Scientia Iranica, Vol. 8, No. 4, pp 287–296

SCIENTIA
IRANICA

Modeling Performance of Distributed Programs by Stochastic Decision Free Petri Nets

M. Ghodsi[1]

In this paper, performance modeling of synchronization delays is considered in a distributed program which consists of a number of processes that interact via message passing only. A class of timed Petri nets called Stochastic Decision Free Petri Nets (SDFPNs) is used to model such distributed programs with deterministic control flow. An exact solution technique is proposed for this model which does not follow the usual approach of reachability analysis for Petri nets and solving global balance equations for a Markovian system. Therefore, it does not require exponential distributions and does not suffer from state space explosion. The complexity of exact solution is still exponential in terms of the number of transitions. Based on this solution, an iterative approximate algorithm is also proposed which is applicable to reasonably large models with much less complexity. Experimental results verify this claim.

INTRODUCTION

In recent years a great deal of interest has been observed with respect to distributed and concurrent system research, which has generated a growing demand for cost-effective and accurate performance prediction tools to support design of such systems. In particular, efficient techniques are needed for predicting the performance measures of distributed programs running on a network of workstations. Such a tool requires an appropriate model to represent the important attributes of the distributed programs and an efficient solution technique for solving reasonably large models.

In this paper, the performance modeling of communication delays in a distributed program has been considered that consists of a number of processes, each allocated to some node of a computer network. It is assumed that these processes interact only via deterministic message passing primitives. The performance of such programs is mainly affected by two kinds of delays: a) Those due to resource contention (e.g., contention for CPU, disks, communication processors, etc.) and b) Delays caused by synchronization and communication between processes. The main emphasis of this paper is on the second aspect, since contention can be adequately handled by classical queuing network models. It is assumed, throughout the paper, that all processes are cyclic and dynamic process creation and/or destruction occur in a controlled way.

Many techniques have been proposed for modeling synchronization and communication delays. One of the early models is Petri Nets (PNs) which was originally developed as a specification tool for concurrency and was later used to model arbitrary synchronization in concurrent systems [1]. Petri nets have been augmented with a notion of time and used for performance evaluation. The "time" parameter has been assigned to places, ([2] timed-place PNs) as well as transitions ([3] timed-transition PNs.) The time value has been chosen to be deterministic [4,5] or stochastic [6-8]. In Stochastic Petri Nets (SPNs), the resulting models can be cast as Markovian models and solved accordingly. In [7], exponentially distributed transition firing times are used for the first time. Arbitrary distributions were first permitted in [9] for transitions that are not simultaneously enabled. All of these approaches perform reachability analysis to create the state space which grows exponentially with the net parameters. With careful modeling, it is possible to slow down the growth of state space, as shown in [5]; however, complex models remain intractable. Nevertheless, some restricted classes of SPNs, like the one in [10], have been characterized to have product-form solutions. This has led to development of algorithms for these

1. *Department of Computer Engineering, Sharif University of Technology, Tehran, I.R. Iran.*

classes of SPNs, similar to those for product-form queuing networks [11].

There have been other approaches to modeling synchronization and communication delays. Kant proposed a combination of classical queuing networks and Petri nets [12]. Heuristic modeling and solution techniques based upon product form queuing networks have also been used in many instances [13,14]. Synchronization delays can, in many cases, be adequately modeled by Markovian models, however, it is typically difficult to do better than just solving the global balance equations. Since the state space generally grows very fast with the model complexity, this approach is incapable of handling large models.

In this paper, a restricted class of timed-place stochastic Petri nets called Stochactic Decision Free Petri Nets (SDFPNs) is used. The times that are assigned to places can have arbitrary distributions, representing normal computation including activities of production, transmission and consumption of messages. Transitions represent synchronization between sender and receiver, since message passing is the only interaction scheme available. It is assumed that the basic message passing is *rendezvous*, i.e., the sender must wait until the receiver is ready. SDFPNs can model many complex distributed programs with deterministic flow graphs. There are many "real-world" programs that do not, however, satisfy the simplifying assumptions to be modeled by SDFPNs. For these programs, there seem to be no modeling solutions other than expensive reachability analysis techniques.

