Sufficient conditions for existence of a fixed point in stochastic reward net-based iterative models *

Varsha Mainkar†
AT&T Laboratories
Rm. 1K220, 101, Crawfords Corner Rd.
Holmdel, NJ 07733, USA

Kishor S. Trivedi
Center for Advanced Computing and Communication
Dept. of Electrical and Computer Engineering
Duke University, Box 90291
Durham, NC 27708-0291, USA

Abstract

Stochastic Petri Net models of large systems that are solved by generating the underlying Markov chain pose the problem of largeness of the state-space of the Markov chain. Hierarchical and iterative models of systems have been used extensively to solve this problem. A problem with models which use **fixed-point iteration** *is the theoretical proof of existence, uniqueness, and convergence of the fixed-point equations, which still remains an "art". In this paper, we establish conditions, in terms of the net structure and the characteristics of the iterated variables, under which* **existence** *of a solution is guaranteed when fixed-point iteration is used in stochastic Petri nets.*

We use these conditions to establish the existence of a fixed point for a model of a priority scheduling system, at which tasks may arrive according to a Poisson process or due to spawning or conditional branching of other tasks in the system.

1 Introduction

Stochastic Petri nets (SPN) [18], and their variants [1, 5] offer a powerful graphical capability for the specification of Markov models. With the use of tools that take an SPN specification and generate and solve the underlying continuous time Markov chain (CTMC) [6], the task of specifying and solving a model becomes greatly simplified. However, modeling complex systems using stochastic Petri nets often leads to the problem of a large underlying Markov chain. This entails the solution of a CTMC with a large number of states – sometimes so large that the CTMC infinitesimal generator cannot be stored in the memory of a fairly large-sized workstation, even with sparse storage techniques. This problem can be solved by using many approaches. A technique commonly used in reliability modeling is that of *state truncation* [11]. In this technique, states that are highly unlikely (e.g., states with many

*This research was sponsored by IBM under the IBM/Duke University Research Agreement # RAL-R93010-00.

†This work was done while the author was at Duke University.

failed components in an ultra-reliable redundant system) are not generated, thus saving state space. Another approach is that of *hierarchical modeling* [2, 23, 29]. Many systems are hierarchically built; this naturally translates into a hierarchical model. In this technique, a system is solved by identifying subsystems which can be modeled in isolation, and then aggregating their results into (often, but not always, approximate) results for a higher level model and so on. A hierarchical model results in a substantial reduction of state space and solution time. In many cases, however, the model cannot be decomposed very "cleanly", i.e., there are interactions between submodels that cannot be ordered, and thus are not strictly hierarchical. In such cases, *fixed-point iteration* is used to determine those model parameters that are not available directly as input or by solving other models [3, 4, 8, 26]. In this technique, the relationships between model parameters and model outputs result in an equation of the type

$$\mathbf{x} = \mathbf{G}(\mathbf{x}) \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_n)$ is the vector of iteration variables. This is the *fixed-point equation* corresponding to the iterative model, and the vector $\mathbf{x}$ that satisfies this equation is called a *fixed point* of this equation. The simplest way of finding this fixed point is by *successive substitution*. In this method, starting with an initial guess $\mathbf{x}_0$, we iterate in the following way:

$$\mathbf{x}_n = \mathbf{G}(\mathbf{x}_{n-1}).$$

This iteration is terminated when the difference between two successive iterates is below a certain tolerance level. Note that this iteration may not always converge. If it converges, it may not always converge to the same value. Also, before we use the iterative method, we must be sure that *a solution to the Equation (1) exists.*

In this paper, we focus on one of these theoretical problems that arise while using iteration with stochastic Petri nets: specifically, *stochastic reward nets (SRNs)*. When using an iterative model it is extremely desirable to be able to prove the existence and uniqueness of a fixed point and convergence of the iterative method to that point. In most cases only existence can be proven; uniqueness has also been proven in a few instances [5, 12]. Very often, the rate of convergence remains an issue of empirical judgment. Further, existence may also be proven only under certain conditions [17, 26], and so far this has been done on a case-by-case basis for each specific SRN model. In this paper, we describe general sufficient conditions in terms of the SRN and its underlying reachability graph that guarantee the existence of a solution to a fixed-point equation that is a result of iteration between some SRNs. In [5], some pointers to proofs of existence were first provided, and general guidelines for *how* an SRN can be decomposed were provided. Our contributions with respect to previous work are as follows :

- We provide a *general* set of conditions that can be used to verify existence of a fixed point for any SRN iteration model. This is different from previous work [4, 26], where proofs were provided, but only for the particular example being modeled.
- In [5], some suggestions on possible approaches for proof of existence were discussed, but definitive conditions on the SRN were not provided. In this paper, we provide

necessary conditions on the SRN in such a way that once those conditions are met, no further delving into the details of existence theorems is required.

- We provide, for the first time, a proof of continuity of the iteration function, which is required, but was simply assumed in the previous papers.

However, in this paper we do not address the issue of how an SRN can be decomposed, but once a decomposition has been proposed, we present comprehensive sufficient conditions under which existence of a fixed point can be proved. These conditions may be used as a first "check" when one is using an iterative model, to confirm the existence of a solution. Note that the conditions are only sufficient, not necessary. For a thorough and formal approach on *how* to decompose SRN models, see [8].

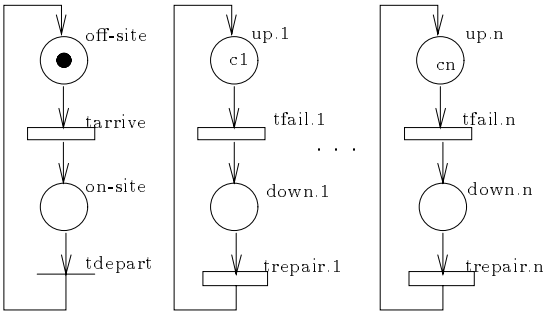
The remainder of the paper is organized as follows. In Section 2 we describe an example of an iterative model to set the background and provide some motivation for the problem; in Section 3, we describe sufficient conditions required for the existence of a solution; in Section 4 we present examples. In Section 5 we present an application of iterative modeling using SRNs for analysis of a priority scheduling system. We conclude the paper in Section 6. A brief overview of stochastic reward nets may be found in Appendix A.

2 Motivating Example

Place	Description	Initial # of tokens
off-site	Repairman is off site	1
on-site	Repairman is on site	0
up.i	Number of operational components in subsystem i	c_i
down.i	Number of components failed in subsystem i	0
Transition		**Rate or Probability**
tfail.i	Time to failure of a component of subsystem i	(#(up.i)) λ_i
trepair.i	Time to repair of a component of subsystem i	μ_i
tarrive	Time of arrival of repairman to the site	μ_t
tdepart	Repairman departs	1

Table 1: Description of the SRN for availability

To motivate the need for the work described in this paper we use an example in availability modeling, which was presented by Tomek and Trivedi in [26]. Availability at time t, of a system which is subject to failure and repair is defined as the probability that the system is operational at time t. The steady-state probability of the system being operational



Transition	Guard
tarrive	$\sum_{i=1}^{n} \#(\text{down.i}) > 0$
tdepart	$\sum_{i=1}^{n} \#(\text{down.i}) = 0$
trepair.i	$\sum_{j=1}^{i-1} \#(\text{down.j}) = 0$ and $\#(\text{on-site}) = 1$

Figure 1: SRN Model for Availability

is termed steady-state availability. Tomek and Trivedi have described a steady-state availability model which accounts for the situation that the repairperson is not on site when a failure occurs, and hence there is a need to model the added travel time. The model consists of a system with n subsystems; subsystem i has c_i components. There is only one repairperson to be shared among all the subsystems and components. When a subsystem fails, it may take some time for the repairperson to arrive at the site. The repairperson's service depends on the priority of the subsystem. Subsystems are ordered by priority according to their importance. Thus, if components of subsystem i and subsystem j both fail, and $i < j$, the repairperson first goes to subsystem i. Further, subsystem i has *preemptive* priority over subsystem j. Suppose that the mean time to failure of components of subsystem i is $1/\lambda_i$, the mean repair time is $1/\mu_i$, and the mean travel time is $1/\mu_t$. The SRN representing this system as described in [26] is reproduced in Figure 1 (we assume all transition firing times are exponentially distributed). Table 1 describes the places and transitions. The first column contains the names of the places and transitions. The second column is a short description, and the third column is the initial number of tokens in case of places and the rate/probability in case of transitions. The guards corresponding to some transitions are tabulated in Figure 1. Note that even though graphically the SRNs look "independent" they are related through these guards.

For systems with large number of subsystems and subsystem components, this SRN will result in a very large underlying Markov reward model. Thus a decomposition approach was proposed in [26] wherein each subsystem is considered in isolation and the effects of the other subsystems are incorporated into transition rates and probabilities. Let q_{i-1} be the probability that the repairperson is on site repairing components of any one of subsystems $1, \ldots, i-1$. Further, let r_i denote the probability that the repairperson is busy repairing a component of subsystem i. Thus

$$q_{i-1} = \sum_{j=1}^{i-1} r_j .$$

Figure 2 describes the SRN representing the isolated subsystem i. Table 2 describes the places and transitions of the SRN. Note that the repair rate is now given by $\mu_i(1 - q_{i-1})$; i.e., it is slowed down by a factor $(1 - q_{i-1})$ corresponding to the probability that no subsystem of higher priority is being repaired. This incorporates the effect of the preemptive priority on repair exercised by higher priority subsystems. Also, the immediate transition *Toff-site.i* has a probability $\prod_{j \neq i}(1 - r_j)$, which is the probability that the repairperson is off site (no other subsystem component is being repaired). Note that this transition has probability 0 if the repairperson is already on site repairing a component of subsystem i. This is implemented by an inhibitor arc from place $Prepair.i$ to *Toff-site.i*. Note that either $\#(Ptravel.i)$ or $\#(Prepair.i)$ represents the number of failed components. Thus, $\#(Ptravel.i) \times \#(Prepair.i) = 0$. The inhibitor arc from place $Ptravel.i$ to *Ton-site.i* represents that the repairperson cannot be on-site if in transit. The arcs from $Ptravel.i$ to $Ttravel.i$ and from $Travel.i$ to $Prepair.i$ are variable multiplicity arcs. Their multiplicity is defined to be $\#(Ptravel.i)$, if $\#(Ptravel.i) > 0$ and 1 otherwise.

