufficient conditions for the GDTT_SPN to be a DSPN or a MRSPN. Actually, even the state space construction may not be sufficient in the more general case of MRSPN, since it may be the case that additional informations, computable from the reachability graph, may be needed.

Another possible classification of GDTT_SPN could be based on the relationships between the behaviour of the timed models and the behaviour of the underlying P/T net. Indeed, in those cases in which the reachability graph of the timed net is identical to the one of the P/T net, the standard qualitative analysis techniques can be applied; in particular, all properties proved on the P/T net structure and/or on the reachability graph of the P/T net are valid also for the timed model, typically P- and T-invariants, boundedness, liveness, etc. This is the case for SPN.

In the case in which general firing delay pdf are allowed, 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, or not all interleavings of concurrent transitions can actually take place in any timed sequence.

A sufficient condition for the two reachability graphs to be identical is that $\mathcal{W}$ is taken from the set of pdf that have unlimited support [2, 3]. This condition in general is not met by GDTT_SPN with deterministic distributions, or uniform distributions over a finite interval.

In the case of GSPN, instead, the reachability set is identical to the one of the underlying PN system *with priorities*. Not too many qualitative analysis results are available for this class of nets, but it can be observed that the presence of inhibitor arcs and priorities only restricts the reachability set with respect to the one of the basic underlying net.

The restriction of the reachability set guarantees that all P-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. For what concerns T-invariants, the presence

of priorities and inhibitor arcs may make non fireable a fireable invariant.

4 Stochastic Petri nets

SPN models were proposed by researchers active in the applied stochastic modelling 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, while for the first two, it was chosen to introduce time in such a way as not to alter the untimed behaviour of the system.

In order to obtain an equivalence between a PN and a CTMC, it is necessary that the determination of the next reachable states depends only on the current one (true by definition in Petri nets), and that sojourn times in markings are random variables with negative exponential pdf.

This idea formed the basis of the doctoral dissertations of S.Natkin [84] at the Conservatoire National des Arts et Métiers in Paris, France, and of M.K.Molloy [80] at the Computer Science Department of 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 PN transitions was already present in the doctoral dissertation of F.J.Symons within the definition of Numerical Petri nets [98, 97].

4.1 The Basic Model

An SPN is a GDTT_SPN in which the $\mathcal{W}$ function assigns to each transition an exponential pdf. Since the exponential distribution is fully characterized by its mean value (or by its inverse, the *rate*), and its memoryless characteristics makes inessential, as we shall explain later, the definition of an execution policy for transitions, then the definition of an SPN is simpler than the one for GDTT_SPN, namely:

$$SPN = (P, T, I, O, H, M_0, W)$$

where:

- (P, T, I, O, H, M_0) is the underlying PN system[4], as for GDTT_SPN,
- $W : T \to \Re$ is a *weight function*; $w(t)$ is the rate of the exponential distribution associated with transition t.

$w(t)$ is also called the *firing rate* of transition t.

According to some definitions of SPN, the weight function can depend on the current marking: in this case we write $w(t, M)$ to express the weight of transition

[4] Inhibitor arcs were not present in the first definitions of SPN, but they have become very popular in the field, so that we prefer to include them in the basic definition.

t in marking M. Common types of marking dependency are the ones borrowed from queueing theory, namely *single, multiple and infinite server*, but while in queueing systems the dependence of the service rate normally is on the number of customers in the queue, in SPN the dependence is on the transition enabling degree [8].

The average firing delay of transition t in marking M is therefore $[w(t, M)]^{-1}$.

Under the race policy that we are considering, the transition with the minimum delay is the one that fires first. The firing delay computed for each transition at the ingress in a new marking is either a new value sampled from the exponential distribution associated with the transition (for transitions that are "restarted"), or the residual firing time of the timer (for transitions that "continue"). However, due to the memoryless property of the exponential distribution, the distribution of the whole firing time is identical to the one of the residual firing time, hence the race is always among exponentially distributed variables, thus making not necessary the definition of the execution policy in SPN models. Since the minimum of two random variables with negative exponential pdf and parameters μ_1 and μ_2 is a random variable which still is exponentially distributed, with parameter $(\mu_1 + \mu_2)$, the sojourn time in marking M is a random variable with negative exponential pdf, with mean

$$\left[\sum_{t \in E(M)} w(t, M) \right]^{-1}$$

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

The fact that all firing delays have exponential pdf permits a simple expression to be written for the probability that a given transition, say t, is the one for which the minimum delay has been sampled, and hence determines the change of marking by firing:

$$P\{t|M\} = \frac{w(t, M)}{\sum_{t' \in E(M)} w(t', M)} \qquad t \in E(M)$$

The reachability set of an SPN is identical to the one of the underlying untimed PN (with interleaving semantics) due to the unlimited support and to the continuity of the exponential distribution.

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. Indeed, if in a marking M two transitions t_1 and t_2 are enabled, either concurrently or in conflict, with $M[t_1\rangle M_1$ and $M[t_2\rangle M_2$, the sojourn time in M is a random variable distributed as the minimum of the two exponential distributions of the random variables associated with t_1 and t_2, that is to say, it is exponentially distributed with rate $w(t_1) + w(t_2)$. Since the probability of firing t_1 is $\frac{w(t_1)}{w(t_1)+w(t_2)}$, then the rate at which the system moves from M to M_1 is $[w(t_1) + w(t_2)]\frac{w(t_1)}{w(t_1)+w(t_2)} = w(t_1)$. A similar computation leads to $w(t_2)$ for t_2.

The steady-state solution of the model is then obtained by solving the system of linear equations

$$\boldsymbol{\pi Q} = \mathbf{0}$$
$$\sum_{M \in RS} \pi[M] = 1$$

$\boldsymbol{\pi}$ is the equilibrium pmf over the reachable markings, and we write $\pi[M]$ for the steady-state probability of a given marking M.

The transient solution of the model is instead obtained solving the set of differential equations

$$\frac{d\boldsymbol{\pi}(t)}{dt} = \boldsymbol{Q}\boldsymbol{\pi}(t)$$

where $\pi(t)[M]$ is the probability of the system being in state M at time t.

We may now go back to the lamp example discussed in Section 2.3. The SPN model describing the system considered in the example is depicted in Figure 8. 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 state machine [91].

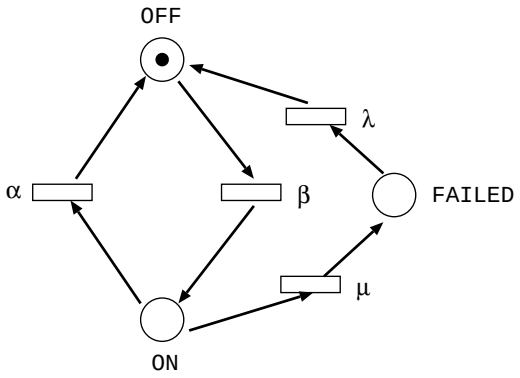

Fig. 8. SPN model of the lamp example.

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

Consider as a second example the SPN depicted in Figure 9. This is the SPN representation of the M/M/1 queue described in Section 2.4. 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.