Here, an exact solution technique is proposed for computation of the SDFPNs performance measures, such as synchronization delay distributions. The solution does not need reachability analysis and is not Markovian. The complexity of exact solution is still exponential in the worst case and is comparable to that of Markovian based solutions. However, this approach promises a good approximate algorithm with polynomial complexity which is also presented in this paper.

There have been several works on analysis of SDFPNs. Li and Woodside, for example, proposed a state reduction technique and an iterative approximate solution for stochastic marked graphs [15]. In [16], the authors have found various bounds for asymptotic cycle time and throughput of SDFPNs. Recently, Sereno has proposed an iterative approximate solution based on Mean Value Analysis (MVA) [17] for SDFPNs [18]. The approach presented here for solving SDFPNs is new in comparison to previous works.

The organization of this paper is as follows. In the next section, some definitions and terminologies are reviewed. Exact and approximate solution techniques for DFSPN with some examples are also presented. Then, the experimental validation of the approximate solution techniques is discussed.

BASIC TERMINOLOGIES AND DEFINITIONS

Recall that a Petri Net $\Sigma = (\mathcal{P}, \mathcal{T}, \mathcal{E}, \mathcal{M}_0)$ is a bipartite graph where $\mathcal{P}$, $\mathcal{T}$ and $\mathcal{E}$ are sets of places, transitions and arcs respectively and $\mathcal{M}_0$ is the initial marking. Let,

$$\mathcal{P} = \{p_1, p_2, \ldots, p_m\},$$

$$\mathcal{T} = \{t_1, t_2, \ldots, t_n\},$$

$$\mathcal{E} \subset \{\mathcal{P} \times \mathcal{T}\} \cup \{\mathcal{T} \times P\},$$

$$\mathcal{M}_0 = \{m_{01}, m_{02}, \ldots, m_{0m}\}.$$

For any transition $t \in \mathcal{T}$, ${}^*t$ and t^* are respectively defined as the set of input and output places of t. Similar sets are defined for input and output transitions of any place $p \in \mathcal{P}$ as ${}^*p$ and p^*. A marking $\mathcal{M}$ is said to be *reachable* from $\mathcal{M}_0$, denoted by $\mathcal{M}_0 \xrightarrow{\sigma} \mathcal{M}$, if there exists a sequence of transition firing, σ, such that starting from $\mathcal{M}_0$ and after the firing of σ, the PN ends in marking $\mathcal{M}$. The set of all markings reachable from $\mathcal{M}_0$ is called the *reachability set* and represented by $\mathcal{R}(\mathcal{M}_0)$. A transition t is said to be *live* if starting from any $\mathcal{M} \in R(\mathcal{M}_0)$, there is a firing sequence that enables t. A PN is live if all of its transitions are live. A PN is *k-bounded* if for all $\mathcal{M} \in R(M_0)$, every place has at most k tokens. A 1-bounded PN is called *safe*.

A PN can be represented compactly using matrices. Define matrix A as $A^+ - A^-$ where A^+ represents arcs from $\mathcal{T}$ to $\mathcal{P}$ and A^- represents arcs to $\mathcal{T}$ from $\mathcal{P}$. (Self loop places are not represented in A. This can be taken care of by adding an extra transition and place after those places.) A is an $n \times m$ matrix with elements of 0, 1 and -1 (multiple arcs are not permitted). Also let M, an $m \times 1$ column vector, represent a marking $\mathcal{M}$. Then, if $\mathcal{M}_0 \xrightarrow{\sigma} \mathcal{M}$, there exist a firing vector, F (an $n \times 1$ column vector), such that $M = M_0 + A^T F$. $F[i]$ is the number of firings of transition i in σ. (Note that there exit Fs that result in feasible but unreachable markings $\mathcal{M}$.)