In this decomposition, the model for subsystem i, M_i, is parameterized by $r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_n$. Further, an output from model M_i, i.e., r_i, is used to param-

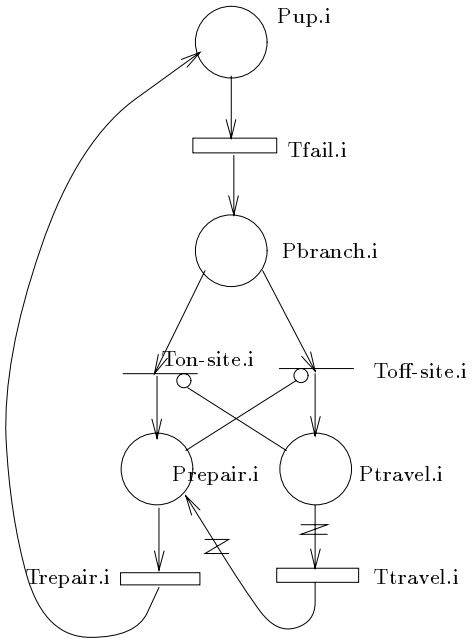


Figure 2: Approximate model of isolated subsystem

Place	Description	Initial # of tokens
Pup.i	# of operational components	c_i
Pbranch.i	Decision place	0
Prepair.i	# of failed components	0
Ptravel.i	Repairman traveling	0
Transition		Rate or Probability
Tfail.i	Component failures	#(up.i) $\times$ λ_i
Toff-site.i	Repairman off site	$\prod_{j \neq i}(1 - r_j)$
Ton-site.i	Repairman on site	$1 - \prod_{j \neq i}(1 - r_j)$
Trepair.i	Repair time	$\mu_i(1 - q_{i-1})$
Ttravel.i	Travel time	μ_t

Table 2: Description of Approximate SRN model for Availability

eterize all the other models. Thus all the models are interdependent, and a fixed-point iteration is necessary to solve the models. In steady state let $\pi^i(k)$ denote the probability that there are k tokens in place $Prepair.i$. Note that this probability is a function of the input parameters $r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_n$. Then the probability r_i is given by

$$\begin{aligned} r_i &= \sum_{k=1}^{c_i} \pi^i(k) \\ &= g_i(r_1, \ldots, r_{i-1}, r_{i+1}, \ldots r_n), \quad i = 1, \ldots, n. \end{aligned} \tag{2}$$

If $\mathbf{r}$ denotes the vector $(r_1, r_2, \ldots, r_n)$, then the above set of equations can be written in as

$$\mathbf{r} = \mathbf{G}(\mathbf{r}) \tag{3}$$

where $\mathbf{G}(\mathbf{r}) = (g_1(\mathbf{r}), \ldots, g_n(\mathbf{r}))$. This is the fixed-point equation corresponding to the iterative model. The fixed point can be computed numerically by successive substitution.

The use of decomposition for this example resulted in enormous savings in computation time. The data provided in [26] shows that when the number of subsystems is 8, the exact solution takes 220 seconds, while the approximate solution takes about 10 seconds. With 9 subsystems, the exact solution took 650 seconds, while the approximate solution took about 12 seconds. The exact solution could not be obtained for more that 9 subsystems due to memory constraints. The approximation error was less than 10^{-8}.

This example shows that there is a lot of practical benefit to be gained by using the technique of decomposition and iteration. The technique would, however, stand on more solid ground if it were theoretically supported. To do this we must prove the existence, convergence and uniqueness of the solution of this fixed-point equation, and quantify the accuracy of the solution. The existence of a solution was proved in [26], using **Brouwer's fixed-point theorem** for the case of this particular example. However, this example belongs to a more general set of iterative models which satisfy certain conditions from which existence of a solution may be directly inferred. In the following sections we will explain these conditions, and provide a proof of existence based on these conditions.

Note that we do not suggest methods or guidelines for iterative decomposition, but only outline sufficient conditions under which a fixed point will exist. For work regarding guidelines for decomposition see [8, 28]. We also do not address the uniqueness and convergence issues in this paper; they are nonetheless very important, and are topics for further research.

3 Conditions for existence of a fixed point

In this section, we shall first give a formal definition of what we term as "Iterative SRN models", for establishing the framework in which we shall be providing our conditions (Section 3.1). Then, we discuss the continuity properties of the steady-state probability vector of an SRN, as this is required for the proof of the sufficient conditions (Section 3.2). In Section 3.3, we give a formal statement of the conditions and a proof.

3.1 Preliminaries

We shall first define what we consider as iteration variables, and what the fixed-point equation is:

Definition 1 (Iterative SRN models) *Suppose a system is being modeled using n stochastic reward nets labeled $M_1, M_2, \ldots, M_n$. Let π^i denote the steady state probability vector corresponding to M_i. Suppose m_i measures, $x_1^i, x_2^i, \ldots, x_{m_i}^i$ are derived from the steady-state solution of M_i; i.e., x_j^i is a function f_j^i, of π^i, $\forall\ j = 1, \ldots, m_i$. Here $f_j^i : (\mathbb{R}^+ \cup \{0\})^{|\Omega_i|} \rightarrow \mathbb{R}^+ \cup \{0\}$, where Ω_i is the state space of the CTMC underlying the SRN M_i. Now suppose for each i, some of the transition firing rates/probabilities of the model M_i are functions of one or more $x_l^k, k = 1, \ldots n; l = 1, \ldots, m_k$. Since π^i is a function of the rates/probabilities of the SRN M_i,*

$$\begin{aligned} x_j^i &= f_j^i(\pi^i(x_1^1, \ldots, x_{m_n}^n)) && (4)\\ &= g_j^i(x_1^1, \ldots, x_{m_n}^n) \qquad j = 1, \ldots, m_i. && (5) \end{aligned}$$

Let $\mathbf{x} = (x_1^1, \ldots, x_{m_n}^n)$. Let the function $\mathbf{G} : \mathbb{R}^N \rightarrow \mathbb{R}^N$, where $N = \sum_{i=1}^n m_i$ be defined by $\mathbf{G}(\mathbf{x}) = (g_1^1(\mathbf{x}), g_2^1(\mathbf{x}), \ldots, g_{m_n}^n(\mathbf{x}))$. Then the above equation may be written in vector form as:

$$\mathbf{x} = \mathbf{G}(\mathbf{x}). \qquad (6)$$

This is the fixed-point equation *corresponding to the iterative model. We term x_j^i as the* iteration variables *and* $\mathbf{x}$ *as the* iteration vector *corresponding to the iterative SRN model.*

Note that in this process, the iteration variables are the x_j^i's. However, an *implicit* iteration will be carried out on the steady-state probability vector of the two SRNs. The definition of the fixed-point equation could as well have been made in terms of the steady-state probability vectors themselves. However, the definition in terms of measures that aggregate these probabilities is more natural, and corresponds more to the way in which a modeler would most likely implement the iteration process. Since the primary motivation of this work is simplifying the proof of existence, we have chosen to define our fixed-point equation in terms of these measures, which are functions of the steady-state probability. However, we would like to stress the point that one of the key properties that will be exploited in building sufficient conditions, is that the iteration variables depend on each other only *through* the steady-state probabilities. This will be explained more clearly in the following sections.

3.2 Background

The sufficient conditions under which Equation (6) has a fixed point are proven using the following theorem:

Theorem 1 (Brouwer's fixed-point theorem [21]) *Let $\mathbf{G} : S \subset \mathbb{R}^N \rightarrow \mathbb{R}^N$ be continuous on the compact, convex set S, and suppose that $\mathbf{G}(S) \subseteq S$, where $\mathbf{G}(S)$ stands for $\cup_{x \in S}\{\mathbf{G}(x)\}$. Then $\mathbf{G}$ has a fixed point in S.*

The conditions that we will outline in the Section 3.3 are such that if the iteration function $\mathbf{G}$ satisfies those conditions, it also satisfies the conditions of the Brouwer's fixed-point theorem. In our proof of the sufficiency of those conditions, therefore we must prove each property of $\mathbf{G}$ as outlined in the above theorem. Note that among other properties, the continuity of the function $\mathbf{G}$ is required. From Equation (5), it is clear that continuity of the function $\mathbf{G}$ depends on the continuity of the steady-state probability vector π. Therefore, as a background for the theorem and its proof, we will first establish the continuity of the steady-state probability vector of an SRN under certain conditions. The steady-state probability is a function of the entries of $\mathbf{Q}$, the infinitesimal generator matrix corresponding to that SRN. In the next section, we shall discuss the continuity of this function π.

