Applications of Non-Markovian Stochastic Petri Nets

Ricardo M. Fricks, Antonio Puliafito, Miklós Telek, Kishor S. Trivedi*

February 6, 2001

Abstract

Petri nets represent a powerful paradigm for modeling parallel and distributed systems. Parallelism and resource contention can easily be captured and time can be included for the analysis of system dynamic behavior. Most popular stochastic Petri nets assume that all firing times are exponentially distributed. This is found to be a severe limitation in many circumstances that require deterministic and generally distributed firing times. This has led to a considerable interest in studying non-Markovian models. In this paper we specifically focus on non-Markovian Petri nets. Analytical approach through the solution of the underlying Markov regenerative process is dealt with and numerical analysis techniques are discussed. Several examples are presented and solved to highlight the potentiality of the proposed approaches.

Stochastic Petri Nets, Markov regenerative processes, preemption policies, numerical analysis.

1 Introduction

Over the past decade, stochastic and timed Petri nets of several kinds have been proposed to overcome limitations on the modeling capabilities of Petri nets (PNs). Although very powerful in capturing synchronization of events and contention for system resources, the original paradigm was not complete enough to capture other elements indispensable for dependability and performance modeling of systems. Thus, new extensions allowing for time and randomness abstractions became necessary. Despite the consensus on which elements to add, a certain uncertainty existed on where to aggregate the proposed extensions. From among several alternatives, a dominant one was soon established where the Petri nets could have transitions that once enabled would fire according to exponential distributions with different rates (EXP transitions). This led to well known net types: Generalized Stochastic Petri Nets (GSPNs) [1] and Stochastic Reward Nets (SRNs) [2].

The resulting modeling framework allowed the definition and solution of stochastic problems enjoying the Markov property [3]: *the probability of any particular future behavior of the process, when its current state is known exactly, is not altered by additional knowledge concerning its past behavior.* These *Markovian stochastic Petri nets* (MSPNs) were very well accepted by the modeling community since a wide range of real dependability and performance models fall in the class of Markov models. Besides the ability to capture various types of system dependencies intrinsic to the underlying Markov models, other advantages of the Petri net framework also contributed to the popularity of the MSPNs. Among these reasons, we point out the power of concisely specifying very large Markov models, and the equal ease with which steady-state, transient, cummulative transient and sensitivity measures could be computed. One of the key restrictions, however, is that only exponentially distributed firing times are captured. This led to the development of non-Markovian stochastic Petri nets.

*R.M. Fricks is with the SIMEPAR Laboratory and Pontificia Universidade Católica do Paraná, Curitiba/PR, Brazil. A. Puliafito is with the Istituto di Informatica, Università di Catania, Catania, Italy. M. Telek is with the Department of Telecommunications, Technical University of Budapest, Budapest, Hungary. K.S. Trivedi is with the Department of Electrical and Computer Engineering, Duke University, Durham/NC, USA. E-mails: fricks@simepar.br, ap@iit.unict.it, telek@hit.bme.hu, and kst@ee.duke.edu.

Non-Markovian stochastic Petri nets (NMSPNs) were then proposed to allow for the high level description of non-Markovian models. Likewise in the original evolutive chain, several alternative approaches to extend the Markovian Petri nets were proposed. Their distinctive feature was the underlying analytical technique used to solve the non-Markovian models. Candidate solution methods considered included the deployment of supplementary variables [4], the use of phase-type expansions approximations [5, 6], and the application of Markov renewal theory [7, 8]. Representative non-Markovian Petri nets proposed, listed according to the underlying solution techniques, are the Extended Stochastic Petri Nets (ESPNs) [9], the Deterministic and Stochastic Petri Nets (DSPNs) [10], the Stochastic Petri Nets with Phase-Type Distributed Transitions (ESPs) [11], and the Markov Regenerative Stochastic Petri Nets (MRSPNs) [12]. As a consequence of these evolutive steps, we observe that the restriction imposed on the distribution functions regulating the firing of timed transitions was progressively relaxed from exponential distributions to a combination of exponential and deterministic distributions, then to any distribution represented by phase type approximations, and finally to any general distribution function (GEN transitions).

However, this flexibility also brought a new requirement with it. If an enabled GEN transition is disabled before firing, a scheduling policy is needed to complete the model definition. Consider the generic client/server NMSPN model in Fig. 1 for instance. Requests from clients arrive according to a Poisson process (EXP transition t_1). Tokens in place P_1 represent clients already in the system. In a single server configuration only one of the queued requests will be serviced at a given time. The service requirement γ_g of each request is sampled from a general distribution function $G_g(t)$ that coordinates the firing of the GEN transition t_2. An age variables a_g associated with a request keeps track of the amount of service actually received by the request. Service will be completed (i.e., transition t_2 will fire) as soon as the age variable a_g of the active request (the one receiving server's attention) reaches the value of its service requirement γ_g. After that, the request leaves the system and its associated age variable is destroyed.

Furthermore, suppose that the server is failure-prone with constant failure and repair rates. A token in place P_2 represents the active state of the server while a token in place P_3 indicates server being down (undergoing repair). Consequently, firing of the EXP transitions t_3 and t_4 correspond to the failure and end-of-repair events associated with the server. Whenever down, the server cannot service new clients or complete the service requirement of the current request, as shown by the inhibitor arc from place P_3 to transition t_2. Clearly a scheduling policy is then necessary to precisely define how the server must proceed when brought up again. In MSPNs with EXP transitions this was not a problem because of the *memoryless* property of the exponential distributions [3][1]. The remaining processing time of an nterrupted request is also represented by the EXP transition t_2.

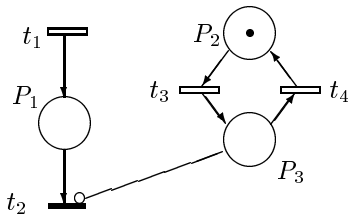

Figure 1: Fault-tolerant client/server model.

In the favorable case, the server is able to completely service the current request before a failure occurs (as shown in Fig. 2a). Otherwise the system behavior depends on the amount of remaining service at the time of the interruption, and whether the service already received by the request will be discarded. The service requirement γ_g may increase or decrease as an indirect consequence of system events responsible by the server interruption. For instance, the failure of the server in Fig. 1 may render certain activities of the client unnecessary, which would then reduce its service requirement to a lower $\gamma_g\prime$ value. Likewise, the age variable a_g related to the active request may also be affected by the server interruption since the amount of service already provided to the request may be

[1] If the scheduling policy is non-work-conserving and the service requirement of the client needs to be preserved then even the EXP transition has to be dealt with like a GEN transition.

preserved or lost. We distinguish both situations calling the first a work conserving scheme, and the second non-work-conserving. With these four conditions we constructed the table in Fig. 2b. Note that, although the service requirement is shown to be increasing after the interruption in the illustration in the bottom row of the table, the situation where $\gamma_g\prime < \gamma_g$ is also possible[2].

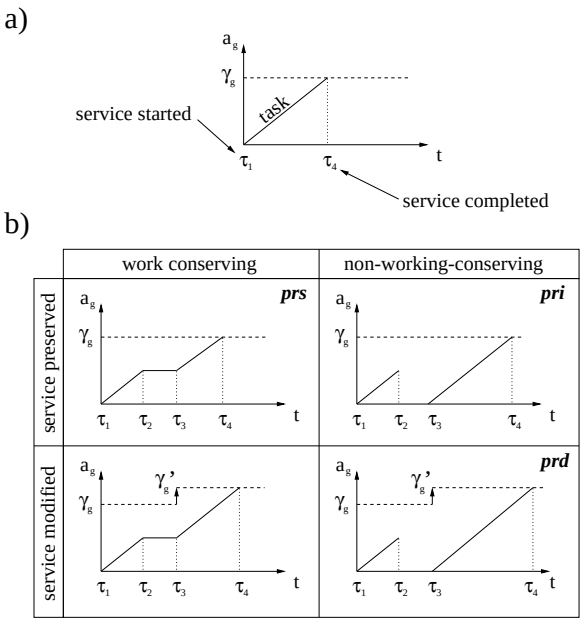

Figure 2: Different scheduling policies.

Fig. 2b can be interpreted from two distinct perspectives. From the clients' perspective, all curves correspond to the same client whose service is momentarily interrupted between times τ_2 and τ_3. From the server's perpective, clients requests live only from interruption-to-interruption. There is a single age variable associated with the server, and what happens after interruptions is defined by the scheduling policy which may be preemptive or non-preemptive, depending on if the server swaps clients before finishing service or not. Preemptive policies are usually based on a hierarchical organization of requests (e.g., priority scheduling) or on an allocation of service based on time quotas (e.g., round-robin scheduling). In this case, system behavior is strongly affected by the preemptive policy and the overall performance will depend on the strategy adopted to deal with the preempted requests, as described in the following:

- The work done on the request prior to interruption is discarded so that the amount of work a_g is lost. The server starts processing a new request which has a work requirement $\gamma_g\prime$; i.e., a new sample is drawn from the service time distribution of the client. The server then starts serving this new request from the beginning (i.e., $a_g = 0$), as shown in the bottom-right sketch in Fig. 2b.

- The server returns back to the preempted request with the original service requirement γ_g. No work is lost so that the age variable retains its value a_g prior to the interruption. The request is resumed from the point of interruption as shown in the top-left sketch in Fig. 2b.