A Decision Free Petri Net (DFPN), also called a *marked graph* [19], is a PN such that for all $p \in \mathcal{P}$, $|{}^*p| = |p^*| = 1$. In marked graphs, places are eliminated without altering the PN topological properties. A PN is called *Free Choice* when for all $p \in \mathcal{P}$, if $|p^*| > 1$, then $\forall t \in p^*, {}^*t = \{p\}$. That is, if p is a conflict input place for a set of transitions, it should be their only input place. A *loop* in a DFPN is a directed cycle in its underlying marked graph. The notations $t \in \ell$ and $p \in \ell$ will be used to show that transition t and place p are in the loop ℓ.

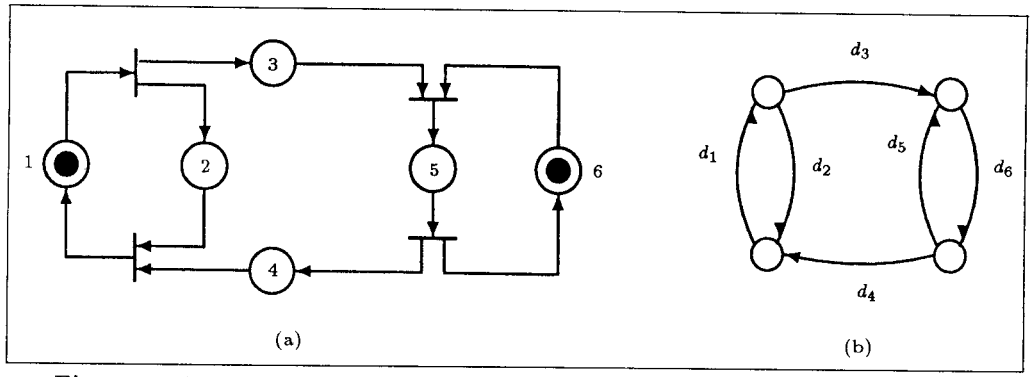

Figure 1. A simple communication protocol: (a) SDFPN and (b) its marked graph.

For a DFPN Σ, the following early results are known that will be used in the analysis [19,20]:

1. The token count in any loop is invariant under any transition firing;
2. The maximum number of tokens at $p \in \mathcal{P}$, in any $\mathcal{M} \in R(M_0)$, is the minimum total number of tokens in all the loops that include p;
3. Σ is live iff it does not have a loop with a token count of zero;
4. Rank of the incidence matrix A is $n-1$;
5. A safe and live DFPN with a connected underlying undirected graph is strongly connected;
6. A safe and live DFPN is a union of one-token loops.

A *Timed Petri Net* is a PN augmented to have a time parameter associated with either its places or its transitions. In this paper, timed-place PNs are considered due to being more natural for modeling processes. The results presented in this paper also hold for timed-transition PNs or a combination of both.

A Stochastic Decision Free Petri Net (SDFPN) is a DFPN with randomly distributed time associated with its places. Let $X_i(k), i = 1 \ldots m, k = 1, 2, \ldots$ be the delay (possibly zero) that is encountered by the kth token arriving at place p_i. The token is "unavailable" for this period of time which is called *computation delay*. After this time, the token becomes available and, only then, it can enable the output transition $t \in p_i^*$. The same token may have to wait for the arrival of other tokens at t. This waiting time is called *synchronization delay* and is denoted by a random variable $D_i(k)$. The total delay that the kth token encounters on an arc i (from transition ${}^*p_i$ to p_i^*) in the underlying marked graph is then $d_i(k) = X_i(k) + D_i(k)$. The time between the kth and the $(k+1)$th firings of a transition t is denoted by $C_t(k)$. It is assumed that for a given i, $X_i(k), k = 1, 2, \ldots$ are independent and identically distributed and represented by a continuous time random variable X_i. $X_i, i = 1, \ldots, m$ are also assumed to be independent.