3.2.1 Continuity of the steady-state vector

The infinitesimal generator matrix $\mathbf{Q}$, that corresponds to an SRN that is a part of an iterative model, has some entries that are variable. Consider first the case of continuity at values of these variables at which $\mathbf{Q}$ remains irreducible. For this case, the steady-state vector π is given by

$$\pi\mathbf{Q} = \mathbf{0}, \qquad \sum_i \pi_i = 1.$$

The above equation can be transformed into a discrete-time Markov chain (DTMC) steady-state equation by defining $\mathbf{P} = \mathbf{Q}/q + \mathbf{I}$, where $q > \max_i\{-Q_{ii}\}$. The DTMC constructed in this way is aperiodic, and has a unique steady-state vector. This DTMC vector is equal to the steady-state vector corresponding to the CTMC with generator matrix $\mathbf{Q}$. Let $\mathbf{P}^\infty = \lim_{n\to\infty} \mathbf{P}^n$. For an aperiodic, irreducible, finite DTMC this limit is unique, all the rows of $\mathbf{P}^\infty$ are equal, and are equal to the steady-state vector of the DTMC [15]. Now, define

$$\mathbf{B} = \text{adj}(\mathbf{I} - \mathbf{P}),$$

where $\text{adj}(\mathbf{A})$ is the *adjoint* of the matrix $\mathbf{A}$. Let $\mathbf{M}_{ij}$ be the matrix obtained by deleting the ith row and the jth column of matrix $\mathbf{A}$. Let $\det(\mathbf{A})$ denote the determinant of matrix $\mathbf{A}$. Then the (i,j)th entry of the adjoint matrix is given by $(-1)^{(i+j)}\det(\mathbf{M}_{ji})$. Furthermore, suppose $c(\gamma)$ is the characteristic polynomial of $\mathbf{P}$, i.e., $c(\gamma) = \det(\gamma\mathbf{I} - \mathbf{P})$. Let $c^{(1)}(\gamma)$ denote its first derivative. Then $c^{(1)}(1)$ is the value of this derivative function at 1. Then $\mathbf{P}^\infty$ is given by [16]

$$\mathbf{P}^\infty = \frac{\mathbf{B}}{c^{(1)}(1)}.$$

Since the entries of the adjoint matrix, $\mathbf{B}$, are determinants, they are polynomials in terms of the entries of matrix $\mathbf{Q}$. Thus its entries are continuous functions of the entries of matrix $\mathbf{Q}$. The operation of division is also continuous[1], hence entries of the matrix $\mathbf{P}^\infty$ are continuous functions of the entries of matrix $\mathbf{Q}$. It follows then, that π is a continuous function of the entries of matrix $\mathbf{Q}$.

[1] The division operation is continuous if the denominator is non-zero. Since this definition of $\mathbf{P}^\infty$ is made only for aperiodic, irreducible DTMC's, we know that $\mathbf{P}^\infty$ exists and is unique. Thus, in this case the denominator cannot be zero, and the operation is continuous.

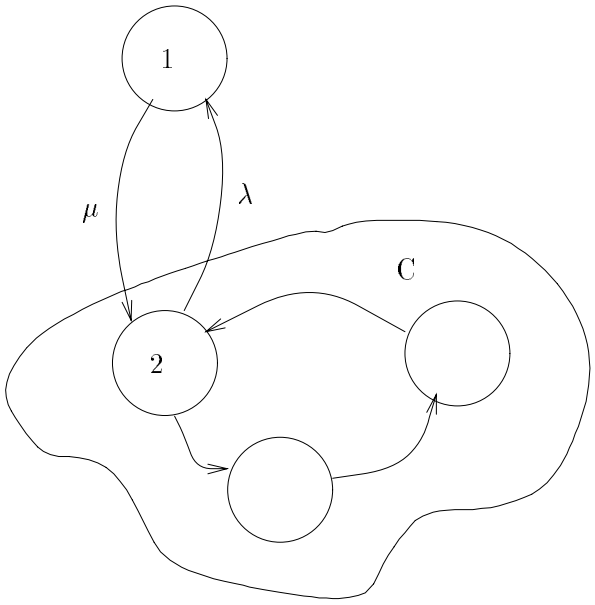


Figure 3: A CTMC with one communicating class

Consider now, the case of continuity at points which affect the connectivity of the states of the CTMC. For example, consider an irreducible CTMC with a state space Ω. Let the structure of the CTMC be as shown in Figure 3. Thus $\Omega = \{1\} \cup C$, where C is a communicating class[2]. Consider the simple case when only λ is the variable and all other transition rates are constant. Such a CTMC will correspond to a $\mathbf{Q}$ matrix as follows:

$$\mathbf{Q}(\lambda) = \begin{bmatrix} -\mu & \mu & \mathbf{0} \\ \lambda & \mathbf{Q}_C(\lambda) & \\ \mathbf{0}^T & & \end{bmatrix},$$

where $\mathbf{Q}_C$ is the $(|\Omega|-1)\times(|\Omega|-1)$ matrix restricted to the communicating class C depicted in Figure 3 and $\mathbf{0}$ denotes a row vector of zeroes. Note that even though all other transition rates are constant, $\mathbf{Q}_C$ is a function of λ, because $Q_{2,2}$ is defined as $-\sum_{j\in\Omega, j\neq 2} Q_{2,j}$. Since $Q_{2,1} = \lambda$, this entry of the sub-matrix $\mathbf{Q}_C$ is a function of λ.

The steady-state solution, π, of this CTMC is given by solving the following system of linear equations:

$$\pi(\lambda)\mathbf{Q}(\lambda) = \mathbf{0}, \qquad \sum_i \pi_i = 1. \tag{7}$$

[2]A subset of states in a CTMC is called a *communicating class* if all the states in this subset are reachable from each other. A communicating class C is *closed* if the states outside class C are not reachable from states in class C.

Thus π is also a function of λ. At $\lambda > 0$, the continuity of this function is guaranteed by the previously addressed case. However at $\lambda = 0$, the CTMC is no longer irreducible, and the continuity needs to be verified. Thus we need to prove that

$$\lim_{\lambda \to 0} \pi(\lambda) = \pi(0).$$

We would like to properly define $\pi(0)$. At $\lambda = 0$, the CTMC has one communicating class C. Let T denote the set of transient states. Let $X(t)$ denote the state at time t.

For reducible CTMCs, with one transient class T, and sets of closed communicating classes $C_1, \ldots C_k$, the steady state is defined as follows [15]: let for $i \in T$ and $1 \leq r \leq k$,

$$\alpha_i(r) = P[X(t) \in C_r \text{ for some } t \geq 0 \mid X(0) = i].$$

Further for $j \in C_r$, define the conditional probability,

$$\pi_j^r = \lim_{t \to \infty} P[X(t) = j \mid X(0) \in C_r].$$

Then, for $i \in T$ and $j \in C_r$, the steady state probability of being in state j given that the initial state was i is given by [15]:

$$\lim_{t \to \infty} P[X(t) = j \mid X(0) = i] = \alpha_i(r)\, \pi_j^r.$$

The conditional steady state solution vector $\pi^r = [\pi_j^r]$ $(j \in C_r)$ is computed by solving:

$$\pi^r \mathbf{Q}_{C_r} = \mathbf{0}, \qquad \sum_{j \in C_r} \pi_j^r = 1. \tag{8}$$

where $\mathbf{Q}_{C_r}$ is the generator matrix restricted to class C_r. Also, for $j \in T$, $\lim_{t \to \infty} P[X(t) = j] = 0$.

In our case, we have only one closed communicating class $C_1 = C$. Also, we have $T = \{1\}$. Thus it is obvious that $\alpha_i(1) = 1, \ \forall i \in T$. Thus for $j \in C$, the unconditional steady-state probability π_j is given by $\pi_j = \pi_j^1$. Also, $\pi_1 = 0$. Let us denote the restriction of the steady-state probability vector to class C, by π_C. Thus, $\pi_C = \pi^1$, where π^1 is given by Equation (8).

Thus $\pi(\lambda)$ is given by Equation (7), when $\lambda > 0$ and by $(0, \pi_C)$ when $\lambda = 0$. Given this definition we shall now check $\pi(\lambda)$ for continuity at 0.

Equation (7) can be written as

$$\begin{aligned}
-\mu\pi_1 + \lambda\pi_2 &= 0 && (9)\\
\mu\pi_1 + \pi_C \mathbf{Q}_C^1(\lambda) &= 0 && (10)\\
\pi_C \mathbf{Q}_C^i(\lambda) &= \mathbf{0} \quad i \geq 2 && (11)\\
\sum_i \pi_i &= 1 && (12)
\end{aligned}$$

where $\mathbf{Q}_C^i$ denotes the ith column of the matrix $\mathbf{Q}_C$.

From Equation (9), we can see that the limit of π_1 as λ tends to zero, is 0. Substituting $\pi_1 = 0$ in Equations (10) , (11) and, (12) yields

$$\pi_C \mathbf{Q}_C(\lambda) = \mathbf{0}, \qquad \sum_{j \in C} \pi_{Cj} = 1,$$

which gives as defined before (Equation (8)), the solution of the CTMC when $\lambda = 0$. Thus the steady-state solution vector $\pi(\lambda)$, is continuous at $\lambda = 0$ in this case. Note that the continuity depended on the fact that the number of closed communicating classes in the CTMC was not more than one at $\lambda = 0$. If there were more than one communicating class, the steady state vector would not have been uniquely defined, and the function would not have been continuous. Using an argument similar to the one above, we can conclude that :

> The steady-state vector of a CTMC as a function of some non-zero entries $\lambda_1, \lambda_2, \ldots, \lambda_k$ of the $\mathbf{Q}$ matrix, is continuous at all values of $\lambda_i \geq 0, i = 1, 2, \ldots, k$ if for all values of $\lambda_i \geq 0, i = 1, 2, \ldots, k$, the CTMC has exactly one closed communicating class.