- The server also returns to the same request with the original service requirement γ_g. But the work done prior to the interruption is lost and the age variable a_g is set to zero. The request processing starts from the beginning as shown in the top-right sketch in Fig. 2b.

As in [13], the above policies are referred to as *preemptive repeat different* (prd), *preemptive resume* (prs) and *preemptive repeat identical* (pri), respectively[3]. The case shown in the bottom-left sketch in Fig. 2b is not considered in the literature as it is unrealistic. Note that in [15], the authors indicated the *prd* and *prs* type policies as enabling and age type. The *pri* policy of Petri net transitions was introduced for the first time in [16]. The *prd* and *prs* (with phase-type distributed firing times) policies are the only ones considered in the available tools modeling NMSPNs [17, 11, 18, 19].

Note that when the scheduling is preemptive: (i) the *prs* and *prd* policies produce the same results with EXP transitions, but *pri* is different; (ii) The *prd* and *pri* policies have the same effect for transitions firing according to a deterministic random variable, but *prs* is different; and (iii) otherwise, all three policies will produce distinct results for otherwise same NMSPNs [14].

[2] Naturally, $\gamma_g\prime \geq a_g$ at the time of the interruption needs to be always imposed.

[3] The *prd*, *prs* and *pri* names were borrowed from queueing theory [14].

In this paper, we deal with the general class of non-Markovian Petri nets using examples of MRSPNs, which can be analyzed by means of Markov regenerative processes. The remaining sections of the paper are organized as follows. The next section introduces Markov Regenerative Petri nets and describes how to deal with the underlying Markov Regenerative Process. Section 3 shows how to model a failure/repair process in a parallel machine through MRSPN. Section 4 further extends this model by adopting a different repair facility scheduling scheme. Preemption in a multi-tasking environment is analyzed in Section 5 through the WebSPN tool; the resulting model contains several concurrently enabled general transitions and different memory policies. Conclusions are finally presented in Section 6.

2 Markov Regenerative Petri Nets

MRSPNs allow transitions with zero firing times (immediate transitions), exponentially distributed or generally distributed firing times. The dynamic behavior of an MRSPN is modeled by the execution of the underlying net, which is controlled by the position and movement of tokens. At any given time, the state of an MRSPN is defined by the number of tokens in each of its places, and is represented by a vector called its marking. The set of markings reachable from a given initial marking (i.e., the initial state of the system) by means of a sequence of transition firings defines the reachability set of the Petri net. This set together with arcs joining its markings and indicating the transition that cause the state transitions is called reachability graph.

Two types of markings can be distinguished in the reachability graph. In a vanishing marking at least one immediate transition is enabled to fire, while in a tangible marking no immediate transitions are enabled. Vanishing markings are eliminated before analysis of the MRSPN using elementary probability theory [12]. The resultant reduced reachability graph is a right-continuous, piecewise constant, continuous-time stochastic process $\{Z_t; t \geq 0\}$, where Z_t represents the tangible marking of the MRSPN at time t. Choi, Kulkarni, and Trivedi [12] showed that this marking process is a Markov Regenerative Process (MRGP) (if the GEN transitions are of *prd* type and at most one GEN transition is enabled at a time), a member of a powerful paradigm generally grouped under the name Markov renewal theory [7, 8]. Mathematical definition and solution techniques for MRGP are summarized next.

2.1 Markov Renewal Sequence

Assume a given system we are modeling is described by a stochastic process $\mathbf{Z} \stackrel{d}{=} \{Z_t; t \geq 0\}$ taking values in a countable set Φ. Suppose we are interested in a single event related with the system (e.g., when all system components fail). Additionally, assume the times between successive occurrences of this type of event are independent and identically distributed (*i.i.d.*) random variables. Let $S_0 < S_1 < S_2 < ...$ be the time instants of successive events to occur. The sequence of non-negative *i.i.d.* random variables, $\mathbf{S} \stackrel{d}{=} \{S_n - S_{n-1}; n \in \mathcal{N} = \{0, 1, 2, ...\}\}$ is a *renewal process* [20, 21]. Otherwise, if we do not start observing the system at the exact moment an event has occurred (i.e., $S_0 \neq 0$) the stochastic process $\mathbf{S}$ is a *delayed renewal process*.

However, suppose instead of a single event, we observe that certain transitions between identifiable system states X_n of a subset Ω of Φ, $\Omega \subseteq \Phi$, also resemble the behavior just described, when considered in isolation. Successive times S_n at which a fixed state X_n is entered form a (possibly delayed) renewal process[4]. Additionally, when studying the system evolution we observe that at these particular times the stochastic process $\mathbf{Z}$ exhibits the Markov property, i.e., at any given moment S_n, $n \in \mathcal{N}$, we can forget the past history of the process. The future evolution of the process depends only on the current state at these embedded time points. In this scenario we are dealing with a countable collection of renewal processes progressing simultaneously such that successive states visited form an embedded discrete-time Markov chain (EMC) with state space Ω. The superposition of all the identified renewal processes gives the points $\{S_n; n \in \mathcal{N}\}$, known as *Markov regeneration epochs* (also called *Markov renewal moments*[5]), and to-

[4] We are assuming X_n is the system state at time S_n.

[5] Note that these instants S_n are not renewal moments

gether with the states of the EMC define a Markov renewal sequence.

In mathematical terms, the bivariate stochastic process $(\mathbf{X}, \mathbf{S}) \stackrel{d}{=} \{X_n, S_n; n \in \mathcal{N}\}$ is a Markov renewal sequence (MRS) provided that

$$\Pr\{X_{n+1} = j, S_{n+1} - S_n \leq t \mid X_0, ..., X_n; S_0, ..., S_n\} = \Pr\{X_{n+1} = j, S_{n+1} - S_n \leq t \mid X_n\},$$

for all $n \in \mathcal{N}$, $j \in \Omega$, and $t \geq 0$. We will always assume time-homogeneous MRS's; that is, the conditional transition probabilities $K_{ij}(t)$, where

$$K_{ij}(t) \stackrel{d}{=} \Pr\{X_{n+1} = j, S_{n+1} - S_n \leq t \mid X_n = i\}$$

are independent of n for any $i, j \in \Omega$, $t \geq 0$. Therefore, we can always write

$$K_{ij}(t) = \Pr\{X_1 = j, S_1 \leq t \mid X_0 = i\}, \ \forall i, j \in \Omega, \ t \geq 0.$$

The matrix of transition probabilities $\mathbf{K}(t) \stackrel{d}{=} [K_{ij}(t)]$ is called the *kernel* of the MRS.

2.2 Markov Regenerative Processes

A stochastic process $\{Z_t; t \geq 0\}$ is a Markov regenerative process iff it exhibits an embedded MRS **(X,S)** with the additional property that all conditional finite distributions of $\{Z_{S_n+t}; t \geq 0\}$ given $\{Z_u; 0 \leq u \leq S_n, X_n = i, i \in \Omega\}$ are the same as those of $\{Z_t, t \geq 0\}$ given $X_0 = i$. As a special case, the definition implies that [8]

$$\Pr\{Z_{S_n+t} = j \mid Z_u, 0 \leq u \leq S_n, X_n = i\} = \Pr\{Z_t = j \mid X_0 = i\}, \quad \forall i \in \Omega, \forall j \in \Phi.$$

This means that the MRGP $\{Z_t; t \geq 0\}$ does not have the Markov property in general, but there is a sequence of embedded time points $(S_0, S_1, ..., S_n, ...)$ such that the states $(X_0, X_1, ..., X_n, ...)$ respectively of the process at these points satisfy the Markov property. It also implies that the future of the process $\mathbf{Z}$ from $t = S_n$ onwards depends on the past $\{Z_u, 0 \leq u \leq S_n\}$ only through X_n.

The stochastic process between consecutive Markov regeneration epochs, usually refered to as *subordinated process*, can be any continuous-time discrete-state stochastic process over the same probability space. Recently published examples considered subordinated homogeneous CTMCs [12, 22], non-homogeneous CTMCs [23], semi-Markov processes (SMPs) [24], and MRGPs [25].

=

2.3 Solution of Problems

Let $\mathbf{Z} = \{Z_t; t \geq 0\}$ be a stochastic process with discrete state space Φ and embedded MRS $(\mathbf{X}, \mathbf{S}) = \{X_n, S_n; n \in \mathcal{N}\}$ with kernel matrix $\mathbf{K}(t)$. For such a process we can define a matrix of conditional transition probabilities as:

$$V_{ij}(t) \stackrel{d}{=} \Pr\{Z_t = j \mid Z_0 = i\}, \ \forall i \in \Omega, \forall j \in \Phi, t \geq 0.$$

In many problems involving Markov renewal processes, our primary concern is finding ways to effectively compute $V_{ij}(t)$ since several measures of interest (e.g., reliability and availability) are related to the conditional transition probabilities of the stochastic process.

At any instant t, the conditional transition probabilities $V_{ij}(t)$ of $\mathbf{Z}$ can be written as [7, 8]:

$$\begin{aligned} V_{ij}(t) &= Pr\{Z_t = j, S_1 > t \mid Z_0 = i\} + Pr\{Z_t = j, S_1 \leq t \mid Z_0 = i\} \\ &= \Pr\{Z_t = j, S_1 > t \mid Z_0 = i\} + \sum_{k \in \Omega} \int_0^t dK_{ik}(u) V_{kj}(t-u), \end{aligned}$$

for all $i \in \Omega$, $j \in \Phi$, and $t \geq 0$. If we construct a matrix $\mathbf{E}(t) = [E_{ij}(t)]$ with

$$E_{ij}(t) \stackrel{d}{=} \Pr\{Z_t = j, S_1 > t \mid Z_0 = i\},$$

then the set of integral equations $V_{ij}(t)$ defines a Markov renewal equation, and can be expressed in matrix form as

$$\mathbf{V}(t) = \mathbf{E}(t) + \int_0^t d\mathbf{K}(u)\mathbf{V}(t-u), \qquad (1)$$

where the Lebesgue-Stieltjes integral[6] is taken term by term.