As an additional example, consider a system that exhibits choice, concurrency, as well as splitting and joining of customers, whose model is given in Figure 10. A process executes a choice between a fork activity (modelled by transition $T1$), and a computation (modelled by transition $T2$). The computation is followed by an additional activity modelled by transition $T4$. The execution of the two forked processes is modelled by the two independent transitions $T5$

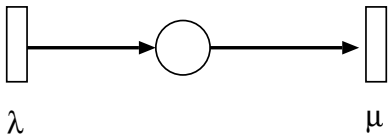

Fig. 9. SPN representation of the M/M/1 queue.

and $T6$, while transition $T3$ represents the join, and indeed its enabling requires that both subprocesses have finished their execution. Transition $T7$ represents a subprocess common to the two branches, that takes the process back to its initial state.

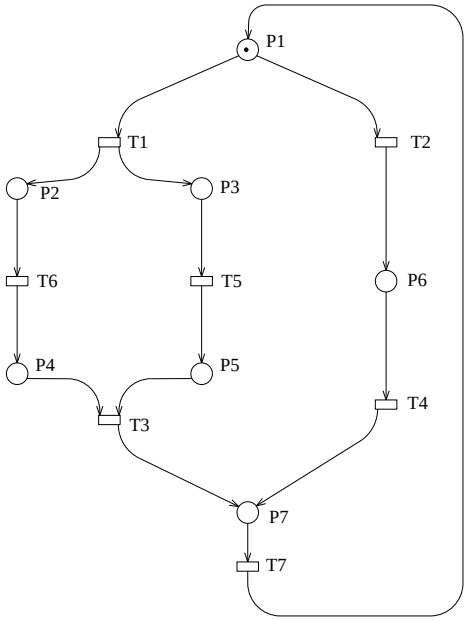

Fig. 10. SPN model of a fork and join system.

The state space of this SPN comprises seven states, shown in Table 2, while the infinitesimal generator is given in Table 3, and its state transition diagram is depicted in Figure 11.

It is important to remark that, in case of conflict, the rate of conflicting transitions accounts for both the *duration* of the activity and the *probability* of that activity. For example, if in the system we are modelling the average time

M_1	P1
M_2	P2 + P3
M_3	P6
M_4	P2 + P5
M_5	P3 + P4
M_6	P7
M_7	P4 + P5

Table 2. State space of the Fork and Join SPN model.

	M_1	M_2	M_3	M_4	M_5	M_6	M_7
M_1		$w(T1)$	$w(T2)$				
M_2				$w(T5)$	$w(T6)$		
M_3						$w(T4)$	
M_4							$w(T6)$
M_5							$w(T5)$
M_6	$w(T7)$						
M_7						$w(T3)$	

Table 3. Representation of the infinitesimal generator of the Fork and Join SPN model.

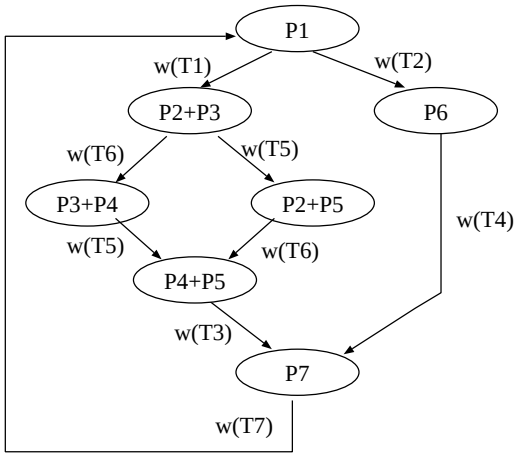

Fig. 11. State transition rate diagram of the Fork and Join SPN model.

to decide whether to perform a fork and join ($T1$) or a simple computation ($T2$) is equal to 0.0001 time units, and the probability of a fork and join is 99% against the 1% of the normal computation, we obtain a rate for $T1$ and $T2$ of: $w(T1) = 9,900$ and $w(T2) = 100$: in SPN there is no way to split the duration from the probabilistic choice.

4.2 Performance Indices

Several kinds of aggregate results are easily obtained from the steady-state or transient distributions over reachable markings. In this section we quote some of the most commonly and easily computed aggregate performance parameters [9].

- The *expected fraction time spent in a subset of markings* $\mathcal{M}$, in the interval $[0 \dots t]$, can be computed as

$$\psi\{\mathcal{M}, t\} = \frac{1}{t} \sum_{M \in \mathcal{M}} \int_0^t \pi(z)[M] dz$$

  where $\pi(t)[M]$ is the probability of being in state M at time t. From the theory of Markov chains, it is well known that as t approaches infinity, $\psi\{\mathcal{M}, t\}$ becomes equal to the steady-state probability

$$\pi(\mathcal{M}) = \sum_{M \in \mathcal{M}} \pi[M]$$

- 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 $\mathcal{A}$ defined through a condition that holds true for the markings $M \in \mathcal{M}$ is obtained as:

$$P\{\mathcal{A}\} = \sum_{M \in \mathcal{M}} \pi[M]$$

  while the probability of event A at time t is

$$P\{\mathcal{A}, t\} = \sum_{M \in \mathcal{M}} \pi(t)[M]$$

  This formula can also be used to compute the probability that a given condition is satisfied *for the first time* at time t, provided that the SPN is such that all and only the states satisfying the conditions that define event $\mathcal{A}$ are deadlock states (if this is not the case, the SPN should be modified accordingly). The same result can be obtained by changing the marking process rather than the SPN: it is only necessary to make the states in $\mathcal{M}$ absorbing.

– The *pmf of the number of tokens at steady-state in a place*, say p, can be obtained by computing the individual probabilities in the pmf as probabilities of the event "place p contains k tokens". The *pmf of the number of tokens in a place, at time* t, can be obtained similarly, using the event "place p contains k tokens at time t"
– The *average number of tokens in a place (at time t)* can be computed from the pmf of tokens in that place (at time t).
– The *expected number of firings of transition* t_k in the interval $[0, t]$, $f_{t_k}(t)$, can be computed integrating over the given interval the firing rate of transition t_k, expressed as the sum over all states M enabling t_k of the firing rate of t_k in M, weighted by the probability of being in M at time t

$$f_{t_k}(t) = \int_0^t \sum_{M:t_k \in E(M)} w(t_k, M)\ \pi(z)[M]\ dz$$

where $E(M)$ is the set of transitions enabled in M, and $w(t_k, M)$ is the firing rate of t_k in M. The sum and the integral can be exchanged to get:

$$f_{t_k}(t) = \sum_{M:t_k \in E(M)} \int_0^t w(t_k, M)\ \pi(z)[M]\ dz$$

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

$$f_{t_k} = \sum_{M:t_k \in E(M)} w(t_k, M)\ \pi[M]$$

All the above performance indices can be defined using a unified approach based on rewards: a reward is a function $r(t) : M \to \Re$ for transient analysis and $r : M \to \Re$ for steady state. An *average reward* can be computed as the weighted sum

$$\begin{aligned} R(t) &= \sum_{M \in RS} r(M)\pi(t)[M] \\ R &= \sum_{M \in RS} r(M)\pi[M] \end{aligned}$$

for transient and steady state analysis, respectively.