As an example of a case when π is not continuous, consider again the CTMC as shown in Figure 3. In this case, if we consider λ and μ both as variables, then $\pi(\lambda, \mu)$ is not continuous at (0,0), since at this point, the CTMC is reducible with two disjoint closed communicating classes. The steady-state vector will now depend on the initial state of the CTMC, and $\pi(0, 0)$ will not be uniquely defined.

3.3 Statement of Conditions

The conditions for existence of a fixed point for Equation (6) are now stated as follows:

Theorem 2 *Consider a set of interdependent SRNs as described in Definition 1 with a fixed-point equation given by Equation (6). Suppose f_j^i can be expressed as one of the following:*

1. $f_j^i = f_1 + f_2$,
2. $f_j^i = f_1 \times f_2$,
3. $f_j^i = (f_1)^a$,
4. $f_j^i = c \times f_1$,
5. $f_j^i = f_1 + c$,
6. $f_j^i = \pi_k^i$,

where a, c are **constants** *and $a, c \in \mathbb{R}^+$. f_1 and f_2 are non-negative real-valued functions and f_1 and f_2 can also be expressed in one of the ways described above.*

Then, $\forall\ i = 1, \ldots n;\ j = 1, \ldots, m_i$, we can find $l_j^i, u_j^i \in \mathbb{R}^+ \cup \{0\}$ such that $x_j^i \in [l_j^i, u_j^i]$, whenever x_j^i exists and is well-defined. Suppose the underlying CTMC corresponding to

each M_i is such that it has exactly one closed communicating class for all values of $x_j^i \in [l_j^i, u_j^i]$, $\forall\ i = 1, \ldots, n;\ j = 1, \ldots, m_i$. Then, a fixed point x_j^i, corresponding to this equation exists in $[l_j^i, u_j^i]$.

Proof

We prove this result by establishing that the function $\mathbf{G}$ satisfies the conditions required in Brouwer's fixed-point theorem. Recall that $\mathbf{G}$ is defined as

$$\begin{aligned} \mathbf{G}(\mathbf{x}) &= (g_1^1(\mathbf{x}), \ldots, g_{m_n}^m(\mathbf{x})) && (13) \\ &= (f_1^1(\pi^1(\mathbf{x})), \ldots, f_{m_n}^m(\pi^m(\mathbf{x}))) && (14) \end{aligned}$$

1. *The Set S*: We must first identify the set S for which we can show that $\mathbf{G}(S) \subseteq S$. This is done by showing that if f_j^i is defined at some point $\mathbf{y} \in (\mathbb{R}^+ \cup \{0\})^{|\Omega_i|}$, then $f_j^i(\mathbf{y}) \in [l_j^i,\ u_j^i]$ for some $l_j^i, u_j^i \in \mathbb{R}^+ \cup \{0\}$. It is at this point that we shall be making use of the restricted definition of f_j^i, and the fact that it is a defined in terms of π^i.

Since $f_j^i(\mathbf{y})$ is defined recursively, we shall prove this by *structural induction*. We first consider the base case of $f_j^i = \pi_k^j$. Then $f_j^i(\mathbf{y}) \in [0, 1]$. This forms the basis of our induction. Now, suppose $f_1(\mathbf{y}) \in [l_1, u_1]$ and $f_2(\mathbf{y}) \in [l_2, u_2]$ where $l_1, u_1, l_2, u_2 \in \mathbb{R}^+ \cup \{0\}$. Then if

1. $f_j^i(\mathbf{y}) = f_1(\mathbf{y}) + f_2(\mathbf{y})$, then $f_j^i(\mathbf{y}) \in [l_1 + l_2, u_1 + u_2]$.
2. $f_j^i(\mathbf{y}) = f_1(\mathbf{y}) \times f_2(\mathbf{y})$, then $f_j^i(\mathbf{y}) \in [l_1 \times l_2, u_1 \times u_2]$.
3. $f_j^i(\mathbf{y}) = (f_1)^a(\mathbf{y})$, then $f_j^i(\mathbf{y}) \in [(l_1)^a, (u_1)^a]$.
4. $f_j^i(\mathbf{y}) = c \times f_1(\mathbf{y})$, then $f_j^i(\mathbf{y}) \in [cl_1, cu_1]$.
5. $f_j^i(\mathbf{y}) = c + f_1(\mathbf{y})$, then $f_j^i(\mathbf{y}) \in [c + l_1, c + u_1]$.

Thus, by induction, $f_j^i \in [l_j^i,\ u_j^i]$, where $l_j^i,\ u_j^i$ is defined by one of the sets above. From Equation (14), $g_j^i(\mathbf{x}) = f_j^i(\pi^i(\mathbf{x}))$, therefore, $g_j^i(\mathbf{x})$ is also in $[l_j^i,\ u_j^i]$. Since $x_j^i = g_j^i(\mathbf{x})$, this implies that $x_j^i \in [l_j^i,\ u_j^i]$.

Now, since we assume (according to the theorem) that for all $x_j^i \in [l_j^i,\ u_j^i]$, the underlying CTMC of every SRN has only one communicating class, the steady-state solution π_i will be unique and well-defined whenever $x_j^i \in [l_j^i, u_j^i]$ (from Section 3.2.1), for all $i = 1, \ldots, m$ and $j = 1, \ldots, m_n$. From the definition of f_j^i it is clear that if π_j^i exists, f_j^i exists, which implies that g_j^i exists . Therefore $g_j^i(\mathbf{x})$ is defined whenever $x_j^i \in [l_j^i, u_j^i], \forall i = 1, \ldots, m$ and $j = 1, \ldots, m_n$.

Define $S = [l_1^1, u_1^1] \times \ldots \times [l_{m_n}^n, u_{m_n}^n]$. Then, from the preceding discussion, $\mathbf{x} = \mathbf{G}(\mathbf{x}) \in S$, whenever $\mathbf{G}$ exists, and $\mathbf{G}(\mathbf{x})$ exists whenever $\mathbf{x} \in S$. That is :

$$\mathbf{G}(S) \subseteq S.$$

2. *The compactness and closedness of S*: Since S is closed, it is compact [10]. S is also convex because any interval on the real line is a convex set (Example 3, Section 1-4 in [10]) and convexity is preserved under the operation of cartesian product (Prop. 1.2.3, Chap III, Sec 1.2 in [14]).

3. *Continuity of G:* $\mathbf{G}$ is continuous in S, if each f_j^i is continuous in the set $\{\pi^i(\mathbf{x})|\mathbf{x} \in S\}$. When $\mathbf{x}$ is in$S$, we have assumed that the underlying CTMC of each SRN has a single closed communicating class. Therefore, from the discussion of Section 3.2, the steady-state vector, π^i, is continuous in S, $\forall i = 1, \ldots, m$.

The operations in (1)–(6) that are involved in the definition of f_j^i are also all continuous. Therefore, $f_j^i(\pi^i)$ is continuous in $[0, 1]$. Therefore it is also continuous in $\{\pi^i(\mathbf{x})|\mathbf{x} \in S\}$.

Therefore, we can conclude that $\mathbf{G}$ is continuous in S.

Thus, $\mathbf{G}$ satisfies all the conditions of the Brouwer's fixed-point theorem, and a fixed point $\mathbf{x} \in S$ exists such that $\mathbf{x} = \mathbf{G}(\mathbf{x})$.

Corollary : It follows from the above theorem that if the variables x_j^i are *expected reward rates* and the underlying CTMC of each SRN model M_i has exactly one closed communicating class for all values of x_j^i, a fixed point will exist. The constructive method outlined in the proof above can identify the set in which a fixed point will be found.

This corollary is especially useful with the use of SRN tools. In other words, if an iterative SRN model is being specified through an SRN tool and the variables being iterated upon are expected reward rates (specified through the tool), then the second condition of one closed communicating class can be verified (also using a SRN tool such as SPNP [7]), and existence of a fixed point can be confirmed.

4 Examples

In this section we show how the sufficient conditions make proof of existence of a fixed point in SRN-based iterative models easier. The first example is the motivating example from Section 2. In this case, the proof of existence was given in the original paper, here we show that it is a special case of our general conditions [26]. The next example is a trivial example to show a case where the theorem conditions are not satisfied, and hence existence could not be proven.

4.1 Availability Modeling

Equation (3) from Section 2 can be proved to have a fixed point using Theorem 2. The function $r_i = \sum_{k=1}^{c_i} \pi^i(k)$ obviously satisfies the conditions of the theorem. Now, at some values of $r_j, j = 1, \ldots, n$, in $[0, 1]$ the probability of either transition *Toff-site.i* or *Ton-site.i* is zero. Further, the rate of transition $Trepair$ may also evaluate to zero. However, it can be verified (using, for instance, a SRN to CTMC mapping tool such as SPNP [7]) that at these and all other values of $r_j \in [0, 1]$, the SRN gives rise to a CTMC with only one closed communicating class. Thus a fixed point exists for this equation in the interval $[0, 1]$.

4.2 Example : Trivial Pair of SRNs

Now we show an example where conditions of the theorem are not met, and hence existence of a fixed point cannot not be proven using this theorem.