To better distinguish the roles of matrices $\mathbf{E}(t)$ and $\mathbf{K}(t)$ in the description of the MRGP we call

as described in renewal theory, since the distributions of the time interval between consecutive moments are not necessarily *i.i.d.*.

[6] $\int_0^t dK(u)V(t-u) = \int_0^t k(u)V(t-u)du$ when $K(t)$ possesses a density function $k(t) = \frac{dK(t)}{dt}$.

the matrix $\mathbf{E}(t)$ as the *local kernel* of the MRGP, since it describes the state probabilities of the subordinated process during the interval between successive Markov regeneration epochs. Since matrix $\mathbf{K}(t)$ describes the evolution of the process from the Markov regeneration epoch perspective, without describing what happens in between these moments we call it the *global kernel* of the MRGP.

In the special case when the stochastic process $\mathbf{Z}$ does not experience state transitions between successive Markov regeneration epochs; i.e.,

$$Z_t = Z_{S_n^+} \quad \text{where } S_n^+ = max\{S_n \mid S_n \leq t, n \in \mathcal{N}\},$$

$\mathbf{Z}$ is called a semi-Markov process and $\mathbf{E}(t)$ is a diagonal matrix with elements

$$E_{ii}(t) = 1 - K_i(t),$$

where

$$\begin{aligned} K_i(t) &\stackrel{d}{=} \Pr\{S_1 \leq t \mid Y_0 = i\}, \quad \forall i \in \Omega \\ &= \sum_{j \in \Omega} K_{ij}(t) \end{aligned}$$

is the sojourn time distribution in state i. Hence, the global kernel matrix alone (which in this case is usually denoted as $\mathbf{Q}(t)$) completely describes the stochastic behavior of the SMP.

The Markov renewal equation represents a set of coupled *Volterra integral equations of the second kind* [26] and can be solved in time-domain or in Laplace-Stieltjes domain. One possible time domain solution is based on a discretization approach to numerically evaluate the integrals presented in the Markov renewal equation. The integrals in Eqn. 1 are solved using some approximation rule such as trapezoidal rule, Simpson's rule or other higher order quadrature methods. Another time domain alternative is to construct a system of partial differential equations (PDEs), using the method of supplementary variables [4]. This method has been considered for steady-state analysis of DSPNs in [22] and subsequently extended to the transient case in [27].

An alternative to the direct solution of the Markov renewal equation in time-domain is the use of transform methods. In particular, if we define $\mathbf{E}^{\sim}(s) = \int_0^{\infty} e^{-st} d\mathbf{E}(t)$ and $\mathbf{V}^{\sim}(s) = \int_0^{\infty} e^{-st} d\mathbf{V}(t)$, the Markov renewal equation becomes

$$\begin{aligned} \mathbf{V}^{\sim}(s) &= \mathbf{E}^{\sim}(s) + \mathbf{K}^{\sim}(s)\mathbf{V}^{\sim}(s) \\ &= [\mathbf{I} - \mathbf{K}^{\sim}(s)]^{-1}\mathbf{E}^{\sim}(s) \end{aligned}$$

After solving the linear system for $\mathbf{V}^{\sim}(s)$, transform inversion is required[7]. In very simple cases, a closed-form inversion might be possible but in most cases of interest, numerical inversion will be necessary. The transform inversion however can encounter numerical difficulties especially if $\mathbf{V}^{\sim}(s)$ has poles in the positive half of the complex plane.

For a thorough discussion of Markov renewal equation solution techniques see [28, 29], and for generic Volterra integral equations numerical methods see [30, 31]. References for the application of Markov renewal theory in the solution of performance and reliability/availability models see [16, 32, 23, 28, 33, 34, 35, 36, 37].

3 Modeling Failure/Repair Activities in a Parallel Machine Configuration

The use and analysis of MRSPNs is initially demonstrated using a computer system performability model. Two machines (**a** and **b**) are working in a parallel configuration sharing a single repair facility with a First-Come First-Served (FCFS) scheduling discipline. Due to the non-preemptive nature of this discipline, we do not need age variables in this case (once enabled all GEN transitions in the model will never be disabled until firing). We assume that both machines have exponential lifetime distributions with constant parameters λ_a and λ_b respectively. Whenever one of the machines fails it immediately requests repair. When the single repair facility is busy and a second failure occurs, the second machine to fail waits in a repair queue until the first machine is put back into service. The repair-time of the machines is defined by the general distribution functions $G_a(t)$ and $G_b(t)$.

The overall behavior of the system can be understood from the MRSPN illustrated in Fig. 3a. Machine **a** is working whenever there is a token in place P_1. The EXP transition f_a with rate λ_a represents the failure of machine **a**. When machine **a** fails, a token is deposited in place P_6 and its repair is requested. If the repair facility is available (i.e.,

[7] This being the approach addopted in the solution of all examples presented in this paper.

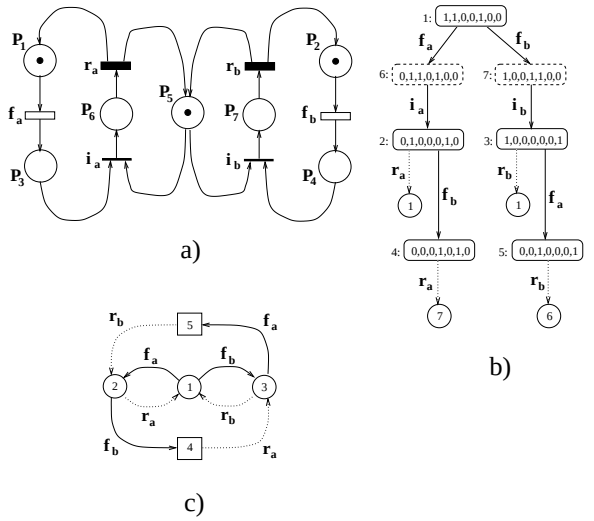

Figure 3: Parallel system model: a) MRSPN; b) reachability graph; and c) state transition diagram.

there is a token in place P_5), it is appropriated with the firing of immediate transition i_a. The GEN transition r_a, firing according to the distribution function $G_a(t)$, represents the random duration of repair. A token in place P_3 means that machine **a** is queued waiting for the availability of the single repair facility while machine **b** is undergoing repair (there is a token in place P_7). A symmetrical set of places and transitions describes the behavior of machine **b**. The system is down whenever there are no tokens in both the places P_1 and P_2.

The reachability graph corresponding to the Petri net is shown in Fig. 3b. Each marking in the graph is a 7-tuple keeping track of the number of tokens in places P_1 through P_7. In the graph, solid arcs represent state changes due to the firing of immediate transitions or EXP transitions, while dotted arcs denote the firing of GEN transitions. The vanishing markings (enclosed by dashed ellipses in the diagram) are eliminated when the reduced reachability graph is constructed (not shown), and based on the reduced version we constructed the state transition diagram of Fig. 3c.

Define the stochastic process $\mathbf{Z} = \{Z_t; t \geq 0\}$ to represent the system state at any instant, where

$$Z_t = \begin{cases} 1 & \textit{if both machines are working at } t \\ 2 & \textit{if machine } \mathbf{a} \textit{ is under repair while} \\ & \textit{machine } \mathbf{b} \textit{ is working at } t \\ 3 & \textit{if machine } \mathbf{b} \textit{ is under repair while} \\ & \textit{machine } \mathbf{a} \textit{ is working at } t \\ 4 & \textit{if machine } \mathbf{a} \textit{ is under repair while} \\ & \textit{machine } \mathbf{b} \textit{ is waiting for repair at } t \\ 5 & \textit{if machine } \mathbf{b} \textit{ is under repair while} \\ & \textit{machine } \mathbf{a} \textit{ is waiting for repair at } t \end{cases}$$

Note that possible values of Z_t are the labels corresponding tangible markings in Fig. 3b. We are interested in computing performability measures associated with the system. To do so, we need to determine the conditional probabilities $\Pr\{Z_t = j \mid Z_0 = 1\}$, $\forall j \in \Phi = \{1, 2, ..., 5\}$. Analysis of the resultant (reduced) reachability graph shows that $\mathbf{Z}$ is an MRGP with an EMC defined by the states 1, 2, and 3; i.e., $\Omega = \{1, 2, 3\}$. We can observe that transitions to states 4 and 5 do not correspond to Markov renewal epochs because they occur while GEN transitions are enabled. An additional step adopted before starting the synthesis of the kernel matrices was the construction of a simplified state transition diagram. Fig. 3c shows a simplified version of the reduced reachability graph where the markings were replaced by the corresponding state indices. We preserved the convention for the arcs and extended the notation by representing states of the EMC by circles, and other states by squares.