For example, the mean number of tokens in a place p can be computed by defining $r(M) = M(p)$, and the throughput of a transition can be computed by the reward function

$$r(M) = \begin{cases} w(t, M) & \text{if } t \in E(M) \\ 0 & \text{otherwise} \end{cases}$$

From the above indices it is possible to compute the *average delay of a token* in traversing a subnet in steady-state conditions by using Little's formula [69, 77]

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

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.

5 Generalized SPN

The key factor that limits the applicability of SPN models is the complexity of their analysis. This is due to many factors. 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 to ease the description of complex marking changes, so that no timing can be reasonably associated with them.

Generalized SPN (GSPN) were originally proposed in order to tackle these problems [6]. The definition was later improved in order to better exploit the structural properties of the modelling tool [4, 5]. The book [8] presents in detail the GSPN formalisms together with a number of application examples.

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

A GSPN is thus an eight-tuple

$$GSPN = (P, T, \Pi, I, O, H, M_0, W)$$

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

– the set P of places,
– the set T of transitions,
– the input, output and inhibitor functions $I, O, H: T \to N$,
– the initial marking M_0.

Additionally, the GSPN definition comprises the priority function $\Pi : \Pi : T \to N$ which associates lowest priority (0) with timed transitions and higher priorities (≥ 1) with immediate transitions:

$$\Pi(t) = \begin{cases} 0 & \text{if } t \text{ is timed} \\ \geq 1 & \text{if } t \text{ is immediate} \end{cases}$$

Finally, the last item of the GSPN definition is the function $W : T \to \Re$, that associates a real value with transitions. $w(t)$ is:

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

In the graphical representation of GSPN, immediate transitions are drawn as segments, and exponential transitions as white rectangular boxes.

The untimed underlying model of a GSPN is a P/T net with inhibitor arcs and global priorities: the addition of priorities to a P/T system can reduce the number of reachable states, and it may destroy eventuality properties like liveness and home states, while all safety properties are maintained.

The stochastic 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*.

In the case of a tangible marking, the timers of the enabled timed transitions either resume their decrement, or are re-initialized 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

$$P\{t\} = \frac{w(t)}{\sum_{t' \in E(M)} w(t')}$$

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.

Observe that in the above formula the probabilities are normalized over all enabled transitions, so that the normalization takes place also among non-conflicting transitions. From a modeller point of view it may be difficult to specify transitions weights if they are then normalized over the whole net. However, it was proved in [3] that if no confusion is present in the net (that is to say if no interplay exists between conflict and concurrency), it is possible to determine at a structural level the sets of possibly conflicting transitions, called "extended conflict sets (ECS)," and the normalization of weights can be done only among transitions that belong to the same ECS.

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 [16].

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 labelled 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 [12].

An example of construction of the tangible reachability graph is presented in Figure 12, that depicts a very simple GSPN in which a conflict exists among immediate transitions. Its reachability graph is shown in the upper right portion. Dotted lines for state $p2$ indicate that the state is vanishing: indeed, when transition $T1$ fires the system enters marking $p2$ in which two immediate transitions are enabled, and the marking changes in zero time to either $p3$ or $p4$, with probability $\frac{\alpha}{\alpha+\beta}$ and $\frac{\beta}{\alpha+\beta}$ respectively. The tangible reachability graph in the lower right portion is obtained by eliminating the vanishing marking $p2$. The rate at which the system moves from $p1$ to $p3$ ($p4$) is obtained by multiplying the rate μ of the state transition from $p1$ to $p2$ with the probability of going from state $p2$ to $p3$ ($p4$).

Figure 13 shows a GSPN model of a fork and join behaviour similar to the one described with the SPN in Figure 10. The GSPN model is obtained from the SPN model by making immediate those transitions that describe only a logical behaviour, or an activity of negligible duration; in particular the choice is modelled as a conflict between two immediate transitions and the join is also considered as an immediate action. Markings $P1$ and $P2+P3$ are now vanishing.

The application of SPN and GSPN modelling techniques has been very productive in several areas. The factor that has however limited their acceptance as a modelling tool lies in the (graphical and computational) *complexity* of the models of realistic systems. Different efforts for the problem solution are summarized in Section 6.

It must also be stressed that the use of SPN and GSPN heavily relies on the availability of adequate software *tools*, without which the model construction

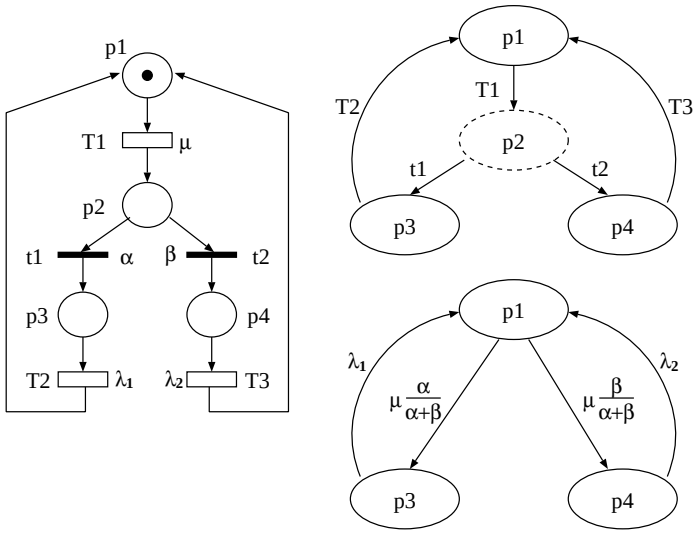

Fig. 12. Tangible reachability graph construction.

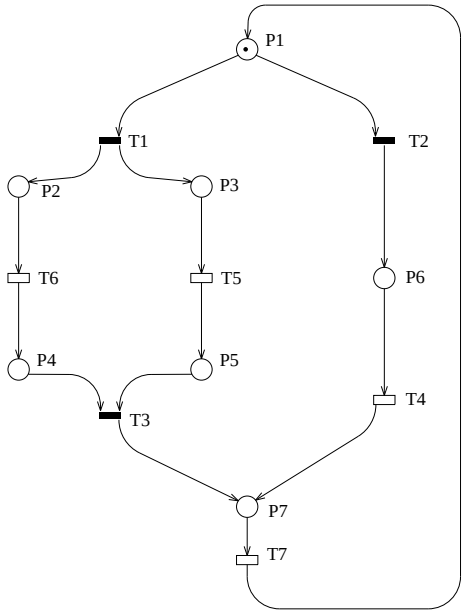

Fig. 13. GSPN representation of a fork and join system.

and solution is possible only for the smallest toy examples[5].

As a complete example of the use of GSPN both from a modelling and from an analysis point of view, we consider a multiserver multiqueue system, also known in the literature as a *multiserver cyclic polling system.*

5.1 GSPN Model of a Multiserver Cyclic Polling System

A single-server cyclic polling system comprises a set of waiting lines that receive arrivals from the external world, and one server that cyclically visits the queues, providing service to customers, if any is waiting. The GSPN description of such a polling system is provided in Figure 14.