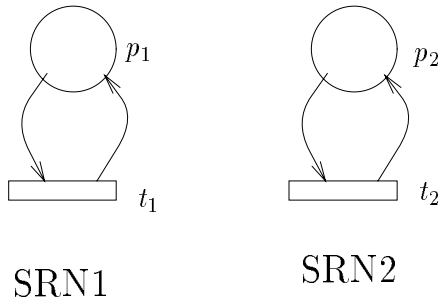


Figure 4: Example where the conditions are not satisfied

Consider a trivial pair of SRNs as shown in Figure 4. Let μ_1 and μ_2 denote the firing rates of transitions t_1 and t_2 respectively. Let the measures derived from each of these SRNs be the throughputs of transitions t_1 and t_2 respectively. Denote these by x_1^1 and x_1^2. Then :

$$x_1^1 = \pi_{p_1}^1 \mu_1 = \mu_1, \tag{15}$$

$$x_1^2 = \pi_{p_2}^2 \mu_2 = \mu_2, \tag{16}$$

where $\pi_{p_1}^1$ is the probability of a token being in place p_1, and $\pi_{p_2}^2$ is the probability of a token being in place p_2. For the SRNs shown, it is clear that both these probabilities are equal to 1.

Now suppose that μ_1 depends on x_1^2 as follows :

$$\mu_1 = 2 \times x_1^2. \tag{17}$$

Also, let μ_2 depend on x_1^1 as follows :

$$\mu_2 = 2 \times x_1^1. \tag{18}$$

Then from Equation (15), (16), (17) and (18), the fixed-point equation can be written as :

$$\begin{aligned} x_1^1 &= 2x_1^2 \\ x_1^2 &= 2x_1^1 \end{aligned} \tag{19}$$

The functions f_j^i of Theorem 2 are given here in Equation (15) and (16) and for $i = 1, 2$ and $j = 1$. These functions do not satisfy the requirements of Theorem 2. Though it may seem like f_j^i can be expressed using Rules (4) and (6) of the theorem, that is incorrect,

because in Rule (4), the multiplier c must be a *constant*. In Equations (15) and (16), the multipliers μ_1 and μ_2 are unknowns, and hence violate this rule. Thus Theorem 2 cannot be used to prove the existence of a fixed point for Equation (19).

Note that in the example, there is a trivial fixed point, which is $(x_1^1, x_1^2) = (0, 0)$. However, the conditions of Theorem 2, which are only sufficient and not necessary, could not be used to prove the existence of this trivial fixed point.

5 Approximate analysis of priority scheduling systems

In this section[3] we present a performance analysis of a heterogeneous multiprocessor system which uses priority discipline to schedule its tasks. The tasks may arrive to this system from a Poisson source or due to spawning or conditional branching by other tasks in the system. We model this system using SRNs and apply decomposition and iteration to solve the model. We shall again see that Theorem 2 can be used to establish the existence of a fixed point for the iterative model. This section is divided into the following subsections : Section 5.1 provides a brief background and motivation. In Section 5.2 we give a precise description of the system we are analyzing, in Section 5.3 we give an example of the system and describe the SRN model for this system. In Section 5.4 we show how the example model could be decomposed into two iterative SRNs. Section 5.5 shows how the existence of a fixed point can be proven using Theorem 2, and Section 5.6 describes results showing the savings in state space, and the accuracy of the approximation.

5.1 Background

Priority queues with tasks that could feedback into the system through spawning and conditional branching were analyzed in [9, 24]. Similar multi-tasking systems have also been modeled in [13, 25] though not in the domain of priority scheduled systems. Nishida [20] analyzed a heterogeneous multiprocessor system with priority scheduled jobs which arrive from a Poisson source. A continuous-time Markov chain is employed for the analysis, and a lumping scheme is used to tackle the problem of large state space of the CTMC.

In a previous paper [17], we had extended this work in priority queues to consider a more general task arrival behavior. We allow tasks that can arrive to the system externally as well as through spawning/probabilistic branching of other tasks; furthermore this arrival may depend on the number of executions of another task.

This task structure models some important computing systems. For instance, in fault-tolerant computing systems, diagnostic tasks are often spawned by other tasks on occurrence of certain events or errors. On the other hand, a routine which uses a hardware component may initiate diagnostic routines on that component after every n uses of the component. The performance of such systems may be analyzed using the model presented in [17]. Since the resulting state-space is very large, decomposition and iteration was used. In [17] a general model was developed for such systems, and a generalized iterative decomposition

[3]This section is based on the paper "Approximate Analysis of priority scheduling systems using stochastic reward nets" that was presented at the "Thirteenth International Conference on Distributed Computing Systems" in May, 1993.

scheme for such models was presented. A proof for existence of a fixed point was also provided in [17]. In this paper we shall only present an example of such a system, and show that it again fits into the set of general models satisfying certain conditions as described in this paper. Thus we will prove the existence of a fixed point by applying Theorem 2.

5.2 System description

The system analyzed in this section is a heterogeneous multiprocessor system with non-preemptive priority scheduling of tasks. The tasks are classified according to the way they arrive to the system.

- **Poisson Tasks**: these arrive according to a Poisson process, so that the interarrival time of these tasks is exponentially distributed.
- **Sporadic Tasks**: these arrive to the system by spawning or conditional branching of other Poisson and/or sporadic tasks.
- **Poisson and Sporadic**: These arrive from a Poisson source or are created by other tasks.

Each task has a finite buffer where its instances wait to execute on a processor. For instances of the same task, the service discipline is FCFS. Different tasks are served according to different priorities. After a task acquires a processor, it executes on the processor for an exponentially distributed amount of time. A task may create some instances of other sporadic tasks. When the finite buffer limit is reached, any new arriving task of the corresponding type is lost.

The task system consists of a set of tasks, $\mathcal{T} = \{T_1, T_2, \ldots, T_N\}$. Each task T_i is characterized by:

- λ_i : the parameter of the Poisson process according to which the tasks arrive to the computing system. If the task is strictly sporadic, $\lambda_i = 0$.
- A parent set : $\mathcal{P}_i = \{(T_j, m_{j,i}, n_{j,i}, q_{j,i}) \mid T_j \in \mathcal{T}, T_j \text{ spawns task } T_i\}$. In the 4-tuple $(T_j, m_{j,i}, n_{j,i}, q_{j,i})$, $q_{j,i}$ is the probability that $n_{j,i}$ instances of task T_i are spawned by task T_j after the $m_{j,i}$th execution of T_j. This set is empty for strictly Poisson tasks.
- μ_i : the parameter of the exponentially distributed service demand. (Service demand could be number of instructions to be processed, or some other measure of the amount of service required by a task.)
- M_i : the buffer limit.
- $p_i,\ 1 \leq p_i \leq N$: task priority; we use the convention that $p_i > p_j$ implies that T_i has higher priority than T_j.

A task system in which each task is a strictly sporadic task, is a degenerate system; i.e., if we start with an empty system, no task will ever arrive to the system. Hence we require that at least one of the tasks in the task system be Poisson.

The processing system is specified by the following:

- Number of processors: P, the ith processor is denoted by P_i.
- The **capacity** of each processor: $C_i, i = 1, \ldots, P$. The capacity of a processor is its rate of providing service, for example it could be number of instructions that it can process in one second.
- Discipline for allocation of idle processors to tasks: we assume this also to be by order of pre-assigned priority. Let pp_i denote the priority of the processor P_i, where $1 \leq pp_i \leq P$.

We would like to compute performance measures such as the throughputs, utilizations, mean queue lengths and mean response times of this system.

5.3 An example

Consider a two processor heterogeneous system, that maintains information which is regularly read and updated. In this system, we would like to provide the read tasks with as up-to-date information as possible. One way to achieve this effect is to give preference to the update tasks, so that the read tasks are executed *after* the latest update has been performed. Thus update tasks are assigned higher priority than the read tasks. To avoid excessive scheduling overhead, the system adopts non-preemptive priority.

Let T_1 denote the update task, and T_4 denote the read task. During its execution, the update task may come across an erroneous condition in the system. If this error is not critical, the update task spawns an error-handler task[4] denoted by T_5. As a measure of preventive diagnostics, the update task also schedules a diagnostic task (denoted by T_2) after every five executions. Task T_2 runs short diagnostic checks on the system. After every ten executions of the error-handler task (T_5), it schedules another diagnostic task, T_3. This task runs further in-depth diagnostics on the entire system. We assume that the priority of these tasks is in the order $p_1 > \ldots > p_5$ (thus T_1 has highest priority and T_5 has lowest priority). We also assume that priority of processor P_1 is greater than that of P_2. We further assume that tasks T_1 and T_4 arrive to the system according to independent Poisson processes with parameters λ_1 and λ_4 respectively. Finally, let q denote the conditional probability that a non-critical error occurs in the system given that T_1 is running.

This system can be represented by an SRN as shown in Figure 5. Corresponding to each task $T_i, i = 1, \ldots, 5$ there is a place pw_i which represents the buffer. For Poisson task T_1 (T_4), a timed transition tps_1 (tps_4), with firing rate λ_1 (λ_4), deposits tokens in the place pw_1 (pw_4), representing the arrivals of the task according to the Poisson process. For each $i = 1, 4$, the arc from tps_i to pw_i has the marking dependent multiplicity of $\min\{1, M_i - \#(pw_i)\}$. This represents the fact that tasks arriving to a full buffer are lost. (In the figure, variable cardinality arcs are denoted by a "Z" across the arcs.) A token in place pa_j denotes that processor P_j is available, for $j = 1, 2$. For $i = 1, \ldots, 5$ and $j = 1, 2$, the transition $gp_{i,j}$ represents the acquisition of processor P_j by task T_i. The priority

[4] If the error is critical, the processor may take drastic actions such as shutting the system down. Since in this section we are concerned only with performance of a system while it is operational, we do not consider this possibility.