The construction of kernel matrices can proceed with the analysis of possible state transitions. The only non-zero elements in global kernel matrix $\mathbf{K}(t)$ correspond to the possible single-step transitions between states of the EMC. Consequently, we have the following structure of the matrix (identified directly from Fig. 3c):

$$\mathbf{K}(t) = \begin{bmatrix} 0 & K_{1,2}(t) & K_{1,3}(t) \\ K_{2,1}(t) & 0 & K_{2,3}(t) \\ K_{3,1}(t) & K_{3,2}(t) & 0 \end{bmatrix}$$

Let the random variables L_a and L_b be the respective time-to-failure of the two machines, we can determine $K_{1,2}(t)$ in the following way:

$$\begin{aligned} K_{1,2}(t) &= Pr\{X_1 = 2, S_1 \leq t \mid X_0 = 1\} \\ &= Pr\{\textit{machine } \mathbf{a} \textit{ fails by time } t \textit{ and} \\ &\qquad \textit{is the first one to fail}\} \end{aligned}$$

$$
\begin{aligned}
&= Pr\{L_a \le t \wedge L_b > L_a\} \\
&= \int_0^t \left[1 - \left(1 - e^{-\lambda_b \tau}\right)\right] d\left\{1 - e^{-\lambda_a \tau}\right\} \\
&= \int_0^t e^{-\lambda_b \tau} \lambda_a e^{-\lambda_a \tau} d\tau \\
&= \frac{\lambda_a}{\lambda_a + \lambda_b}\left[1 - e^{-(\lambda_a+\lambda_b)t}\right].
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
K_{1,3}(t) &= Pr\{X_1 = 3, S_1 \le t \mid X_0 = 1\} \\
&= Pr\{\textit{machine } \mathbf{b} \textit{ fails by time } t \textit{ and} \\
&\qquad \textit{is the first one to fail}\} \\
&= \frac{\lambda_b}{\lambda_a + \lambda_b}\left[1 - e^{-(\lambda_a+\lambda_b)t}\right].
\end{aligned}
$$

Determination of the elements $K_{2,1}(t)$ and $K_{2,3}(t)$ is quite alike, so we only show how $K_{2,1}(t)$ is determined. The third row is completelly symmetrical to the second, so it can be easily undestood once $K_{2,1}(t)$ is understood. We need some auxiliary variables to help in the explanation of the constructive process of $K_{2,1}(t)$. Hence, we define the random variables R_a and R_b to respectively represent times necessary to repair machines **a** and **b**. The distribution function of R_a (R_b) is G_a (G_b). Using this new variables we can compute $K_{2,1}(t)$:

$$
\begin{aligned}
K_{2,1}(t) &= Pr\{X_1 = 1, S_1 \le t \mid X_0 = 2\} \\
&= Pr\{\textit{repair of } \mathbf{a} \textit{ is finished by time } t \\
&\qquad \textit{and } \mathbf{b} \textit{ has not failed during the} \\
&\qquad \textit{repair of } \mathbf{a}\} \\
&= Pr\{R_a \le t \wedge L_b > R_a\} \\
&= \int_0^t Pr\{L_b > \tau\} dG_a(\tau) \\
&= \int_0^t \left[1 - \left(1 - e^{-\lambda_b \tau}\right)\right] dG_a(\tau) \\
&= \int_0^t e^{-\lambda_b \tau} dG_a(\tau).
\end{aligned}
$$

To summarize, the elements of the global kernel matrix are:

$$
\begin{aligned}
K_{1,2}(t) &= \frac{\lambda_a}{\lambda_a + \lambda_b}\left[1 - e^{-(\lambda_a+\lambda_b)t}\right], \\
K_{1,3}(t) &= \frac{\lambda_b}{\lambda_a + \lambda_b}\left[1 - e^{-(\lambda_a+\lambda_b)t}\right], \\
K_{2,1}(t) &= \int_0^t e^{-\lambda_b \tau} dG_a(\tau), \\
K_{2,3}(t) &= \int_0^t \left(1 - e^{-\lambda_b \tau}\right) dG_a(\tau), \\
K_{3,1}(t) &= \int_0^t e^{-\lambda_a \tau} dG_b(\tau), \text{ and} \\
K_{3,2}(t) &= \int_0^t \left(1 - e^{-\lambda_a \tau}\right) dG_b(\tau).
\end{aligned}
$$

Note that the global kernel will always be a square matrix. In this case with dimensions 3×3, since we have 3 states in the embedded Markov chain. However, the local kernel matrix is not necessarily a square matrix, since the cardinality of the state space of $\mathbf{Z}$ can be larger than the cardinality of the state space of the embedded Markov chain. This can be seen, for instance, in this system since the embedded Markov chain has only 3 states while the MRGP has 5 possible states.

We construct the local kernel matrix $\mathbf{E}(t)$ following a similar inductive procedure. In this case we are looking for the probability that the MRGP will move to a given state before the next Markov renewal moment. Careful analysis of Fig. 3c reveals the structure of the local kernel matrix $\mathbf{E}(t)$:

$$
\begin{bmatrix}
E_{1,1}(t) & 0 & 0 & 0 & 0 \\
0 & E_{2,2}(t) & 0 & E_{2,4}(t) & 0 \\
0 & 0 & E_{3,3}(t) & 0 & E_{3,5}(t)
\end{bmatrix}
$$

Since in a single step the system can only go from state 1 to the other two states of the EMC then $E_{1,1}$ should be the complementary sojourn time distribution function in state 1, that is,

$$
\begin{aligned}
E_{1,1} &= 1 - (K_{1,2}(t) + K_{1,3}(t)) \\
&= e^{-(\lambda_a+\lambda_b)t}.
\end{aligned}
$$

The difficulty comes with the induction of $E_{2,2}(t)$ and $E_{2,4}(t)$ (complement of $E_{2,2}(t)$). Once we solve for these, we have the solution for the remaining components of the matrix due to the symetry of the problem. Therefore, we explain the induction process that leads to $E_{2,2}(t)$:

$$
\begin{aligned}
E_{2,2}(t) &= Pr\{Z_t = 2, S_1 > t \mid X_0 = 2\} \\
&= Pr\{\textit{repair of } \mathbf{a} \textit{ is not finished up to } t \\
&\qquad \textit{and } \mathbf{b} \textit{ has not failed until } t\} \\
&= Pr\{\textit{repair of } \mathbf{a} \textit{ is not finished up to } t \\
&\qquad \times Pr\{\mathbf{b} \textit{ has not failed until } t\} \\
&= [1 - G_a(t)]e^{-\lambda_b t}.
\end{aligned}
$$

We can now express the remaining non-zero elements of the local kernel matrix as

$$E_{2,4}(t) = \left(1 - e^{-\lambda_b t}\right) G_a^c(t)$$
$$E_{3,3}(t) = e^{-\lambda_a t} G_b^c(t)$$
$$E_{3,5}(t) = \left(1 - e^{-\lambda_a t}\right) G_b^c(t)$$

with

$$G_a^c(t) = 1 - G_a(t), \text{ and}$$
$$G_b^c(t) = 1 - G_b(t).$$

We can always verify our answers by summing the elements in each row of both kernel matrices. Corresponding row-sums of the two matrices must add to unity, condition that is easily verified to hold in the example.

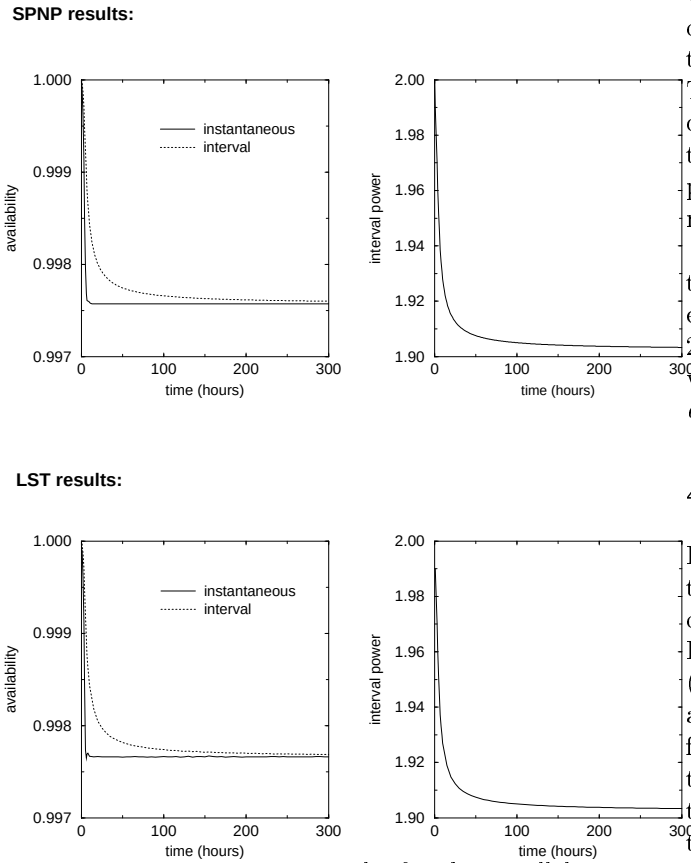

Figure 4: Numerical results for the parallel system with non-premptive repair.