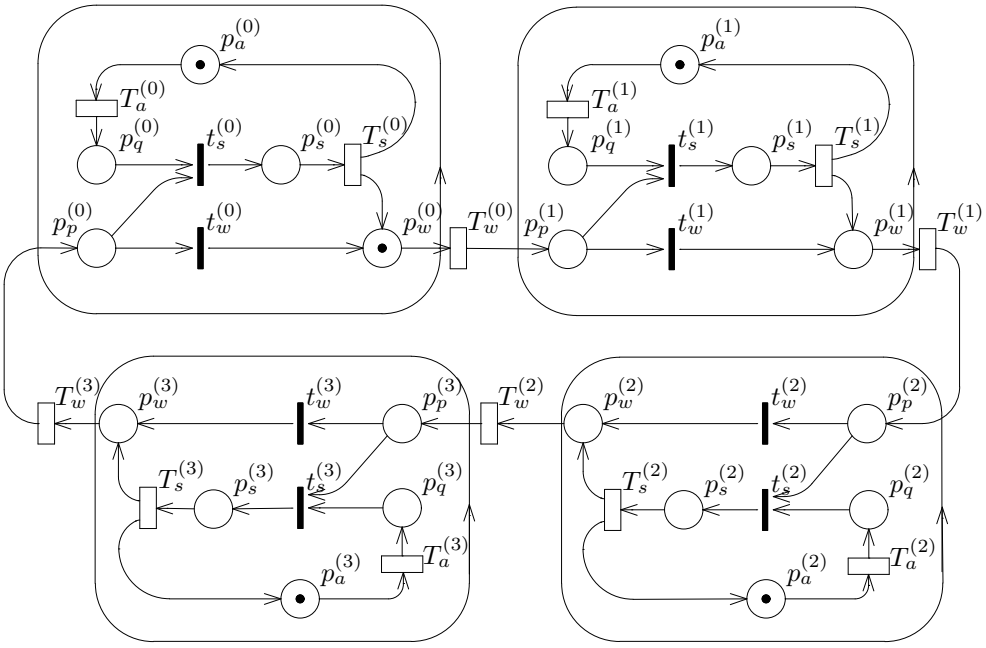

Fig. 14. GSPN representation of a single-server cyclic polling system.

The GSPN model in Figure 14 comprises four replicas of the subnet that describes the internal organization of each individual queue (enclosed within ovals), interconnected by four replicas of the subnet that describes the movement of the server from a queue to the next one.

[5] Readers interested in untimed and timed Petri nets tools can find many useful informations at the Petri net Web site "www.daimi.aau.dk/PetriNets"

Transition $T_a^{(q)}$ models the customer arrival process at queue[6] q ($q = 0, 1, 2, 3$). Customers waiting for a server are queued in place $p_q^{(q)}$, while a token in place $p_p^{(q)}$ represents the server when polling queue q. The two immediate transitions $t_s^{(q)}$, and $t_w^{(q)}$ have priorities 2, and 1, respectively ($\Pi(t_s^{(q)}) = 2, \Pi(t_w^{(q)}) = 1$). Transition $t_s^{(q)}$ fires if the server finds a waiting customer when it polls the queue, so that service can be provided; if $t_s^{(q)}$ cannot fire when the queue is polled by a server, i.e., if the server finds no waiting customers, then $t_w^{(q)}$ fires, and the server walks to the next queue.

One token in place $p_s^{(q)}$ represents a customer of queue q being served, as well as the server when providing service at queue q. $T_s^{(q)}$ is timed with a delay equal to the customer service time.

The server moves to place $p_w^{(q)}$ at the end of the visit at queue q (after the service completion represented by the firing of $T_s^{(q)}$, if a waiting customer was found; after the firing of $t_w^{(q)}$ if no waiting customer was found). From $p_w^{(q)}$ the server walks to the next queue. Transition $T_w^{(q)}$ models the server walk time from queue q to the next queue in the cyclic schedule.

The GSPN model in Figure 14 represents the case of a single customer in each queue ($M(p_a^{(q)}) = 1$), and of a single server initially placed at the exit of queue 0 ($M(p_w^{(0)}) = 1$). GSPN models for K customers in each queue, and S circulating servers can be obtained by assigning an initial marking with $M(p_a^{(q)}) = K$, and $M(p_w^{(0)}) = S$

The characteristics of the timed and immediate transitions in the GSPN model in Figure 14 are summarized in Tables 4 and 5, respectively: note that transitions that represent activities of servers ($T_s^{(q)}$ and $T_w^{(q)}$) are of infinite-server type, since we want each server to be able to either walk or serve in parallel with any other server; vice-versa, the transitions that represent an arrival of a customer to a queue ($T_a^{(q)}$) are single-server, to emulate a finite Poisson source of customers.

transition	rate	semantics
$T_a^{(q)}$	$\lambda^{(q)}$	single-server
$T_s^{(q)}$	$\mu^{(q)}$	infinite-server
$T_w^{(q)}$	$\omega^{(q)}$	infinite-server

Table 4. Characteristics of the timed transitions in the GSPN model of a cyclic single-server polling system ($q = 0, 1, 2, 3$).

[6] The superscript (q) indicates that the place (or transition) belongs to the model of queue q.

transition	weight	priority
$t_s^{(q)}$	1	2
$t_w^{(q)}$	1	1

Table 5. Characteristics of the immediate transitions in the GSPN model of a cyclic single-server polling system ($q = 0, 1, 2, 3$).

Typical aggregate performance figures of interest in the analysis of polling systems are the average customer delay, the probability of customers having to wait for a server, as well as the throughputs of the whole system, and of individual queues.

The structural analysis of the GSPN model detects five P-semiflows that cover all the places of the GSPN. Four of them cover the triplets of places $p_a^{(q)}$, $p_q^{(q)}$, $p_s^{(q)}$, with $q = 0, 1, 2, 3$. The resulting P-invariants are $M(p_a^{(q)}) + M(p_q^{(q)}) + M(p_s^{(q)}) = N^{(q)}$. This guarantees that places $p_a^{(q)}$, $p_q^{(q)}$, $p_s^{(q)}$, with $q = 0, 1, 2, 3$ are bounded, as was expected, since the number of customers either waiting or being served at any queue cannot exceed the number of customer in the closed arrival generation process. The fifth P-semiflow covers places $p_p^{(q)}$, $p_s^{(q)}$, $p_w^{(q)}$, with $q = 0, 1, 2, 3$. The token count of the resulting P-invariant is S, since the P-invariant refers to the conservation of the number of servers in the system. As a result, the GSPN model is bounded, and thus the number of reachable states is finite.

The GSPN model in Figure 14 is covered by 16 T-semiflows; each of them represents a possible path of a server in the cycle of queues. The simplest cycle (no customer is available at any queue) corresponds to the T-semiflow $t_w^{(q)}$, $T_w^{(q)}$, with $q = 0, 1, 2, 3$. The 16 T-semiflows correspond to the possible combinations of the conditions "customer ready for service" and "customer not ready for service" in the 4 queues. For example, the T-semiflow that represents a cycle in which the server provides service only at queue 0, is: $T_a^{(0)}$, $t_s^{(0)}$, $T_s^{(0)}$, $T_w^{(0)}$, and $t_w^{(q)}$, $T_w^{(q)}$, with $q = 1, 2, 3$.