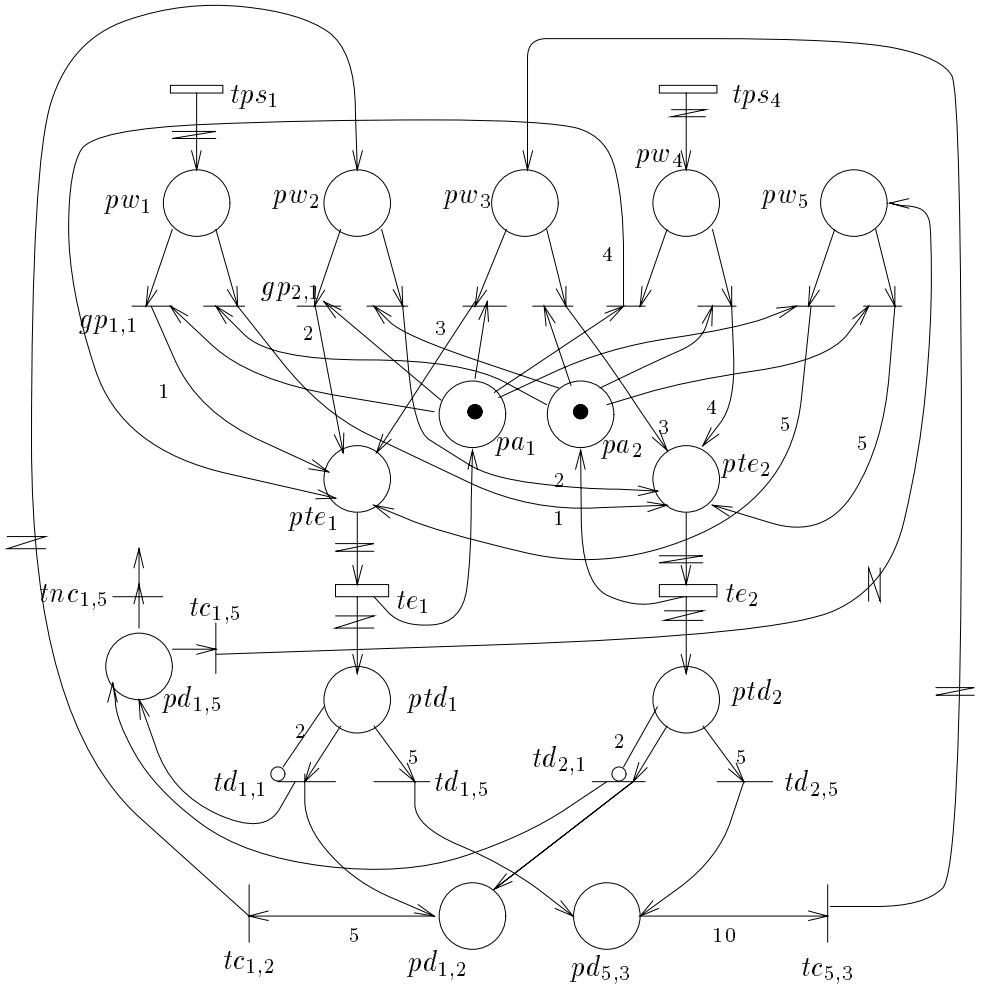


Figure 5: SRN corresponding to the example

scheduling discipline is reflected in the net by assigning the following priority to transition $gp_{i,j}$: $\max\{P, N\} \times p_i + pp_j$, where $P = 2$ and $N = 5$.

On firing, transition $gp_{i,j}$ puts i tokens into place pte_j, indicating that task T_i is executing on processor P_j. The transition te_j represents the execution time of a task on processor P_j. When there are i tokens in place pte_j, the firing rate of $te_j, j = 1, 2$ must represent the execution rate of task T_i on processor P_j and is given by $\mu_i C_j$, when $\#(pte_j) = i$; thus it is a marking-dependent firing rate. On firing, transition te_j removes all tokens from place pte_j and deposits a token back in pa_j. When the number of tokens in pte_j is equal to either 1 or 5, te_j also deposits the same number of tokens that it removes from pte_j, into ptd_j. When the number of tokens in pte_j is equal to 2,3 or 4, transition te_j puts zero tokens in place ptd_j. This is because we need to keep track of the number of executions of tasks that spawn other tasks. For each such parent task T_i $(i = 1, 5)$ and processor P_j, $j = 1, 2$, there is a transition $td_{j,i}$ that recognizes the completion of task T_i by the arrival of i tokens in place ptd_j. This is accomplished by having an input arc from ptd_j with multiplicity i, and an inhibitor arc from ptd_j with multiplicity $i + 1$ (except when $i = 5$, we do not need the inhibitor arc). Thus $td_{j,i}$ fires when there are exactly i tokens in place ptd_j. For each child task T_l of a parent task T_i, there is a place $pd_{i,l}$ with an output arc from $td_{j,i}$ to $pd_{i,l}$ (places $pd_{1,5}, pd_{1,2}$ and $pd_{5,3}$ in the figure). The place $pd_{i,l}$ together with transition $tc_{i,l}$ counts the number of completions of task T_i required to spawn an instance of task T_l. This is done by setting the multiplicity of the arc from $pd_{i,l}$ to $tc_{i,l}$ equal to $m_{i,l}$. In the case of place $pd_{1,5}$ there is also an arc from this place to a transition $tnc_{1,5}$. This is to reflect the probabilistic spawning of task T_5 by task T_1. Thus $tc_{1,5}$ has probability q and $tnc_{1,5}$ has probability $1-q$. Transition $tnc_{1,5}$ sinks the token out of place $pd_{1,5}$ signifying the event that task T_5 was not spawned.

Each transition $tc_{i,l}$ deposits tokens in place pw_l (which is the place representing the buffer for task T_l); the marking-dependent multiplicity of the arc from $tc_{i,l}$ to pw_l is defined as $\min\{n_{i,l}, M_l - \#(pw_l)\}$. The priorities of immediate transitions $td_{j,i}, tc_{i,l}$ and $tnc_{i,l}$ are set to be higher than the priorities of all the transitions $gp_{i,j}$.

This SRN can be mapped to an irreducible CTMC. We can solve this CTMC for its steady-state behavior, and compute the required performance measures, using the SRN specification and solution tool SPNP [7].

5.3.1 Performance measures

Performance measures of the system are derived using appropriate rewards. Define $I_{i,j}(m)$, a function of marking m, as 1 if $\#(pte_j) = i$ and 0 if $\#(pte_j) \neq i$ in marking m. If we assign a reward rate $I_{i,j}(m)$ to marking m, the expected reward rate at steady state is the utilization of P_j by T_i. Average throughput of task T_i is calculated by assigning a reward rate $\mu_i C_1 I_{i,1}(m) + \mu_i C_2 I_{i,2}(m)$ to a marking m of the SRN. The mean queue length (including the tasks in service), of task T_i is calculated by assigning a reward rate of $\#(pw_i) + I_{i,1} + I_{i,2}$ to each marking.

Let Λ_i be the throughput of task T_i, and let L_i be its average queue length. Then by Little's law [27], mean response time is given by: $R_i = L_i/\Lambda_i$.

5.4 Approximate model of the example system

The model described in Section 5.3 gave rise to 17574 states in the underlying CTMC, when $M_i = 1, \forall i = 1, \ldots, 5$. The state space increases exponentially in terms of the model parameters such as the number of task types.

We use a well-known lumping technique [20] to reduce the size of the state space generated by the SRN model. This lumping technique is based on the following observation: from the viewpoint of task T_i, there are only two sets of tasks: tasks that have a priority higher than itself, and tasks that have a priority lower than itself. In other words, it does not matter *which* of the higher priority tasks are waiting to be executed on the processor; if *any* of them is, task T_i will not acquire the processor. Similarly, all lower priority tasks matter only when they are already executing on the processor(s), and task T_i has to wait for them to finish. We apply this observation to decompose the model into two submodels as follows: the first submodel only models tasks T_1, T_2 and T_3 separately as individual tasks and lumps tasks T_4 and T_5 into one task, T_{45} of priority 2 (lowest). In submodel 2, we lump tasks T_1, T_2 and T_3 into one task of priority 3 (highest) and represent tasks T_4 and T_5 separately as individual tasks. (Note that this lumping is largely ad-hoc, and for expository purposes only. The tasks could have been partitioned into other groups of high priority and low priority tasks). Figures 6(a) and (b) depict the SRNs for the two resulting submodels. (In the figures, we continue to use the subscript 4 to denote task T_{45} and the subscript 3 to denote task T_{123}, so as to avoid cluttering).

Note that in submodel 1, the arrival of task T_3 can no longer be explicitly represented as a spawning from task T_5, because task T_5 is not individually represented. Therefore, the arrival of task T_3 is approximated by a Poisson process. The average rate at which task T_3 would have been spawned by task T_5, is the rate of completion of T_5 divided by the number of executions after which it spawns task T_3. Thus the arrival rate of task T_3 is equal to $\Lambda_5/10$.