The kernel matrices determined can then be substituted in Equation (1) and the resultant system of coupled integral equations solved using one of the approaches described in [28, 29]. The resultant plots, labelled LST in Fig. 4, report system availability and performability computed when time to repair is deterministic; i.e.,

$$G_a(t) = U(t - \mu_a), \qquad \mu_a > 0$$
$$G_b(t) = U(t - \mu_b), \qquad \mu_b > 0$$

where $U(t)$ is the unit step function; the failure rates (parameters λ_a and λ_b) are identical μ_b) takes 5 hours. The interval availability is the expected proportion of time the system is operational during the period $[0, t]$:

$$\bar{A}(t) = \frac{1}{t} \int_0^t E[X(\tau)] \mathrm{d}\tau,$$

when the discrete random variable X represents the operational status of the system; i.e., $X(t) = 1$ if the system is operatinal at time t, and 0 if it is not. The performability measure plotted in the figure corresponds to the interval processing capacity of the system, with the convention that a unit of computing capacity corresponds to that of one active machine.

Following the approach used in [34], we also plotted corresponding Markovian system results, where each DET transition was replaced by an equivalent 25-stage Erlang subnet. The Markovian models were solved using the *Stochastic Petri Net Package* (SPNP) introduced in [38].

4 Preemptive LCFS repair

Fig. 5 shows the *PN* which describes the behavior of the system containing the same machines **a** and **b** of the previous example and applies the preemptive LCFS scheduling scheme. The repair of machine **a** (**b**), represented by a token at P_6 (P_7) is preempted as soon as machine **b** (**a**) fails, i.e., transition f_b (f_a) fires. In this case the repair facility is assigned to the machine which failed later ($i_a^{'}$ or $i_b^{'}$ fires and a token is placed to P_8 or P_9). After the repair of the last failed machine (firing of $r_a^{'}$ or $r_b^{'}$) the repair facility returns to the completion of preempted repair action. Different memory policies can be considered depending on whether the repairman is able to "remember" the work already performed on the machine before preemption or not. In the case

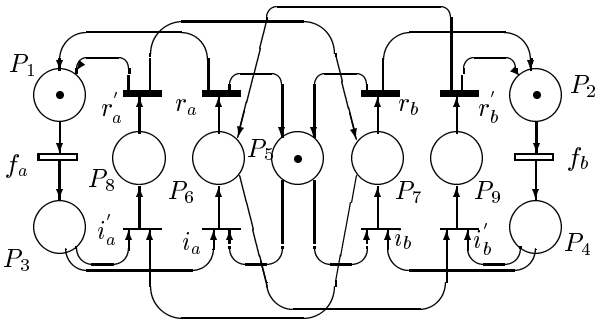

Figure 5: Preemptive LCFS repair with non-identical machines

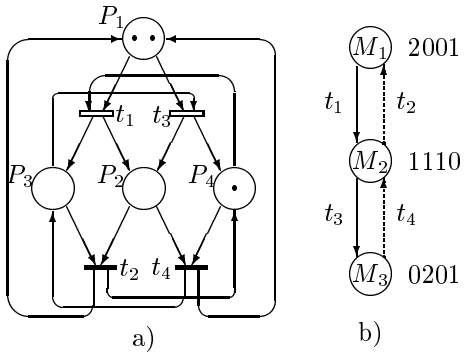

Figure 6: Preemptive LCFS repair with identical machines

that the prior work is lost due to the interruption and the repair must be repeated from scratch with an identical repair time requirement (*pri* policy) or with a repair time resampled from the original cumulative distribution function (*prd* policy). In the case that the prior work is not lost and the time to complete the preempted repair equals the residual repair time given the portion of work already completed before preemption (*prs* policy). The PN on Fig. 5 captures the different memory policies for repair by assigning transitions r_a and r_b the appropriate preemption policies. (The preemption policies of transitions $r_a^{'}$ and $r_b^{'}$ are not relevant since $r_a^{'}$ and $r_b^{'}$ cannot be preempted.)

We analyze a simplified version of the two machine system with preemptive LCFS repair and with *prs* policy. We assume that the two machines are statistically identical, i.e., their failure and repair time distributions are the same. Fig. 6a shows a *PN* which describes the behavior of the system of two identical machines with LCFS scheduling. Tokens in place P_1 represent operational machines, tokens in P_2 count failed machines (including the one under repair), and a token in place P_4 the availability of the single repair facility. In the initial marking $M_1 = (2\ 0\ 0\ 1)$ (Fig. 6b), t_1 is the only enabled transition. Firing of t_1 represents the failure of the first machine and leads to state $M_2 = (1\ 1\ 1\ 0)$. In M_2, transitions t_2 and t_3 are competing. The GEN transition t_2 represents the repair of the failed machine and its firing returns the system to the initial state M_1. The EXP transition t_3 represents the failure of the second machine and its firing disables t_2 by removing one token from P_3 (the first repair becomes dormant). In $M_3 = (0\ 2\ 0\ 1)$ one machine is under repair and the other repair is dormant, and the only enabled transition is the repair of the last failed machine. Firing of the GEN transition t_4 leads the system again to M_2, where the dormant repair is resumed. Assume that the failure times of both machines are exponentially distributed with parameter λ so that the EXP transitions t_1 and t_3 have firing rates 2λ and λ, respectively.

The preemptive policy of transition t_2 has to be assigned based on the system behavior to be evaluated. (The preemptive policy of transition t_4 is irrelevant since t_4 can not be preempted.) Assigning a *prd* policy to t_2 means that each time t_2 is disabled by the failure of the second machine (t_3 fires before t_2), the corresponding age variable a_2 is reset. As soon as t_2 becomes enabled again (the second repair completes and t_4 fires) no memory is kept of the prior repair period, and the execution of the repair restarts from scratch. The *prd* service policies, like this one, are covered by the model definition in [39, 40].

The case when a *pri* policy is assigned to t_2 is very similar to the previous one except that as soon as t_2 becomes reenabled (the second repair completes and t_4 fires), the same repair (same firing time sample) has to be completed from the beginning. This type of *pri* memory policy is covered by the model definition in [16], and can be analyzed by the transform domain method discussed there.

Hereafter we assume that a *prs* policy is assigned to t_2. When a *prs* policy is assigned to t_2, each time

t_2 is disabled without firing (t_3 fires before t_2) the age variable a_2 is not reset. Hence, as the second repair completes (t_4 fires), the system returns to M_2 keeping the value of a_2, so that the time to complete the interrupted repair can be evaluated as the original repair requirement minus the current value of a_2. The age variable a_2 counts the total time during which t_2 is enabled before firing, and is equal to the cumulative sojourn time in M_2. The Markov renewal moments in the marking process correspond to the epochs of entrance to markings in which the age variables associated with all the transitions are equal to zero. By inspecting Fig. 6b, the Markov renewal moments are the epochs of entering M_1 and of entering M_2 from M_1.

The subordinated process starting from marking M_1 is a single step *CTMC* (since t_1 the only enabled EXP transition) and includes the only immediately reachable state M_2 (Markovian regeneration period).

The subordinated process starting from marking M_2 includes all the states reachable from M_2 before firing of t_2; i.e., M_2 and M_3. Since M_2 is the only state in which t_2 is enabled, the age variable a_2 increases only in marking M_2 and maintains its value in M_3. The firing of t_2 can only occur from M_2 leading to marking M_1.

Notice that the subordinated process starting from M_2 is semi-Markov since the firing time of t_4 is generally distributed. The age variable a_2 grows whenever the MRSPN is in marking M_2, and the firing of t_2 occurs when a_2 reaches the actual value of the firing time (which is generally distributed with cumulative distribution function $G(t)$). If we condition that the firing time of t_2 to w, w acts an absorbing barrier for the accumulation functional represented by the age variable a_2, the firing time of t_2 is determined by the first passage time of a_2 across the absorbing barrier w.

The closed form Laplace-Stieltjes transform expressions of the kernel matrices of the LCFS repair *prs* case are derived here in detail, applying the technique based on the Markov renewal theory. We build up the $\mathbf{K}^{\sim}(s)$ and $\mathbf{E}^{\sim}(s)$ matrices row by row by considering separately all the states that can be regeneration states and can originate a subordinated process. M_3 can never be a regeneration state since t_2 is always active when entering to M_3, $\Omega = \{M_1, M_2\}$. The fact that M_3 is not a regeneration marking, means that the process can stay in M_3 only between two successive Markov renewal moments.

The starting regeneration state is M_1 - (Markovian regeneration period) No general transition is enabled and the next regeneration state can only be state M_2. The non-zero elements of the first row of the kernel matrices are

$$K_{12}^{\sim}(s) = \frac{2\lambda}{s + 2\lambda} \quad \text{and} \quad E_{11}^{\sim}(s) = \frac{s}{s + 2\lambda}$$

The starting regeneration state is M_2 - Transition t_2 is GEN so that the next regeneration time point is the epoch of firing of t_2. The subordinated process starting from M_2 comprises states M_2 and M_3 and is an SMP (since t_4 is GEN) whose kernel is:

$$Q^{\sim}(s) = \begin{bmatrix} 0 & \frac{\lambda}{s + \lambda} \\ G^{\sim}(s) & 0 \end{bmatrix}$$

where $G^{\sim}(s)$ is the LST of the distribution function of the firing time of t_4.

The transition t_2 fires when the age variable a_2 reaches actual sample of the firing time γ_2. In general, when a GEN transition is active the occurence of a Markov renewal epoch in the marking process of an NMSPN is due to one of the following two reasons:

- the GEN transition fires,
- the GEN transition of *prd* type becomes disabled.