The covering of the transitions of the net is a necessary condition for the system to be live and reversible, but not sufficient. A reachability graph analysis shows that the GSPN model indeed is live and reversible.

The size of the reachability graph depends on the number of queues, on the number of servers, and on the number K of customers at each queue. The second column of Table 6 shows the size of the reachability graph for 4 queues and 2 servers, for different values of K. Bigger values for the number of queues can be considered, although the reachability set grows quite fast (the GSPN model with $K = 1, 2$ servers, and 8 queues produces $287,328$ tangible markings, the file that stores the reachability graph occupies about 14 megabytes, and the reachability set takes about 2.5 megabytes).

The third and fourth columns of Table 6 show the throughputs of transitions $T_a^{(q)}$ and $T_w^{(q)}$, respectively, when the weights of transitions $T_a^{(q)}$ and $T_s^{(q)}$ are taken equal to 1.0 s^{-1}, and the weight of $T_w^{(q)}$ is set to 5.0 s^{-1} (which implies a delay among queues of 0.2 s), independently of the queue index q. Due to the symmetry of the system, all transitions of equal name, but different queue index, have the same throughput. The last two columns report the probability that at any queue the source of arrival is empty ($P\{p_a^{(q)} = 0\}$), and that no customer is waiting ($P\{p_q^{(q)} = 0\}$).

The analysis was performed only for values of K up to 6, because we can observe that the system reaches the limit maximum throughput for transitions that describe servers; indeed, when the system is fully loaded (a customer is always waiting for service) a server will provide service at each queue, which implies a mean cycle time per server, equal to 4.8 s (4.0 s for service and 0.8 s to walk), and, considering that servers are independent in their services and walks, due to the infinite-server semantics, then the maximum throughput that can be observed for server transitions is $1/4.8 * 2 = 0.41666$ s^{-1}: this value is reached only when the number of customers is so high that the probability that a server finds no customer when polling the queue is negligible. As it can be observed from the last column, the system is very close to this limit condition already for $K = 4$, since the probability that no customer is waiting in the queue is close to zero.

K	$\|RS\|$	$f_{T_a^{(q)}}$	$f_{T_w^{(q)}}$	$P\{p_a^{(q)} = 0\}$	$P\{p_q^{(q)} = 0\}$
1	312	0.34500	0.77497	0.65499	0.69001
2	1,998	0.40964	0.45176	0.59035	0.22832
3	7,008	0.41563	0.42185	0.58437	0.03081
4	18,150	0.41649	0.42185	0.58371	0.00594
5	39,096	0.41651	0.41682	0.58349	0.00107
6	74,382	0.41661	0.41669	0.58334	0.00107

Table 6. Reachability set size, throughputs and probabilities for the polling model of Figure 14.

As an example of transient analysis for this system, consider the event E defined as "all servers are simultaneously providing service at queue 0 for the first time ". We can then compute $P\{E, t\}$, using the trick of making the first encountered state that satisfies E an absorbing state. At the net level this can be obtained by adding an immediate transition with an input arc of weight equal to S from place $p_s^{(0)}$, and an output arc of weight 1 to a new place p_{abs}. The probability of event E at time t can then be computed as the probability of a token in place p_{abs} at time t. Please observe that, for this probability to be

non-zero, we need $K \geq S$. Table 7 shows this probability for a polling system with 4 stations, with the same rate parameters as used for steady-state analysis, and $S = K = 2$.

t	$P\{E, t\}$
1	0.0288
2	0.1124
3	0.1924
4	0.2674
5	0.3376
10	0.6048
50	0.9942

Table 7. $P\{E, t\}$ for the polling model of Figure 14.

6 Current Research on GDTT_SPN

Several groups of researchers are presently active in the field of GDTT_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 is devoted to the application of GDTT_SPN to performance and reliability studies of a very diverse gamut of systems, including distributed computing systems architectures, distributed software, object-oriented systems, data base, real-time systems, communication protocols, VLSI, manufacturing systems, inventory and logistics.

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 [15]).

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 monstrously 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 [57, 58] that the modelling power of GDTT_SPN in the simulation framework

is remarkable, since they are equivalent to generalized semi-Markov processes (GSMP).

A number of research teams are instead attacking the theoretical problems inherent in the management of largeness in GDTT_SPN models. Some of the main lines of research are concisely mentioned below.

Distributed solution algorithms - Numerical distributed algorithms have been specifically developed for both the generation of the reachability graph in a PN and for the solution of the underlying CTMC [11, 39, 33, 79].

Structured representation - An approach to increase the size of analyzable CTMC is to represent the generator matrix in a compact form as a combination of smaller component matrices, and to exploit this representation in the solution algorithm. A compositional technique based on Kronecker operators proposed in [88] was initially transferred to the SPN framework in [49, 50]. Efficient techniques were developed for reachability analysis [66, 67] and for numerical analysis [26, 27, 28, 32, 68], exploiting the structure of the generator matrix. In order to use the structured analysis technique, the PN model normally has to be described by means of submodels interacting via synchronizing transitions. However, structured schemes for asynchronously interacting submodels have also been presented in [25].

SPN and queueing networks - Results obtained in other stochastic modelling fields can be cast into PN. A very interesting line of research is aimed to combining the techniques available for the analysis of queueing networks into the language of PN. In some cases, it is possible to solve subsystems in the form of queueing networks, and to compose the results in a *higher level* GSPN representation specifying the interaction among submodels. The inverse procedure was also followed, first solving independent GSPN submodels that are then connected in a queueing network structure [13, 14]. A package supporting the replacement of GSPN places by queueing systems is presented in [17].

Particular classes of SPN, like those generating queueing models with matrix-geometric structure [85], were also considered, and a tool was built for their analysis [61, 62]

Product form SPN - Several proposals were recently documented to import the product form concept into the PN arena.

In [72], a class of SPN is identified for which a product form equilibrium equation can be written from the knowledge of partial balance equations. The generation of the reachability graph is needed to recognize this class of PNs. An extension of the same line is presented in [74].

Henderson et al. [63] developed a product form criterion based only on the structure of the PN, with no need to generate the reachability graph.

A comparative analysis of these two types of approaches was presented in [51], where the possibility of recognizing whether a PN admits a product form solution using results from the structural analysis was proved for the first time.

A complete characterization of product form SPN is provided in [22], and a necessary and sufficient condition for the existence of a positive solution for the traffic equation can be found in [23]. Specific algorithms for the computation of product form solutions were presented in [44, 94].

Mean value analysis for non-product form SPN was explored in [93].

PN-driven techniques - These techniques deal with the reduction of both memory requirements and time complexity of solution algorithms by using information about the structure of the untimed PN models.

High-Level Stochastic Petri Nets, such as for example Stochastic Well-formed Nets (SWN), often exhibit behavioural symmetries that can be exploited to reduce the size of the state space, and of the corresponding CTMC, by grouping states into *equivalence classes*. The desirable properties of a good analysis method based on this idea are the possibility of automatically discovering the symmetries using only the information contained in the model description at the PN level, and the possibility of directly generating the reduced state space (and the lumped Markov chain) with no need to build the complete reachability graph.