The spawning of task T_5 by task T_1 is now interpreted as the spawning of task T_{45} by T_1, and the places and transitions are changed to reflect that. The service demand of task T_{45} is approximated as a weighted sum of the individual service demands of task T_4 and T_5. The weights are the conditional probability that tasks T_4 or T_5 have arrived, given that the "lumped" task T_{45} has arrived. Thus, μ_{45} is :

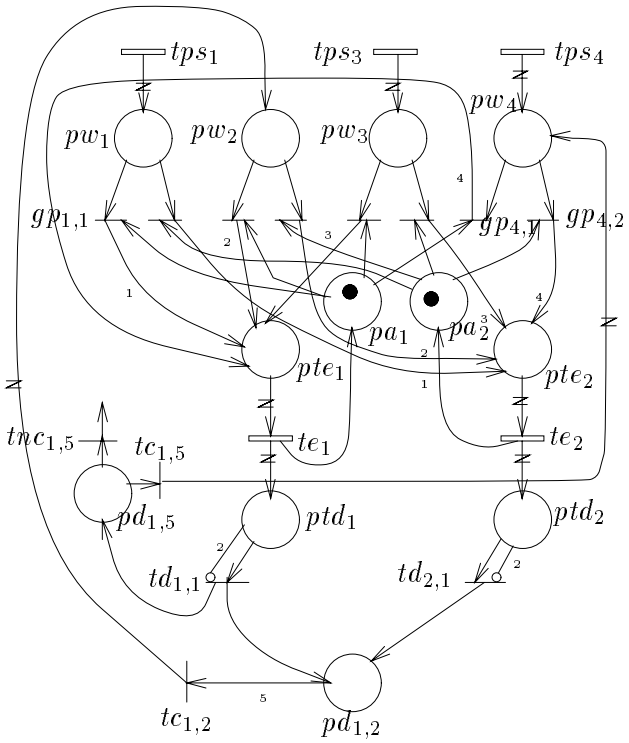
$$\mu_{45} = \frac{\lambda_4}{\lambda_4 + \lambda_5}\mu_4 + \frac{\lambda_5}{\lambda_4 + \lambda_5}\mu_5,$$

where $\lambda_5 = q\Lambda_1$, which is the average rate of arrival of task T_5. The buffer limit for the task T_{45} is set equal to $M_4 + M_5$.

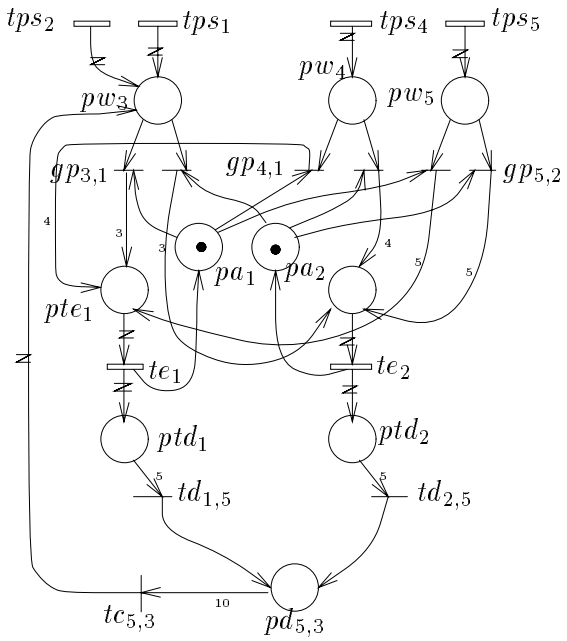
Note that the places and transitions representing the creation of task T_2 by T_1 are left intact, because both tasks T_1 and T_2 are represented individually in the model.

To solve this submodel, we require the measure Λ_5 and Λ_1. Λ_5 can be derived from submodel 2 since it represents task T_5 individually, whereas Λ_1 must be derived from submodel 1 itself, by iterative approximation.

Similarly, in submodel 2, arrivals of tasks T_2 and T_5 are approximated by Poisson processes with rates $\Lambda_1/5$ and $q\Lambda_1$ respectively. Note that the transition tps_2, which has rate



(a)



(b)

Figure 6: (a) Submodel 1 (b) Submodel 2

$\Lambda_1/5$ puts tokens into place pw_3, and so does transition tps_1. This is equivalent to having a single transition with the rate which is the sum of these two rates. The figure shows two transitions for ease of understanding. The service demand of task T_{123} is approximated by :

$$\mu_{123} = \frac{\lambda_1}{\lambda_1+\lambda_2+\lambda_3}\mu_1 + \frac{\lambda_2}{\lambda_1+\lambda_2+\lambda_3}\mu_2 + \frac{\lambda_3}{\lambda_1+\lambda_2+\lambda_3}\mu_3,$$

where $\lambda_2 = \Lambda 1/5, \lambda_3 = \Lambda_5/10$. The buffer limit for task T_{123} is set equal tp $M_1+M_2+M_3$.

To solve this submodel, we need the value of Λ_1, and Λ_5. Λ_1 can be obtained from the solution of sub-model 1, since it represents task T_1 individually. Λ_5 must derived from submodel 2 itself, by iterative approximation.

The two models are clearly interdependent, and must be solved using fixed-point iteration. We must first ensure that a fixed point exists. In the following section, we shall formally define our fixed-point equation, and verify its properties with Theorem 2.

5.5 Existence of a fixed point

In the submodels of Figure 6 there are two iteration variables: Λ_1, the throughput of task T_1, and Λ_5, the throughput of task T_5. The throughputs are calculated as follows:

$$\Lambda_1 = \sum_{m\in\Omega_1} \mu_1(C_1 I_{1,1}(m) + C_2 I_{1,2}(m))\ \pi_m^{(1)}(\Lambda_1,\Lambda_5) \tag{20}$$

$$\Lambda_5 = \sum_{m\in\Omega_2} \mu_5(C_1 I_{5,1}(m) + C_2 I_{5,2}(m))\ \pi_m^{(2)}(\Lambda_1,\Lambda_5) \tag{21}$$

where in Equation (20), Ω_1 is the set of states of submodel 1, and in Equation (21), Ω_2 is the set of states of submodel 2. $\pi^{(1)}$ denotes the steady-state probability vector of of submodel 1 and $\pi^{(2)}$ denotes the steady-state probability vector of of submodel 2. Note that $\pi^{(1)}$ is a function of Λ_1 and Λ_5, because the firing rates of the transitions tps_3 and te_j in SRN 1 are approximated using Λ_5 and Λ_1 respectively. Similarly $\pi^{(2)}$ is a function of Λ_1 and Λ_5, because the firing rate of transitions tps_5 and te_j in SRN 2 are approximated using Λ_1 and Λ_5 respectively. Thus Equations (20) and (21) are the fixed-point equations in two variables corresponding to our iterative scheme.

When Λ_1 and Λ_5 are both greater than zero the two SRNs obviously give rise to irreducible CTMCs with more than one state. When either or both of Λ_1 and Λ_5 are zero it can be verified again that SRNs 1 and 5 give rise to irreducible CTMCs with more than one state. Further, the functions of $\pi^{(1)}$ and $\pi^{(2)}$ in Equations (20) and (21) above satisfy the conditions of Theorem 2, because these functions are weighted sums of state probabilities, where the weights are constants. Therefore a fixed point will exist in a closed subset of $(\mathbb{R}^+ \cup \{0\})^2$. This subset can be identified as follows: From Equation (20), we have,

$$\Lambda_1 \ \leq \sum_{i\in\Omega_1} \mu_1(C_1+C_2) = \mu_1(C_1+C_2)\times|\Omega_1| = u_1 \quad (say). \tag{22}$$

Similarly from Equation (21),

$$\Lambda_5 \ \leq \sum_{i \in \Omega_2} \mu_5(C_1 + C_2) = \mu_5(C_1 + C_2) \times |\Omega_2| = u_5 \ \ (say). \tag{23}$$

A fixed point then exists in the set $[0, u_1] \times [0, u_5]$.

5.6 Numerical Results

In all our experiments, the iteration process converged fairly rapidly, in about 5-6 iterations. The savings gained in space were enormous. Table 3 shows the number of states of the underlying CTMC generated by the same system in the complete model, and in the two sub-models, when the buffer limit on tasks is varied. For system sizes where we could

Exact	Submodel 1	Submodel 2
6788	1085	1630
9134	1301	2170
10157	1625	1900
13676	1949	2530
17574	1533	1510
26182	2029	2230
101398	6689	4690

Table 3: Savings in state space size

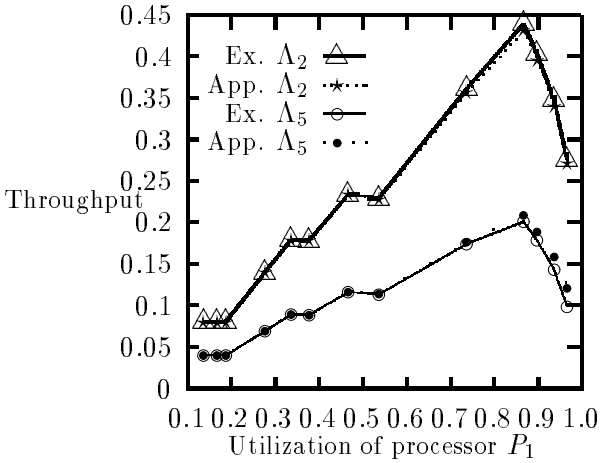


Figure 7: Exact *vs* approximate throughputs

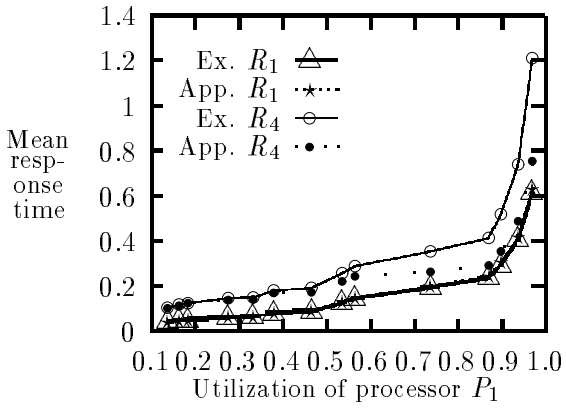


Figure 8: Exact *vs* Approximate response times

solve both the approximate and the complete model, we compared the approximation results with the exact results.