For the analysis of subordinated processes of this kind three matrix functions $\mathbf{F}^i(t,w)$, $\mathbf{D}^i(t,w)$ and $\mathbf{P}^i(t,w)$ (where t denotes the time, w a fixed firing time sample, and the superscript i refers to the initial (regeneration) state of the subordinated process) were introduced in [24]. $\mathbf{F}^i(t,w)$ refers to the case when the next regeneration moment is because of the firing of the GEN transition with the (fixed) firing time sample w. For the analysis of this case an additional matrix ($\mathbf{\Delta}^i$ referred to as branching probability matrix) is introduced, as well, to describe the state transition subsequent to the firing of the GEN transition. $\mathbf{D}^i(t,w)$ captures the case when the next regeneration moment is caused by the disabling of the *prd* type GEN transition. And

$\mathbf{P}^i(t,w)$ describes the state transition probabilities inside the regeneration period.

Since transition t_2 is of *prs* type the matrix function $\mathbf{D}^i(t,w)$ does not play a role in the analysis of the subordinated process starting from marking M_2. The remaining functions can be evaluated based on the kernel of the subordinated SMP $(\mathbf{Q^i(t)} = \{Q^i_{k\ell}(t)\})$ [24]:

$$\begin{aligned} F^{i\sim *}_{k\ell}(s,v) &= \delta_{k\ell}\frac{r_k\left[1-Q^{i\sim}_k(s+vr_k)\right]}{s+vr_k} + \\ &\quad \sum_{u\in R^i} Q^{i\sim}_{ku}(s+vr_k)F^{i\sim *}_{u\ell}(s,v) \\ P^{i\sim *}_{k\ell}(s,v) &= \delta_{k\ell}\frac{s\left[1-Q^{i\sim}_k(s+vr_k)\right]}{v(s+vr_k)} + \\ &\quad \sum_{u\in R^i} Q^{i\sim}_{ku}(s+vr_k)P^{i\sim *}_{u\ell}(s,v) \end{aligned}$$

where $Q^i_k(t) = \sum_\ell Q^i_{k\ell}(t)$; s is the time variable and v is the barrier level variable in transform domain; r_k is the indicator that the active GEN transition is enabled in state k; R^i is the part of the state space reachable during the subordinated process; and the superscript $\sim(*)$ refers to Laplace-Stieltjes (Laplace) transform.

Given that $G_g(t)$ is the distribution function of the firing time of the GEN transition, the elements of the i-th row of matrices $\mathbf{K}(t)$ and $\mathbf{E}(t)$ can be expressed as follows, as a function of the matrices $\mathbf{P}^i(t,w)$, $\mathbf{F}^i(t,w)$ and $\mathbf{D}^i(t,w)$:

$$\begin{aligned} K_{ij}(t) &= \int_0^\infty \left[\sum_{k\in R^i} F^i_{ik}(t,w)\Delta^i_{k,j} + D^i_{ij}(t,w)\right] dG_g(w) \\ E_{ij}(t) &= \int_0^\infty P^i_{ij}(t,w)dG_g(w) \end{aligned}$$

To evaluate the 2nd row of the kernel matrices we are applying these results for the subordinated process starting from regeneration state M_2. Doing so we obtain the following expressions for the non-zero matrix entries:

$$F^{\sim *}_{22}(s,v) = \frac{1}{s+v+\lambda-\lambda G^\sim(s)}$$

$$P^{\sim *}_{22}(s,v) = \frac{s/v}{s+v+\lambda-\lambda G^\sim(s)}$$

$$P^{\sim *}_{23}(s,v) = \frac{\lambda(1-G^\sim(s))/v}{s+v+\lambda-\lambda G^\sim(s)}$$

Unconditioning with respect to the firing time distribution of t_2, and after inverting the Laplace transform (LT) with respect to v, the non-zero entries of the 2nd row of the LST matrix functions $\mathbf{K}^\sim(s)$ and $\mathbf{E}^\sim(s)$ become:

$$\begin{aligned} K^\sim_{21}(s) &= \int_{w=0}^\infty e^{-w(s+\lambda-\lambda G^\sim(s))}\, dG(w) \\ &= G^\sim(s+\lambda-\lambda G^\sim(s)) \\ E^\sim_{22}(s) &= \frac{s[1-G^\sim(s+\lambda-\lambda G^\sim(s))]}{s+\lambda-\lambda G^\sim(s)} \\ E^\sim_{23}(s) &= \frac{\lambda(1-G^\sim(s))\,[1-G^\sim(s+\lambda-\lambda G^\sim(s))]}{s+\lambda-\lambda G^\sim(s)} \end{aligned}$$

The LST of the state probabilities are obtained by solving the Markov renewal equation in transform domain. The time domain probabilities are calculated by numerically inverting the result by resorting to the Jagerman method [41].

To evaluate the performance of the different scheduling schemes, we compared the availability and processing power of the FCFS and the LCFS repair schemes with two different repair time distributions. The FCFS scheme was evaluated by the time domain method introduced in the previous section and the LCFS scheme was evaluated by the above transform domain method. It is assumed that the system is available when at least one machine is working (marking M_1 and M_2) and that the system performance doubles when both machines are working. The failure times of both machines are exponentially distributed with rates $\lambda = \lambda_a = \lambda_b = 0.01$. The repair times of both machines are assumed to be:

- deterministic $\tau = 5$, hence

$$G(t) = G_1(t) = G_2(t) = U(t-\tau)$$

$$G^\sim(s) = e^{-\tau s},$$

- hyperexponentially distributed with $p = 0.625$; $\mu_1 = 0.5$; $\mu 2 = 0.1$, hence

$$G(t) = G_1(t) = G_2(t) = 1 - pe^{-\mu_1 t} - (1-p)e^{-\mu_2 t}$$

$$G^\sim(s) = \frac{p\mu_1}{\mu_1+s}\,\frac{(1-p)\mu_2}{\mu_2+s}.$$

The mean repair time is 5 in both cases. Fig. 7a and 7b show the instantaneous and the interval

measures of availability and processing power with deterministic repair time, respectively. The dotted line shows the instantaneous and the short dashed line shows the interval availability/power with LCFS repair, while the long dashed line shows the instantaneous and the solid line shows the interval availability/power with FCFS repair. It can be observed that the FCFS scheduling performs better in this case. The availability and processing power results for the hyperexponential repair time distribution are plotted on Fig. 7c and 7d, respectively. In these figures the dotted line shows the instantaneous availability/power with LCFS repair, while the dashed line shows the instantaneous availability/performability with FCFS repair. As can be seen from these figures, in contrast with the deterministic repair time the LCFS scheduling performs better with the hyperexponential repair time distribution.

5 Modeling preemption in a multi-tasking environment

NMSPN require complex solution techniques mainly based on theory of Markov regenerative processes. Software packages are then required which can hide solution and implementation details. A big boost in this direction came from two well-known tools, DSPNexpress [42] and TimeNET [43, 44]. Recently, a new software package for non-Markovian Petri nets has been developed in a joint effort between the Universities of Catania and Budapest. This tool, named WebSPN [45],

provides a discrete time approximation of the stochastic behaviour of the marking process which results in the possibility to analyze a wider class of *PN* models with *prd, prs* and *pri* concurrently enabled generally distributed transitions. The approximation of the continuous time model at equispaced discrete time points involves the analysis of the system behavior over a time interval based on the system state at the beginning of the interval and the past history of the system. A Web-centered view has been adopted in its development in order to make it easily accessible from any node connected with the Internet as long as it possesses a Java-enabled Web browser. Sophisticated security mechanisms have also been implemented to regulate the access to the tool which are based on the use of public and private electronic keys. WebSPN is available at the following site: *http://sun195.iit.unict.it/~webspn/webspn2/*

5.1 Model description

In this section we describe and solve a model of Petri net with several concurrently enabled GEN transitions and different memory policies. The system moves between an operative phase, where useful work is produced, and a phase of maintenance where the processing is temporarily interrupted.

The *Petri net* shown in Fig. 8 represents the model of the system that consists of three functional blocks generically referred to as *Block1, Block2* and *Block3*. Block1 models the alternation of the system between the operative phase and the maintenance phase. Block2 models the two sequential phases of processing of jobs. Finally, Block3 models the alternation of the system during the operative phase between the phase of pre-processing and the one of processing of jobs.

Within Block1, the two states of operation where the system can be are represented by places *user* and *system* and by transitions *U_time* and *S_time*. A token in place *user* denotes the operative state, while a token in place *system* denotes the maintenance one. The duration of the operative phase is denoted by transition *U_time*, while the maintenance one is denoted by transition *S_time*. The inhibitor arcs outgoing from place *system* and leading to the timed and immediate transitions contained in Block2 and Block3 *producer, cons1, busy_prod, idle_prod, busy2, idle2* are used for interrupting the activity of the system during the phase of maintenance.

Block2 models the processing of jobs. In particular, the number of jobs to be processed is denoted by the number of tokens contained in place *work*, while the time of pre-processing of each job is represented by transition *producer*. Pre-processed jobs are queued in a buffer (place *buff1*) waiting for the second phase of processing (transition *cons1*).

In Block3, the alternation between the phases of pre-processing and processing of jobs is represented through places *slot1* and *slot2* and transitions *busy_brod, busy2, idle_prod, idle2*. A token in place *slot1* denotes that the system is executing the pre-processing of a job, while a token in place *slot2*

denotes the execution of a phase of processing. An inhibitor arc between *slot1* and *cons1* deactivates the phase of processing when the pre-processing one is active. In the same way, the inhibitor arc between *slot2* and *producer* deactivates the phase of pre-processing when the processing one is active. The time that the system alternately spends for these two activities is represented by transitions *busy_prod* and *busy2*. The immediate transition *idle_prod* (*idle2*) prevents the system to remain in phase 1 (2), even if no job is to be processed. The function of the inhibitor arcs from place *work* to transition *idle_prod* and from place *buff* to transition *idle2* is to enable such transitions when no job is to be processed in the corresponding phase of processing.