The properties mentioned above for a DFPN are also true for any SDFPN. Figure 1 shows a SDFPN model of a simple communication protocol. DFPNs can only model deterministic synchronizations; thus, a more powerful PN, such as free choice PN, has to be used to model non-deterministic behavior like timeout, bad message, lost of acknowledgments and so on.

SOLUTION TECHNIQUES

Exact Algorithm

In this section, an algorithm is provided for exact computation of the performance parameters of any live and safe SDFPN. Consider a transition t with $|{}^*t| = s \geq 2$ which is depicted in Figure 2. Let $\ell_1, \ell_2, \ldots, \ell_s$ be the loops containing t such that $p_i \in {}^*t$ is included in ℓ_i, $i = 1, \ldots s$. Take two such loops, say ℓ_1 and ℓ_2, and assume that the number of tokens in ℓ_1 is q and that for ℓ_2 is one. Loop ℓ_2 must exist since the net is safe. The tokens on ℓ_1 never meet at a place or cross over one another, because of the safeness property. Thus, they might be considered "colored" and denoted by $T_1, T_2, \cdots, T_q$. Now, focus on the time period between two consecutive firings of t caused by

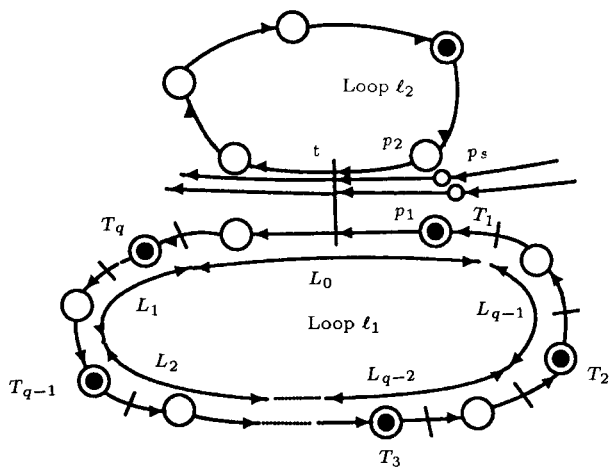

Figure 2. Different loops going through transition t.

one of the tokens in ℓ_1, say T_1. Suppose τ_k is the time of kth firing of t and assume that this firing was caused by T_1. Then, τ_{k+q} is the time that t is fired again by T_1. For notational simplicity, it is assumed that counting token arrival at transitions is started from the time transition t is first enabled. Then,

$$\begin{aligned}\tau_{k+q} - \tau_k &= C_t(k) + C_t(k+1) \\ &\quad + \ldots + C_t(k+q-1) \\ &= \sum_{n=k}^{k+q-1} \sum_{j \in \ell_2} d_j(n). \end{aligned} \tag{1}$$

On the other hand, this period is equal to the total delay that token T_1 encounters in a trip around ℓ_1. At the time τ_k, depending on the position of other $q-1$ tokens in ℓ_1, ℓ_1 is divided into q disjoint paths, $L_i, \quad i = 0 \cdots q-1$ (see Figure 2). When T_1 enables a transition $j \in L_i$ during $[\tau_k, \tau_{k+q}]$, transition j will be firing for its $(k+i)$th times. Therefore,

$$\begin{aligned}\tau_{k+q} - \tau_k &= \sum_{j \in L_0} d_j(k) + \sum_{j \in L_1} d_j(k+1) \\ &\quad + \ldots + \sum_{j \in L_{q-1}} d_j(k+q-1), \end{aligned} \tag{2}$$

It is also obtained that,

$$D_1(k) D_2(k) \ldots D_s(k) = 0 \qquad k = 1, 2, \ldots , \tag{3}$$

representing the fact that one of the synchronization delays at t must always be zero.