A completely automatic method for the construction of a lumped CTMC of a SWN model was presented in [35, 37]. In [59, 60] it was shown that in some cases it is possible to integrate this approach with decomposition methods based on Kronecker Algebra. In [92], the same idea was developed for Stochastic Activity Networks (SAN), and an automatic method was described for the construction of the lumped CTMC starting from the SAN description.

Deterministically Synchronized Sequential Processes (DSSP) are a class of SPN that can be obtained by resorting to simple modular design principles; for this class of models, a well-established theory exists for qualitative analysis [89, 95]. Net-driven techniques, developed for DSSPs, can recognize and extract from the original model a set of simpler auxiliary submodels that can then be analyzed through approximate iterative techniques [31] as well as exact manipulation [32].

Performance Bounds - A complementary approach to the development of efficient solution techniques for the computation of performance measures, is the search for bounds. Bounds require less computational effort with respect to exact analytical solutions, since they are estimated from PN-level equations, and do not require the knowledge of the reachability graph. Moreover, the evaluation of the bounds is, usually, not restricted to the Markovian assumption.

Results in this direction were derived since the beginning of the research on SPN [24, 82].

Subsequently, an extensive amount of work tried to exploit the structure of the PN to obtain efficient computation techniques. The evaluation of bounds for the subclass of marked graphs was presented in [29, 31], and in [30] for PN with a unique consistent firing count vector.

A general approach for the computation of bounds was formulated in [34],

based on operational analysis techniques applied at the PN level, with very weak assumptions on their timing semantics. The bounds can be obtained in polynomial time by solving suitable linear programming problems; they depend only on the mean values of firing times and are insensitive to distributions. In the case of Markovian SPN, an improved solution technique, based on the randomization algorithm, was presented in [78].
A time scale decomposition approach was proposed in [65]. This approach requires that transitions be classified into two classes: fast and slow.

Is summary, it can be observed that research in the GDTT_SPN field has been and still is lively and challenging, possibly quite rewarding for bright young researchers wishing to contribute to the field.

References

1. M. Ajmone Marsan. Stochastic Petri nets: an elementary introduction. In G. Rozenberg, editor, *Advances in Petri Nets 1989, Lecture Notes in Computer Science*. Springer Verlag, 1990.
2. M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. On Petri nets with stochastic timing. In *Proceedings International Workshop on Timed Petri Nets*, pages 80–87, Torino (Italy), 1985. IEEE Computer Society Press no. 674.
3. M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. The effect of execution policies on the semantics and analysis of stochastic Petri nets. *IEEE Transactions on Software Engineering*, SE-15:832–846, 1989.
4. M. Ajmone Marsan, G. Balbo, G. Chiola, and G. Conte. Generalized stochastic Petri nets revisited: random switches and priorities. In *Proceedings International Workshop on Petri Nets and Performance Models - PNPM87*, pages 44–53. IEEE Computer Society, 1987.
5. M. Ajmone Marsan, G. Balbo, G. Chiola, and G. Conte. Generalized stochastic Petri nets: a definition at the net level and its implications. *IEEE Transactions on Software Engineering*, 19:89–107, 1993.
6. M. Ajmone Marsan, G. Balbo, and G. Conte. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 2:93–122, 1984.
7. M. Ajmone Marsan, G. Balbo, and G. Conte. *Performance Models of Multiprocessor Systems*. The MIT Press, 1986.
8. M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. Wiley Series in Parallel Computing, 1995.
9. M. Ajmone Marsan, A. Bobbio, G. Conte, and A. Cumani. Performance analysis of degradable multiprocessor systems using generalized stochastic Petri nets. *IEEE Computer Society Newsletters*, 6, SI-1:47–54, 1984.
10. M. Ajmone Marsan and G. Chiola. On Petri nets with deterministic and exponentially distributed firing times. In *Lecture Notes in Computer Science*, volume 266, pages 132–145. Springer Verlag, 1987.

11. S.C. Allmaier, M. Kowarschik, and G. Horton. State space construction and steady-state solution of GSPNs on a shared-memory multiprocessor. In *7-th International Conference on Petri Nets and Performance Models - PNPM97*, pages 112–121. IEEE Computer Society, 1997.
12. H. H. Ammar and R. W. Liu. Analysis of the generalized stochastic Petri nets by state aggregation. In *Proceedings International Workshop on Timed Petri Nets*, pages 88–95, Torino (Italy), 1985. IEEE Computer Society Press no. 674.
13. G. Balbo, S.C. Bruell, and S. Ghanta. Combining queueing network and generalized stochastic Petri net models for the analysis of some software blocking phenomena. *IEEE Transactions on Software Engineering*, 12:561–576, 1986.
14. G. Balbo, S.C. Bruell, and S. Ghanta. Combining queueing network and generalized stochastic Petri nets for the solution of complex models of system behavior. *IEEE Transactions on Computers*, 37:1251–1268, 1988.
15. G. Balbo, G. Chiola, S.C. Bruell, and P. Chen. An example of modelling and evaluation of a concurrent program using coloured stochastic Petri nets: Lamport's fast mutual exclusion algorithm. *IEEE Transactions on Parallel and Distributed Systems*, 3(1), January 1992.
16. G. Balbo, G. Chiola, G. Franceschinis, and G. Molinar Roet. On the efficient construction of the tangible reachability graph of Generalized Stochastic Petri Nets. In *Proceedings International Workshop on Petri Nets and Performance Models - PNPM87*. IEEE Computer Society, 1987.
17. F. Bause. Queueing Petri nets: a formalism for the combined qualitative and quantitative analysis of systems. In *Proceedings International Workshop on Petri Nets and Performance Models - PNPM93*, pages 14–23. IEEE Computer Society, 1993.
18. C. Beounes and et al. SURF-2: a program for dependability evaluation of complex hardware and software systems. In *Proceedings 23-th International Symposium on Fault-Tolerant Computing*, TOULOUSE, June 1993. IEEE.
19. A. Bertoni and M. Torelli. Probabilistic Petri nets and semi Markov processes. In *Proceedings 2-nd European Workshop on Petri Nets*, 1981.
20. A. Bobbio and M. Telek. Transient analysis of a preemptive resume M/D/1/2/2 through Petri nets. Technical report, Department of Telecommunications - Technical University of Budapest, April 1994.
21. A. Bobbio and M. Telek. Markov regenerative SPN with non-overlapping activity cycles. In *International Computer Performance and Dependability Symposium - IPDS95*, pages 124–133. IEEE Computer Society Press, 1995.
22. R.J. Boucherie. A characterization of independence for competing Markov chain with applications to stochastic Petri nets. In *Proceedings 5th International Workshop on Petri Nets and Performance Models - PNPM93*, pages 117–126. IEEE Computer Society, 1993.
23. R.J. Boucherie and M. Sereno. On the traffic equations for batch routing queueing networks and stochastic Petri nets. Technical Report ERCIM-04/94-R032, European Research Consortium for Informatics and Mathematics, 1994. To appear in the journal *A. on Appl. Prob.*
24. S.C. Bruell and S. Ghanta. Throughput bounds for generalized stochastic Petri ne modelst. In *International Workshop Timed Petri Nets*, pages 250–261, Torino (Italy), 1985. IEEE Computer Society Press No. 674.
25. P. Buchholz. A hierarchical view of GCSPNs and its impact on qualitative and quantitative analysis. *Journal of Parallel and Distributed Computation*, 15:207–224, 1992.