Immediate transition *end* and place *Stop* are used for modeling the processing of all the jobs assigned to the system at the beginning. In fact, transition *end* is inhibited until at least one token is present in places *work* and *buff*. When all the jobs have been processed, transition *end* fires, and immediately moves a token to place *Stop*. All the activities of the system are thus interrupted through the inhibitor arcs outgoing from place *Stop*.

The measure that we evaluate from this model is the distribution of the time required for completing the set of jobs assigned to the system at the beginning. It can be obtained as the distribution of having a token in place *Stop*.

With regard to the distributions of the firing times to be assigned to timed transitions, we assume that the firing times of transitions *U_time, S_time, busy_brod, busy2* are deterministic. We assume that the firing times of transitions *producer* and *cons1* are respectively distributed uniformly and exponentially. The measures considered can therefore be evaluated by changing the memory policy associated with transitions *producer* and *cons1*.

In the case of *prd* policy, the temporary interruption of the processing of a job (either because the whole system enters the phase of maintenance, or because, even if the system is in the production phase, it interrupts the pre-processing phase for changing to the processing one or vice versa) causes the interrupted job to be discarded. A new job is executed when the system is available again. The correspondence with a real system is perhaps hard to find; however, we note that *prd* policy is the most commonly used one in the literature.

Conversely, by adopting *prs* policy, we keep a memory of the work that we were executing. In this case, when transition *producer* is disabled, we keep a memory of the work that has already been executed on the job considered. When the system enters the operative state again, the pre-processing of the job continues from the point we had reached. In this case, the model can represent a system of *manufacturing*, where a machine used for production alternates cycles of production and cycles of maintenance, and production takes place in two sequential phases. We note that *prd* and *prs* policies are equivalent for transition *cons1*, since this one is and EXP transition.

With *pri* policy, when transition *producer* is disabled, the work that had already been produced is lost, but we keep a memory of the job that we were processing. When the transition is enabled again we start from zero, but the amount of work to be produced on the job remains the same, because the job has not been changed. Such a behavior can be easily noted when accessing transactional databases, where each transaction is atomic (i.e., has to be processed with no interruption). If an interruption occurs, the transaction is entirely processed again. If we assume a memory policy like *prs* for transition *cons1*, the model could represent a client/server system where the accesses to database (transition *producer*) take place atomically, and the phase of processing of the query (transition *cons1*) requires a variable time, distributed exponentially.

5.2 Numerical Results

For the solution of the model we assume that the firing time of transition *producer* is distributed uniformly between 0.5 and 1.5; the firing time of transitions *U_time* and *S_time* are deterministic, with a firing time of 1; the firing time of transitions *busy_prod* and *busy2* are deterministic, with a firing time of 0.1; the firing time of transition *cons1* is distributed exponentially, with a firing rate of 0.1; transition *end* is immediate and has a priority of 2; transitions *idle_prod* and *idle2* are immediate and have a priority of 1; the total number of jobs to be processed is 3.

In Fig. 9 we show the distribution of completion time for different memory policies assigned to transitions *producer* and *cons1*. The behavior of

the system changes significantly depending upon the memory policy adopted. The *prs* policy accrues the highest probability of completion within a given time. Both the *prd* and the *prs* policies accomplish the completion of jobs. In fact, curves eventually reach the value 1. Conversely, a different behavior can be observed if we assume a policy like *pri*. In fact, in that case, the resulting distribution is defective, since the unit value is never reached for $t \to \infty$. This is closely connected with the choice of the parameters associated with transitions *producer* and *U_time*. As we note in Fig. 10, when the firing time of transition *U_time* is lower than 1.5, transition *producer* has a positive probability (50%)of not completing its work. Since in the case of *pri* policy the job is processed with the same work requirement, this causes a situation of impasse, which prevents the work assigned to the system to be completed.

Fig. 11 shows how the overall system behavior changes if transition *U_time* is assigned a firing time higher than 1.5 (for example 2.0). In such case, transition *producer* has a finite probability of firing before the system enters the phase of maintenance, and therefore the distribution of completion time with *pri* policy reaches the value 1.

6 Conclusion

We discussed the need for more advanced techniques to capture generally distributed events which occur in everyday life. Among the different approaches proposed in the literature, non-Markovian Petri nets represent a valid analytical alternative to numerical simulation. An approach based on the analysis of the underlying Markov Regenerative Process has been presented. Advanced preemption policies were introduced and several examples solved in detail.

References

[1] M. Ajmone Marsan, G. Conte, and G. Balbo, "A class of generalized stochastic Petri net for the performance evaluation of multiprocessor systems," *ACM Transaction on Computer Systems*, vol. 2, no. 2, pp. 93–122, May 1984.

[2] G. Ciardo, A. Blakemore, P.F. Chimento, J.K. Muppala, and K.S. Trivedi, "Automatic generation and analysis of markov reward models using stochastic reward nets," in *Linear Algebra, Markov Chains, and Queueing Models, IMA Volumes in Mathematics and its Applications*, C. Meyer and R.J. Plemmons, Eds., Heidelberg, Germany, 1992, vol. 48, Springer-Verlag.

[3] Kishor S. Trivedi, *Probability & Statistics with Reliability, Queueing, and Computer Science Applications*, Prentice-Hall, NJ, USA, 1982.

[4] D.R. Cox, "The analysis of non-Markovian stochastic processes by the inclusion of supplementary variables," *Proceedings of the Cambridge Philosophical Society*, vol. 51, no. 3, pp. 433–441, 1955.

[5] R.F. Botta, C.M. Harris, and W.G. Marchal, "Characterizations of generalized hyperexponential distribution functions," *Communications in Statistics - Stochastic Models*, vol. 3, no. 1, pp. 115–148, 1987.

[6] M.F. Neuts, "Renewal process of phase type," *Naval Research Logistics Quartely*, vol. 25, no. 3, pp. 445–454, 1978.

[7] E. Çinlar, *Introduction to Stochastic Processes*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1975.

[8] V.G. Kulkarni, *Modeling and Analysis of Stochastic Systems*, Chapman & Hall, London, UK, 1995.

[9] J. B. Dugan, K. S. Trivedi, R. M. Geist, and V.F. Nicola, "Extended stochastic Petri nets: Applications and analysis," *Performance*, pp. 507–519, 1984.

[10] M. Ajmone Marsan and G. Chiola, "On Petri nets with deterministic and exponentially distributed firing times," *Lecture Notes on Computer Science*, vol. 266, pp. 132–145, 1987.

[11] Aldo Cumani, "ESP - a package for the evaluation of stochastic Petri nets with phase-type distributed transition times," in *Proceedings of International Workshop on Timed Petri Nets*, Torino, Italy, July 1–3, 1985, pp. 144–151.

[12] H. Choi, V.G. Kulkarni, and K.S. Trivedi, "Markov regenerative stochastic Petri nets," *Performance Evaluation*, vol. 20, pp. 337–357, 1994.

[13] M. Telek, A. Bobbio, and A. Puliafito, "Steady state solution of MRSPN with mixed preemption policies," in *2nd Int. Performance and Dependability Symposium*, Urbana-Champaign, IL, 1996, pp. 106–115, IEEE CS Press.

[14] R. Marie and K. S. Trivedi, "A note on the effect of preemptive policies on the stability of a priority queue," *Information Processing Letters*, vol. 24, pp. 397–401, 1987.

[15] 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*, vol. SE-15, pp. 832–846, 1989.

[16] A. Bobbio, V.G. Kulkarni, A. Puliafito, M. Telek, and K. Trivedi, "Preemptive repeat identical transitions in Markov regenerative stochastic petri nets," in *Proceedings of the 6th International Workshop on Petri Nets and Performance Models - PNPM'95*, Durham, NC, USA, October 3-6 1995, pp. 113–122.

[17] 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*, vol. 8, pp. 69–80, September 1991.

[18] 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.

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

[20] D.R. Cox, *Renewal Theory*, Methuen & Co. Ltd, London, England, 1967.

[21] William Feller, *An Introduction to Probability Theory and Its Applications, 2nd ed.*, John Wiley & Sons, New York, NY, USA, 1966.

[22] R. German and C. Lindemann, "Analysis of stochastic Petri nets by the method of supplementary variables," *Performance Evaluation*, vol. 20, no. 1–3, pp. 317–335, May 1994.

[23] S. Garg, A. Puliafito, M. Telek, and K. Trivedi, "Analysis of preventive maintenance in transactions based software systems," *IEEE Transactions on Computers*, vol. 47, pp. 96–107, January 1998.

[24] A. Bobbio and M. Telek, "Markov regenerative SPN with non-overlapping activity cycles," in *Proceedings of the 1st Annual IEEE International Computer Performance & Dependability Symposium - IPDS'95*, Erlangen, Germany, April 24-26 1995, pp. 124–133.

[25] Miklós Telek, *Some Advanced Reliability Modeling Techniques*, Ph.D. thesis, Technical University of Budapest, Departament of Telecomunications, Budapest, Hungary, 1994.