The following lemma helps in simplifying the above equations.

Lemma 1

In a live and bounded timed-place Stochastic PN (SPN) Σ, if the distribution of computation delays (X's) has a rational Laplace transform, then the synchronization delays $D_i(k)$ have Unique Limiting Distributions (ULD) (when $k \to \infty$).

Proof

First an equivalent timed-transition SPN is constructed. This is trivially done by replacing a timed-place by two regular places and a timed-transition with the same distribution as its firing time. Notice that in the equivalent SPN, the timed-transitions each have only one input place and act as delay stations in the queuing theory notation; they remove their input tokens as soon as they become enabled (this firing rule is different from those chosen by some authors, but does not change the result). Since any service time density with rational Laplace transform can be represented by a series-parallel network of exponential stages [17], an equivalent SPN can be constructed with immediate and exponentially distributed timed-transition. This model is now equivalent to the GSPN proposed in [6]. The state for such a model, defined as the number of tokens in places at any time, has been shown to have a ULD. However, since the synchronization delays are defined for individual tokens, the net has to be considered as "colored" and the state should be defined accordingly. It is clear that this model has also a ULD from which it can be concluded that the synchronization delays have a ULD too. In fact, it is possible to compute these distributions from the limiting distribution of the Markov chain and the state diagram.□

Notice that, practically, almost all distributions can be well approximated by a distribution with rational Laplace transform. One exception is the deterministic time distribution. In the rest of this paper, it is assumed that the computation delay distributions have this property.

Lemma 2

In a live and safe SDFPN Σ, all transitions have the same limiting cycle time distribution.

Proof

From Lemma 1, it can be concluded that the cycle time distribution of any transition in the net converges. In this lemma, it is shown that, at the limiting case, the cycle time distributions are all the same. Consider a one-token loop, ℓ with r transitions $t_1, \ldots, t_r$ in that order. Let $\mathcal{S}$ be the set of all states of the equivalent GSPN model described in lemma 1 and let the random variable C_{t_i} denote the limiting cycle time distribution of any transition $t_i \in \ell$. Also, let $\mathcal{S}_i \subset \mathcal{S}$ be the set of states in which the loop token of ℓ is at the input place of the transition t_i. Notice that $\mathcal{S}_i$'s are disjoint sets. For this lemma, concentrate is placed on a subset of the Markov chain with states in $\cup_{i=1}^{r} \mathcal{S}_i$.

Let $P(s_u, s_v)$ $(s_u, s_v \in \mathcal{S}_\ell)$ be the probability that the model changes state from s_u and eventually reaches s_v as a result of a sequence of firings γ such that it includes exactly one firing of transition t_1 and no other transition in ℓ appears in γ. It is clear that $s_u \in \mathcal{S}_i$, $s_v \in \mathcal{S}_j$ and $j = (i+1) \bmod r + 1$. Also let Z_{uv} be a random variable that denotes the amount of delay taken for the system to change the state from s_u to s_v. Z_{uv} can be computed from the original state diagram and in general can be represented in the form of branching Erlang. Now, to compute the cycle time distribution for a transition in ℓ, say t_1, the probability of being at any state $s \in \mathcal{S}_1$ is first found. Then, from the transition function $P(s_u, s_v)$ defined above, all the possible cycles that bring back the state from any $s \in \mathcal{S}_1$ to any state in $\mathcal{S}_1$ can be obtained. The cycle time for each possible cycle is represented by a branching Erlang distribution with

Z_{uv}'s as its components. C_{t_1} can now be computed easily. It can be seen that following the same process for other transitions gives the same final distribution for their cycle times. Since the net Σ is a union of one-token loops, it is concluded that the limiting cycle time distributions of all transitions are the same.□

It is known that in any loop ℓ there is one transition $t \in \ell$, such that for all k, $C_t(k) = \sum_{i \in \ell} d_i(k)$. Therefore, the net cycle time, denoted by a random variable C at the limiting case, satisfies the following equation,

$$C = \sum_{i \in \ell} d_i,$$

for any loop ℓ, where d_j is $d_j(k)$ when $k \to \infty$. (Also D_i is used for $D_i(k), k \to \infty$).