Figures 7 and 8 compare the exact *vs* approximate throughput and the response time, respectively, for different utilization levels of processor P_1. The approximation error of high priority tasks at very high utilization levels stays quite low ($\leq 5\%$). The approximation error of response times of lower priority jobs is small for lower utilizations but deteriorates as the utilization increases.

6 Conclusions

In this paper we outlined some conditions that provide an easy check to verify the existence of a fixed point for an iterative SRN model. This theorem is useful because in many SRN models, the iteration variables are expected reward rates which automatically satisfy part of the conditions. This paper pointed out the intricate properties of continuity etc., that must be kept in mind before assuming that a fixed point exists. This work contributes towards the possibility of automating the solution of SRNs by decomposition and aggregation.

We showed that some existing models from the literature fit into the conditions described in this paper, and also applied this technique to a priority scheduling system with branching and spawning of tasks.

There are many related issues that need further research : the convergence of the iterative method, the correctness and uniqueness of the fixed point, and the errors resulting because of the use of an iterative method, are all problems that are still unresolved in the general context.

Acknowledgements

The authors would like to thank Dr. Phil Chimento, the referees of PNPM'95 and the referees of IEEE-TSE for very detailed and useful comments.

References

[1] M. Ajmone Marsan, G. Conte, and G. Balbo. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Trans. Comput. Syst.*, 2(2):93–122, May 1984.

[2] J. T. Blake and K. S. Trivedi. Reliability analysis of interconnection networks using hierarchical composition. *IEEE Trans. Reliability*, R-38(1):111–120, Apr. 1989.

[3] A. Chesnais, E. Gelenbe, and I. Mitrani. On the modeling of parallel access to shared data. *Comminications of the ACM*, 26(3):196–202, 1983.

[4] Hoon Choi and Kishor Trivedi. Approximate performance models of polling systems using stochastic Petri nets. In *Proceedings of the IEEE INFOCOM 92*, pages 2306–2314, Florence, Italy, May 1992.

[5] G. Ciardo. *Analysis of Large Stochastic Petri Net Models.* PhD thesis, Department of Computer Science, Duke University, Durham, NC, Apr. 1989.

[6] G. Ciardo, A. Blakemore, P. F. Chimento, and K. S. Trivedi. Automated generation and analysis of Markov reward models using stochastic reward nets. In A. Freidman and Jr. W. Miller, editors, *Linear Algebra, Markov Chains, and Queueing Models, IMA Volumes in Mathematics and its Applications*, volume 48, pages 145–191. Springer-Verlag, Heidelberg, 1993.

[7] G. Ciardo, J. Muppala, and K. Trivedi. SPNP: Stochastic Petri net package. In *Proc. Int. Conf. on Petri Nets and Performance Models*, pages 142–150, Kyoto, Japan, Dec. 1989.

[8] G. Ciardo and K. S. Trivedi. A decomposition approach for stochastic Petri net models. *Perf. Eval.*, 18(1):37–59, July 1993.

[9] J. Daigle and C. E. Houstis. Analysis of a task oriented multipriority queueing system. *IEEE Trans. Communications*, 29(11):1669–1677, Nov. 1981.

[10] W. Fleming. *Functions of Several Variables.* Springer-Verlag, New York, 2nd edition, 1977.

[11] B. R. Haverkort. Approximate performability and dependability analysis using generalized stochastic Petri nets. *Performance Evaluation*, 18(1):61–78, July 1993.

[12] P. Heidelberger and K. Trivedi. Queueing network models for parallel processing with asynchronous tasks. *IEEE Transactions on Computers*, C-31(11):1099–1108, Nov. 1982.

[13] P. Heidelberger and K.S. Trivedi. Analytic queuing models for programs with internal concurrency. *IEEE Transactions on Computers*, 32:73–82, 1983.

[14] Jean-Baptiste Hiriart-Urruty and Claude Lemarechal. *Convex analysis and minimization algorithms*. Springer-Verlag, Berlin, 1993.

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

[16] P. Lancaster and M. Tismenetsky. *The Theory of Matrices*. Academic Press, San Diego, CA, U.S.A., 2nd edition, 1985.

[17] V. Mainkar and K. S. Trivedi. Approximate analysis of priority scheduling systems using stochastic reward nets. In *Thirteenth International Conference on Distributed Computing Systems*, pages 466–473, Pittsburgh, PA, May 1993.

[18] M. K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Trans. Comput.*, C-31(9):913–917, Sept. 1982.

[19] J. K. Muppala and K. S. Trivedi. Composite performance and availability analysis using a hierarchy of stochastic reward nets. In G. Balbo, editor, *Proc. Fifth Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, Torino, Italy, Feb. 1991.

[20] T. Nishida. Approximate analysis for heterogeneous multiprocessor systems with priority jobs. *Perf. Eval.*, 15:77–88, June 1992.

[21] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, Inc., New York, 1970.

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

[23] R. A. Sahner and K. S. Trivedi. A hierarchical, combinatorial-Markov method of solving complex reliability models. In *Proc. of the Fall Joint Comput. Conf.*, pages 817–825, Dallas, Texas, Nov. 1986.

[24] B. Simon. Priority queues with feedback. *Journal of the ACM*, 31(1):134–149, 1984.

[25] A. Thomasian and P.F. Bay. Analytic queuing network models for parallel processing of tasks systems. *IEEE Trans. on Computers*, 35(12):1045–1054, Dec. 1986.

[26] Lorrie A. Tomek and K. S. Trivedi. Fixed-point iteration in availability modeling. In M. Dal Cin, editor, *Informatik-Fachberichte, Vol. 91: Fehlertolerierende Rechensysteme*, pages 229–240, Berlin, 1991. Springer-Verlag.

[27] K. S. Trivedi. *Probability & Statistics with Reliability, Queueing, and Computer Science Applications*. Prentice-Hall, Englewood Cliffs, NJ, U.S.A., 1982.

[28] K. S. Trivedi and R. Geist. Decomposition in reliability analysis of fault-tolerant systems. *IEEE Trans. Reliability*, R-32(5):463–468, Dec. 1983.

[29] M. Veeraraghavan and K. S. Trivedi. Hierarchical modeling for reliability and performance measures. In S. K. Tewksbury, B. W. Dickson, and S. C. Schwartz, editors, *Concurrent Computations: Algorithms, Architecture and Technology*. Plenum Press, New York, 1987.

A Overview of SRNs

A Petri net [22] is a directed bipartite graph with two types of nodes called *places* (represented by circles) and *transitions* (represented by rectangles or bars). Directed arcs (represented by arrows) connect places to transitions, and vice versa. If an arc exists from a place (transition) to a transition (place), then the place is called an input (output) place of that transition, and the arc is called an input (output) arc of that transition. Places may contain *tokens* (represented by dots or numbers). The state of a Petri net is defined by the number of tokens in each place, and is represented by a vector $M = (l_1, l_2, \cdots, l_k)$, called a *marking* of the Petri net, where l_i is the number of tokens in place i and k is the number of places in the net. The notation $\#(i, M)$ is used to denote the number of tokens in place i in marking M. A *multiplicity* is a non-negative integer that may be associated with an input or output arc. A transition is said to be *enabled* if each of its input places contains at least as many tokens as that input arc's multiplicity. An enabled transition can fire. When it fires, as many tokens as an input arc's multiplicity are removed from the corresponding input place, and as many tokens as an output arc's multiplicity are deposited in the corresponding output place.

Structural extensions to Petri nets include *inhibitor* arcs (denoted by an arc with a circle instead of an arrow head), which connect places to transitions. A transition can be enabled only if the number of tokens in its inhibitor place is less than the multiplicity of the inhibitor arc.

A set of transitions is said to be *conflicting* when the firing of one disables the rest. Transitions may be assigned priorities that can be used to resolve conflicts between transitions.

Stochastic Petri Nets [18] are Petri nets in which we associate an exponentially distributed time delay with transitions. Generalized Stochastic Petri Nets [1] allow transitions to have an exponentially distributed time delay (*timed* transitions, represented by rectangles) or a zero time delay (*immediate* transitions, represented by bars) associated with them. The *firing rate* of the timed transitions may also be marking-dependent. A marking of a GSPN is said to be *vanishing* if at least one immediate transition is enabled in it and is said to be *tangible* otherwise. Conflicts among immediate transitions in a vanishing marking are resolved by by assigning probabilities to conflicting sets of immediate transitions. This probability may also be marking-dependent.

GSPNs can be mapped to continuous-time Markov chains [5]. If the resulting CTMC is irreducible, we can compute the steady-state probability vector, π of the CTMC.

A *Stochastic Reward Net* [19] is obtained by associating *reward rates* with markings of a GSPN. We associate a reward rate r_i with every tangible marking of the SRN, then the expected reward rate at steady-state can be computed as $\sum_i r_i \pi_i$. Several more extensions have been made in SRNs, which include allowing multiplicities of arcs to be *marking-dependent*. Such arcs are termed *variable cardinality* arcs. Further, enabling functions or *guards* may be associated with transitions. Guards are marking-dependent predicates which have to be satisfied (should evaluate to *true*) for transitions to be considered enabled.

With the use of appropriate reward rates, the expected reward rate can give us several useful measures of a model. In our SRN models, reward rates will be various performance indices; thus expected reward rate at steady-state will give us the average values of the performance measures of the system.