[26] Harold Davis Thayer, *The Theory of The Volterra Integral Equation of the Second Kind*, Bloomington, Indiana, USA, 1930.

[27] Reinhard German, "Transient analysis of deterministic and stochastic Petri nets by the method of supplementary variables," in *Proceedings of the 3rd International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems Conference - MASCOTS'95*, Durham, NC, USA, January 18–20, 1995, pp. 394–398.

[28] R. German, D. Logothetis, and K.S. Trivedi, "Transient analysys of Markov regenerative stochastic Petri nets: A comparison of approaches," in *Proceedings of the 6th International Workshop on Petri Nets and Performance Models*, Durham, NC, USA, October 3-6 1995, pp. 103–111.

[29] M. Telek, A. Bobbio, L. Jereb, A. Puliafito, and K.S. Trivedi, "Steady state analysis of Markov regenerative SPN with age memory

policy," in *Lecture Notes in Computer Science*, H. Beilner and F. Bause, Eds., 1995, vol. 977, pp. 165–179.

[30] H. Brunner and P.J. van der Houwen, *The Numerical Solution of Volterra Equations*, North-Holland, New York, NY, USA, 1986.

[31] Peter Linz, *Analytical and Numerical Methods for Volterra Equations*, vol. 7, SIAM Studies in Applied Mathematics, Philadelphia, PA, USA, 1985.

[32] R. Fricks, M. Telek, A. Puliafito, and K. Trivedi, "Markov renewal theory applied to performability evaluation," in *State-of-the Art in Performance Modeling and Simulation. Modeling and Simulation of Advanced Computer Systems: Applications and Systems*, Kallol Bagchi and George Zobrist, Eds., Newark, NJ, EUA, 1997, pp. 193–236, Gordon and Breach Publishers.

[33] D. Logothetis and K. Trivedi, "Dependability evaluation of the double counter-rotating ring with concentrator attachments," *ACM/IEEE Transactions on Networks*, vol. 2, no. 5, pp. 520–532, October 1994.

[34] D. Logothetis and K. Trivedi, "Time–dependent behavior of redundant systems with deterministic repair," in *Computations with Markov Chains*, W. J. Stewart, Ed., Norwell, MA, USA, 1995, pp. 135–150, Kluwer Academic Publishers.

[35] D. Logothetis and K. Trivedi, "The effect of detection and restoration times for error recovery in communication networks," *Journal of Network and Systems Management*, vol. 5, no. 2, pp. 173–195, June 1997.

[36] V. Mainkar and K. Trivedi, "Approximate analysis of priority scheduling systems using stochastic reward nets," in *Proceedings of the 13th International Conference on Distributed Computing Systems - ICDCS'93*, Pittsburgh, PA, USA, May 1993, pp. 466–473.

[37] M. Telek and A. Pfening, "Performance analysis of Markov regenerative reward models," *Performance Evaluation*, vol. 27 & 28, pp. 1–18, 1996.

[38] G. Ciardo, J. Muppala, and K. Trivedi, "Spnp: Stochastic petri net package," in *Proc. Int. Workshop on Petri Nets and Performance Models*, Los Alamitos, CA, USA, December 1989, IEEE Computer Society Press, pp. 142–150.

[39] H. 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.

[40] G. Ciardo and C. Lindemann, "Analysis of deterministic and stochastic Petri nets," in *Proceedings International Workshop on Petri Nets and Performance Models - PNPM93*. IEEE Computer Society, 1993, pp. 160–169.

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

[42] C. Lindemann and R. German, "DSPNexpress: A software package for efficiently solving deterministic and stochastic Petri nets," in *Performance Tools 1992*, Edinburgh University Press, 1992.

[43] R. German, Ch. Kelling, A. Zimmermann, and G. Hommel, "TimeNET: A toolkit for evaluating non-Markovian stochastic Petri-nets," *Performance Evaluation*, vol. 24, pp. 69–87, 1995.

[44] R. German, C. Kelling, A. Zimmermann, and G. Hommel, "TimeNET - a toolkit for evaluating non-Markovian stochastic Petri nets," in *Proceedings of the 6th International Workshop on Petri Nets and Performance Models - PNPM'95*, Durham, NC, USA, October 3–6, 1995, pp. 210–211.

[45] A. Bobbio, A. Puliafito, M. Scarpa, and M. Telek, "Webspn: A web-accessible petri net tool," in *Proceedings of the Int. Conf. on WEB based Modeling and Simulation*, San Diego, CA, USA, January 11-14 1998.

[46] C. Fröberg, *Introduction to Numerical Analysis, 2nd. ed.*, Addison-Wesley Publishing Company, Reading, MA, USA, 1969.

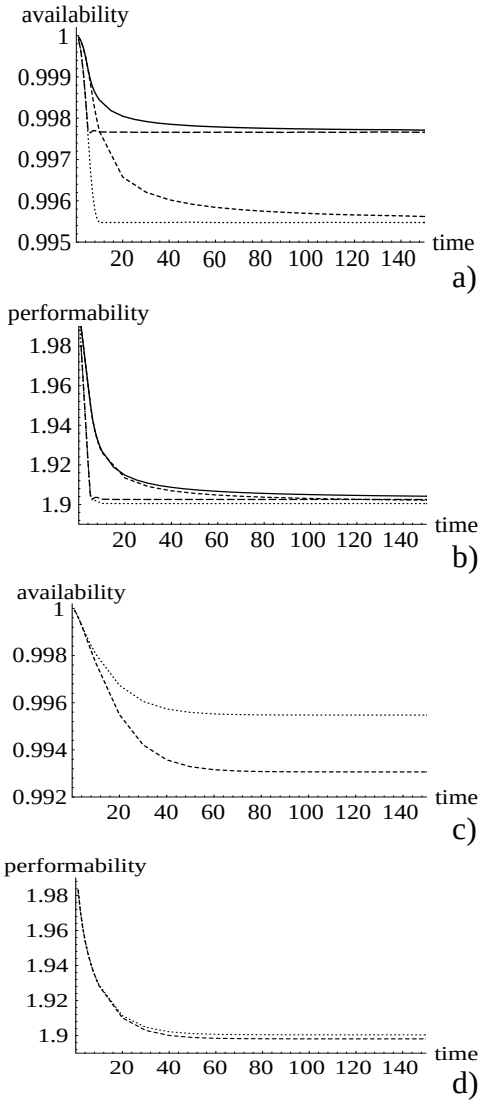

Figure 7: Availability and processing power

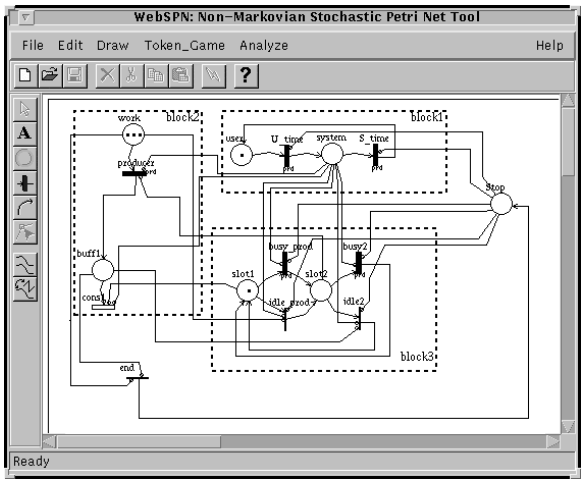

Figure 8: *Petri net* model of the system

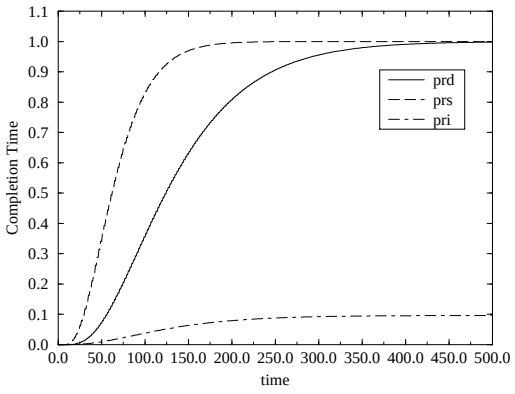

Figure 9: Distribution of completion time

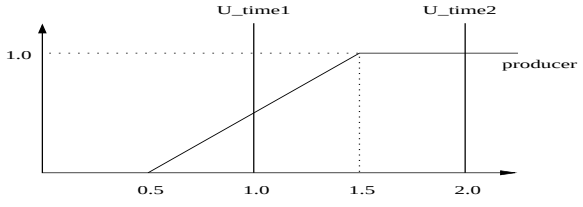

Figure 10: Firing Time Distributions of transitions *producer* and *U_time*

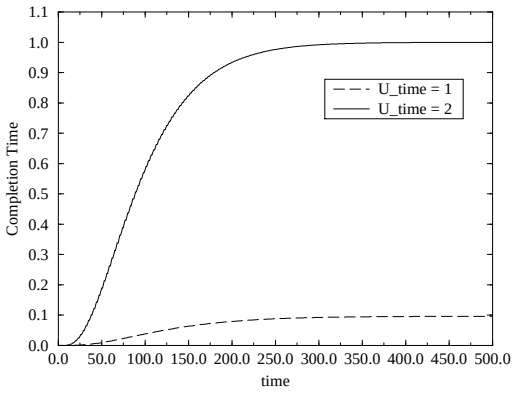

Figure 11: Distribution of completion time