Corollary 1

In a live and safe SDFPN, for any loop ℓ with K_ℓ number of tokens, the following equation holds:

$$\sum_{j \in \ell} d_j = K_\ell * C, \tag{4}$$

where C is the limiting cycle time distribution of the net.

Proof

Proof follows Corollary 1 and Equations 1 and 2 which were written for the general case. □

The number of distinct loops in a DFPN with m places and n transitions, and hence the number of equations in the form of Equation 4 can be exponentially large. However, only $m - n + 1$ linearly independent loop equations are exactly needed. Let L, an m column vector, represent a loop in the net; its entries are one for the places in the loop and zero otherwise. Then, $AL^T = 0$. Since the rank of A is $n - 1$, exactly $m - n + 1$ linearly independent Ls representing a set of loops called *fundamental loops* are obtained. If B, a $(m-n+1) \times m$ matrix, represents a set of fundamental loops, then it can easily be computed from $AB^T = 0$. In general, if the rank of A is r and the transition and place numbers are rearranged such that A_{12} is non-singular in,

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

then, $B = [I_\mu | - A_{11}^T A_{12}^{-T}]$ where I_μ is the identity matrix of dimension $\mu = m - n + 1$ [20]. (Here '|' represents matrix concatenation.)

The set of equations in the form of Equation 4 written for a set of fundamental loops B can be shown by:

$$B(D + X) = KC, \tag{5}$$

1. For each $k = 1 \ldots N = \prod_{i=1}^{n} |{}^*t_i|$ cases, take the following steps:
   (a) Let $I_k = \{k_1, k_2, \ldots, k_n\}$ be the set of place indices such that $D_j = 0$ for $j \in I_k$; one for each transition. Also, let $\bar{I}_k$ denote all other indices, i.e., $\bar{I}_k = \{1, 2, \ldots, m\} - I_k$. Solve the set of equations $Q^{(k)} D^{(k)} = -QD = Y$ where $Q^{(k)}$ is Q whose jth columns, $j \in I_k$ have been eliminated. Similarly, $D^{(k)}$ is D without its jth elements ($j \in I_k$). The solution to this set of equations is of form $D_i^{(k)} = \sum_{j=1}^{m} a_{ij}^{(k)} X_j$ for $i \in \bar{I}_k$. If $Q^{(k)}$ is singular ignore this step.
   (b) Compute the probability of occurrence of this case,

$$P\{\text{case} = k\} = \begin{cases} P\{D_j^{(k)} > 0, \forall j \in \bar{I}_k\} & \text{if case } k \text{ occurs} \\ 0 & \text{otherwise} \end{cases}$$

2. Compute the distribution function and mean of the synchronization delays, $D_i, i = 1 \ldots m$,

$$P\{D_i \le d\} = \sum_{k=1}^{N} P\{D_i^{(k)} \le d, case = k\}$$

$$= \sum_{k=1}^{N} P\{D_i^{(k)} \le d, D_j^{(k)} > 0, j \in \bar{I}_k\}.$$

Algorithm 1. Exact algorithm.

where $X[i] = X_i$, $D[i] = D_i$ and $K[i]$ is the number of tokens in the loop represented by the ith row of B. C can be eliminated in Equation 5 and, thereby, $(m - n)$ equations of the following form are obtained:

$$QD = -QX = Y. \tag{6}$$

The rank of Q (a $(m - n) \times m$ matrix) is $m - n$, since the elements of K are all positive integers. Considering n equations in the form of Equation 3, i.e.,

$$\prod_{p_i \in {}^*t} D_i = 0 \qquad \forall t \in \mathcal{T}, \tag{7}$$

m equations have to be solved for m unknown Ds. However, the fact that X and D are all random variables should be remembered.