26. P. Buchholz. Hierarchical structuring of superposed GSPNs. In *7-th International Conference on Petri Nets and Performance Models - PNPM97*, pages 81–90. IEEE Computer Society, 1997.
27. P. Buchholz and P. Kemper. Numerical analyisis of stochastic marked graphs. In *Proc. 6^{th} Intern. Workshop on Petri Nets and Performance Models*, pages 32–41, Durham, NC, USA, October 1995. IEEE-CS Press.
28. P. Bucholz. Aggregation and reduction techniques for hierarchical GCSPNs. In *Proceedings International Workshop on Petri Nets and Performance Models - PNPM93*, pages 216–225. IEEE Computer Society, 1993.
29. J. Campos, G. Chiola, J.M. Colom, and M. Silva. Properties and performance bounds for timed marked graphs. *IEEE Transactions on Circuits and Systems - I: Fundamental Theory and Applications*, 39:386–401, 1992.
30. J. Campos, G. Chiola, and M. Silva. Ergodicity and throughput bounds of Petri nets with unique consistent firing count vector. *IEEE Transactions on Software Engineering*, 17:117–125, 1991.
31. J. Campos, J.M. Colom, H. Jugnitz, and M. Silva. Approximate throughput computation of stochastic marked graphs. *IEEE Transactions on Software Engineering*, 20:526–535, 1994.
32. J. Campos, M. Silva, and S. Donatelli. Structures solution of stochastic DSSP systems. In *7-th International Conference on Petri Nets and Performance Models - PNPM97*, pages 91–100. IEEE Computer Society, 1997.
33. S. Caselli, G. Conte, and P. Marenzoni. Massively parallel analysis GSPN models. In *Proceedings International Conference on Application and Theory of Petri Nets*, pages 181–200. Springer Verlag - LNCS, Vol 935, 1995.
34. G. Chiola, C. Anglano, J. Campos, J.M. Colom, and M. Silva. Operational analysis of timed Petri nets and application to the computation of performance bounds. In *Proceedings 5th International Workshop on Petri Nets and Performance Models - PNPM93*, pages 128–137. IEEE Computer Society, 1993.
35. G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic Well-Formed coloured nets and multiprocessor modelling applications. In K. Jensen and G. Rozenberg, editors, *High-Level Petri Nets. Theory and Application*. Springer Verlag, 1991.
36. G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. GreatSPN1.7: GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets. *Performance Evaluation*, 24, 1995. Special Issues on Performance Modelling Tools.
37. Hoon Choi, V.G. Kulkarni, and K. Trivedi. Transient analysis of deterministic and stochastic Petri nets. In *Proceedings of the 14-th International Conference on Application and Theory of Petri Nets*, Chicago, June 1993.
38. Hoon Choi, V.G. Kulkarni, and K. Trivedi. Markov regenerative stochastic Petri nets. *Performance Evaluation*, 20:337–357, 1994.
39. G. Ciardo, J. Gluckman, and D. Nicol. Distributed state-space generation of discrete state stochastic models. *ORSA J. Comp. - To appear*, 1997.
40. G. Ciardo and C. Lindemann. Analysis of deterministic and stochastic Petri nets. In *Proceedings International Workshop on Petri Nets and Performance Models - PNPM93*, pages 160–169. IEEE Computer Society, 1993.
41. G. Ciardo, J. Muppala, and K.S. Trivedi. SPNP: stochastic Petri net package. In *Proceedings International Workshop on Petri Nets and Performance Models - PNPM89*, pages 142–151. IEEE Computer Society, 1989.
42. E. Cinlar. *Introduction to Stochastic Processes*. Prentice-Hall, Englewood Cliffs, 1975.

43. J.W. Cohen. *The Single Server Queue*. Elsevier, 1969.
44. J.L. Coleman, W. Henderson, and P.G. Taylor. Product form equilibrium distributions and an algorithm for classes of batch movement queueing networks and stochastic Petri nets. *Performance Evaluation*, 26:159–180, 1996.
45. R.B. Cooper. *Introduction to Queueing Theory*. MacMillan, New York, 1972.
46. J.A. Couvillon, R. Freire, R. Johnson, W.D. Obal, M.A. Qureshi, M. Rai, W. Sanders, and J.E. Tvedt. Performability modeling with UltrasSAN. *IEEE Software*, 8:69–80, September 1991.
47. D.R. Cox and W.L. Smith. *Queues*. Wiley, 1961.
48. A. Cumani. Esp - A package for the evaluation of stochastic Petri nets with phase-type distributed transition times. In *Proceedings International Workshop Timed Petri Nets*, pages 144–151, Torino (Italy), 1985. IEEE Computer Society Press no. 674.
49. S. Donatelli. Superposed stochastic automata: a class of stochastic Petri nets amenable to parallel solution. *Performance Evaluation*, 18:21–36, 1993.
50. S. Donatelli. Superposed generalized stochastic Petri nets: definition and efficient solution. In *Proceedings International Conference on Application and Theory of Petri Nets*, pages 258–277. Springer Verlag - LNCS, Vol 815, 1994.
51. S. Donatelli and M. Sereno. On the product form solution for stochastic Petri nets. In *Proceedings International Conference on Application and Theory of Petri Nets*, pages 154–172. Springer Verlag - LNCS, Vol 616, 1993.
52. J. Bechta Dugan, K. Trivedi, R. Geist, and V.F. Nicola. Extended stochastic Petri nets: applications and analysis. In *Proceedings PERFORMANCE '84*, Paris, 1984.
53. W. Feller. *An Introduction to Probability Theory and its Applications - Vol. I and II*. Wiley, New York, 1968.
54. G. Florin and S. Natkin. Les reseaux de Petri stochastiques. *Technique et Science Informatique*, 4:143–160, 1985.
55. E. Gelenbe and I. Mitrani. *Analysis and Synthesis of Computer Systems*. Academic Press, New York, 1980.
56. R. German, C. Kelling, A. Zimmermann, and G. Hommel. *TimeNET - A toolkit for evaluating non-markovian stochastic Petri nets*. Report No. 19 - Technische Universität Berlin, 1994.
57. P.J. Haas and G.S. Shedler. Regenerative stochastic Petri nets. *Performance Evaluation*, 6:189–204, 1986.
58. P.J. Haas and G.S. Shedler. Stochastic Petri nets with timed and immediate transitions. *Stochastic Models*, 5:563–600, 1989.
59. S. Haddad and P. Moreaux. Evaluation of high level Petri nets by means of aggregation and decomposition. In *6-th International Conference on Petri Nets and Performance Models - PNPM95*, pages 11–20. IEEE Computer Society, 1995.
60. S. Haddad and P. Moreaux. Asynchronous composition of high level petri nets: a quantitative approach. In *Proceedings of the 17-th International Conference on Application and Theory of Petri Nets, ICATPN '96*, pages 192–211, Osaka, Japan, june 1996. Springer Verlag - LNCS, Vol 1091.
61. B. Haverkort. Matrix-geometric solution of infinite stochastic Petri nets. In *International Computer Performance and Dependability Symposium - IPDS95*, pages 72–81. IEEE Computer Society Press, 1995.
62. B. Haverkort and A. Ost. Steady-state analysis of infinite stochastic Petri nets: a comparing between the spectral expansion and the matrix-geometric method. In *7-th International Conference on Petri Nets and Performance Models - PNPM97*, pages 36–45. IEEE Computer Society, 1997.

63. W. Henderson, D. Lucic, and P.G. Taylor. A net level performance analysis of stochastic Petri nets. *Journal of Australian Mathematical Soc. Ser. B*, 31:176–187, 1989.
64. R.A. Howard. *Dynamic Probabilistic Systems, Volume II: Semi-Markov and Decision Processes*. John Wiley and Sons, New York, 1971.
65. S.M.R Islam and H.H. Ammar. On bounds for token probabilities in a class of generalized stochastic Petri nets. In *Proceedings 3rd International Workshop on Petri Nets and Performance Models - PNPM89*, pages 221–227. IEEE Computer Society, 1989.
66. P. Kemper. Numerical analysis of superposed GSPNs. *IEEE Transactions on Software Engineering*, 22, 1996.
67. P. Kemper. Reachability analysis based on structured representation. In *Proceedings International Conference on Application and Theory of Petri Nets*, pages 269–288. Springer Verlag - LNCS, Vol 1091, 1996.
68. P. Kemper. Transient analysis of superposed GSPNs. In *7-th International Conference on Petri Nets and Performance Models - PNPM97*, pages 101–110. IEEE Computer Society, 1997.
69. L. Kleinrock. *Queuing systems, Volume 1: Theory*. Wiley Interscience, New York, 1975.
70. V.G. Kulkarni. *Modeling and Analysis of Stochastic Systems*. Chapman Hall, 1995.
71. S.S. Lavenberg. *Computer Performance Modeling Handbook*. Academic Press, New York, 1983.
72. A.A. Lazar and T.G. Robertazzi. Markovian Petri net protocols with product form solution. *Performance Evaluation*, 12:67–77, 1991.
73. R. Lepold. PEPNET: A new approach to performability modelling using stochastic Petri nets. In *Proceedings 1st International Workshop on Performability Modelling of Computer and Communication Systems*, pages 3–17, University of Twente - Enschede (NL), 1991.
74. M. Li and N.D. Georganas. Parametric analysis of stochastic Petri nets. In G. Balbo and G. Serazzi, editors, *Computer Performance Evaluation*. Elsevier Science Publishers, 1992.
75. C. Lindemann. An improved numerical algorithm for calculating steady-state solutions of deterministic and stochastic Petri net models. *Performance Evaluation*, 18:75–95, 1993.
76. C. Lindemann. DSPNexpress: a software package for the efficient solution of deterministic and stochastic Petri nets. *Performance Evaluation*, 22:3–21, 1995.
77. J.D.C. Little. A proof of the queueing formula $L = \lambda W$. *Operations Research*, 9:383–387, 1961.
78. Z. Liu. Performance bounds for stochastic timed Petri nets. In G. De Michelis and M. Diaz, editors, *Application and Theory of Petri Nets (16-th International Conference), Lecture Notes in Computer Science*, volume 935, pages 316–334. Springer Verlag, 1995.
79. P. Marenzoni, S. Caselli, and G. Conte. Analysis of large GSPN models: a distributed solution tool. In *7-th International Conference on Petri Nets and Performance Models - PNPM97*, pages 122–131. IEEE Computer Society, 1997.
80. M.K. Molloy. On the integration of delay and throughput measures in distributed processing models. Technical report, Phd Thesis, UCLA, 1981.
81. M.K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers*, C-31:913–917, 1982.

82. M.K. Molloy. Fast bounds for stochastic Petri nets. In *International Workshop Timed Petri Nets*, pages 244–249, Torino (Italy), 1985. IEEE Computer Society Press No. 674.
83. T. Murata. Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, 77:541–580, 1989.
84. S. Natkin. Les reseaux de Petri stochastiques et leur application a l'evaluation des systemes informatiques. Technical report, These de Docteur Ingegneur, CNAM, Paris, 1980.
85. M.F. Neuts. *Matrix Geometric Solutions in Stochastic Models*. Johns Hopkins University Press, Baltimore, 1981.
86. J.L. Peterson. *Petri net theory and the modeling of systems*. Prentice Hall, Englewood Cliffs, 1981.
87. C.A. Petri. Kommunikation mit automaten. Technical report, Doctoral Thesis, University of Bonn, 1962. (Available in English as: *Communication with automata*, Technical Report RADC-TR-65-377, Rome Air Development Center, Griffiss NY, 1966).
88. B.D. Plateau. On the stochastic structure of parallelism and synchronization models for distributed algorithms. In *Proceedings ACM/SIGMETRICS Conference*, pages 147–154, Austin, 1985.
89. L. Recalde, E. Teruel, and M. Silva. On well-formedness analysis: the case of Deterministic Systems of Sequential Processes. In J. Desel, editor, *Structures in Concurrency Theory*, pages 279–293. Springer Verlag, 1995.
90. A. Reibman, R. Smith, and K.S. Trivedi. Markov and Markov reward model transient analysis: an overview of numerical approaches. *European Journal of Operational Research*, 40:257–267, 1989.
91. W. Reisig. *Petri nets - An introduction*. Springer-Verlag, 1982.
92. W.H. Sanders and J.F. Meyer. Reduced base model construction methods for stochastic activity networks. *IEEE Journal on Selected Areas in Communications*, 9(1):25–36, January 1991.
93. M. Sereno. Approximate mean value analysis for stochastic marked graphs. *IEEE Transactions on Software Engineering*, 22:654–664, 1996.
94. M. Sereno and G. Balbo. Computational algorithms for product form solution stochastic Petri nets. In *Proceedings 5th International Workshop on Petri Nets and Performance Models - PNPM93*. IEEE Computer Society, 1993.
95. M. Silva, L. Recalde, and E. Teruel. Linear algebraic techniques for the analysis of liveness of Petri nets. *Journal of Circuits, Systems and Computers*, 1998.
96. G.W. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, New Jersey, 1984.
97. F. J. W. Symons. The description and definition of queueing systems by numerical Petri nets. *Australian Telecommunications Research*, 13:20–31, 1980.
98. F. J. W. Symons. Introduction to numerical Petri nets, a general graphical model of concurrent processing systems. *Australian Telecommunications Research*, 14:28–33, 1980.
99. K. Trivedi. *Probability & Statistics with Reliability, Queueing & Computer Science applications*. Prentice-Hall, 1982.

This article was processed using the LaTeX macro package with LLNCS style