Assuming that the joint probability of $D_r = 0$ and $D_s = 0$ for any p_r, $p_s \in {}^*t$ is zero, the set of Equation 7 gives $N = \prod_{i=1}^{n} |{}^*t_i|$ possibilities each of the form $D_{i_1} = D_{i_2} = \ldots = D_{i_n} = 0$ where $p_{i_k} \in {}^*t_k$. It is now possible to present an exhaustive algorithm for the exact solution of any SDFPN, which is shown in Algorithm 1.

Lemma 3

Algorithm 1 correctly computes the limiting distribution of the synchronization delays.

Proof

From the assumption that $P\{D_r = D_s = 0\} = 0$, for $p_r, p_s \in {}^*t$, it is clear that the cases mentioned in

Algorithm 1 are disjoint. It remains to show that the probability of a case k for which $Q^{(k)}$ is singular is in fact zero. It then follows that $\sum_{k=1}^{N} P\{case = k\} = 1$ and since the limiting distribution is unique, the proof is complete. Now, assume that for a k, $Q^{(k)}$ is singular. Then, there exists a non-zero $1 \times (m-n)$ row vector b^T such that $b^T Q^{(k)} = 0$. Without loss of generality, suppose that the indices are rearranged such that $I_k = \{m-n+1, m-n+2, \ldots, m\}$. Then, $Q = \left[Q^{(k)} \quad Q_2^{(k)}\right]$ where $Q_2^{(k)}$ is an $(m-n) \times n$ non-zero matrix. The set of equations can then be written as:

$$Q^{(k)} D^{(k)} = -\left[Q^{(k)} Q_2^{(k)}\right] \begin{bmatrix} X_1^{(k)} \\ X_2^{(k)} \end{bmatrix}.$$

By multiplying both sides of the above equation by b^T,

$$0 = \left[0 \quad b^T Q_2^{(k)}\right] \begin{bmatrix} X_1^{(k)} \\ X_2^{(k)} \end{bmatrix},$$

or $b^T Q_2^{(k)} X_2^{(k)} = 0$ with non-zero b and $Q_2^{(k)}$. This is a dependency relation among random variables $X_i, p_i \in I_k$ which contradicts the assumption. Therefore, $P\{case = k\} = 0$.□

It is easily seen that the above algorithm can also be used for timed-transition DFPNs. Moreover, the following lemma shows that this technique can also be applicable to bounded SDFPNs.

Lemma 4

Algorithm 1 also applies to any bounded SDFPN provided that the unsafe places have delays with exponential distributions.

Proof

If the stated condition is true, then the argument given for the correctness of equations in the form of Equation 4 is also true. This is because of the memory-lessness property of the exponential distributions which allows to statistically assume that the tokens on the unsafe loop do not cross over one another. Notice that an unsafe place can be thought of as a delay station in queuing theoretic sense.□

The complexity of this algorithm is exponential which is comparable to that of Markovian global balance solution techniques used in most SPNs. However, it seems possible to identify the "impossible" cases using simple analysis and as a result reduce the complexity of the algorithm to some extent.

Examples

Consider Figure 1 for the first example. For simplicity, assume that $X_2 = X_4 = 0$ and $X_i, i = 1, 3, 5, 6$, have exponential distribution with rates of $\mu_i, i = 1, 3, 5, 6$ respectively. There are three fundamental loops, a set of which is $\{p_1, p_2\}, \{p_1, p_3, p_5, p_4\}$ and $\{p_5, p_6\}$ each with one token. The equations are:

$$d_1 + d_2 = d_1 + d_3 + d_5 + d_4 = d_5 + d_6,$$

$$D_1 = D_5 = D_2 D_4 = D_3 D_6 = 0,$$

and thus,

$$Q = \begin{bmatrix} 0 & -1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & -1 & -1 \end{bmatrix}$$

and:

$$Y = -QX = Q = \begin{bmatrix} -X_3 - X_5 \\ X_5 + X_6 - X_1 \end{bmatrix}.$$

There are only four cases. For the first case, $k = 1$, $D_4 = D_3 = 0$. The solution for this case is:

$$D_2 = X_3 + X_5,$$

$$D_6 = X_1 + X_3 - X_6,$$

$$\begin{aligned} P\{k=1\} &= P\{X_3 + X_5 > 0, X_1 + X_3 - X_6 > 0\} \\ &= P\{X_1 + X_3 > X_6\} \\ &= 1 - \mu_1\mu_3/(\mu_1 + \mu_6)(\mu_3 + \mu_6). \end{aligned}$$

For the second case, $k = 2$ and $D_4 = D_6 = 0$, for which:

$$D_2 = X_5 + X_6 - X_1,$$

$$D_3 = X_6 - X_1 - X_3,$$

$$\begin{aligned} P\{k=2\} &= P\{X_5 + X_6 > X_1, X_6 > X_1 + X_3\} \\ &= P\{X_6 > X_1 + X_3\} = 1 - P\{k=1\}. \end{aligned}$$

This means that it is not necessary to consider the other two cases. (The probability of both cases indeed turns out to be zero.) The distribution of Ds can be computed using:

$$\begin{aligned} P\{D_2 \le d\} &= P\{0 < X_5 + X_6 - X_1 \le d, X_6 > X_1 + X_3\} \\ &\quad + P\{0 < X_3 + X_5 \le d, X_1 + X_3 > X_6\}, \end{aligned}$$

$$P\{D_3 \le d\} = P\{0 < X_6 - X_1 - X_3 \le d, X_5 + X_6 > X_1\},$$

$$P\{D_6 \le d\} = P\{0 < X_1 + X_3 - X_6 \le d, X_2 + X_3 > 0\},$$

which can be computed easily. The mean values of the synchronization delays are as follows:

$$E[D_3] = \frac{\mu_1\mu_3}{\mu_6(\mu_1 + \mu_6)(\mu_3 + \mu_6)},$$

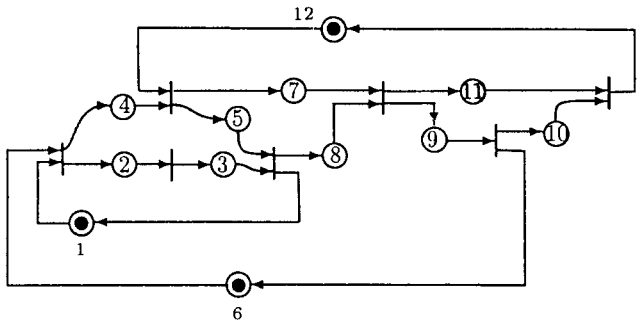

Figure 3. An SDFPN considered for the test of the algorithm.

$$E[D_6] = \frac{1}{\mu_1} + \frac{1}{\mu_3} - \frac{1}{\mu_6} + \frac{\mu_1\mu_3}{(\mu_1+\mu_6)(\mu_3+mu_6)},$$

$$E[D_2] = \frac{1}{\mu_3} + \frac{1}{\mu_5} + \frac{\mu_1\mu_3}{\mu_6(\mu_1+\mu_6)(\mu_3+\mu_6)}.$$

Notice that, in this example, only one of the mean values needs to be calculated using the method shown in Algorithm 1. The rest can be computed using the fundamental loop equations which always hold for the mean values. System throughput can then be found using Little's law.

For the second example, consider the system of Figure 3. There are seven independent loops for which the following equations can be written. Notice that equations for any undirected loop may also, with a little of care, be written. An arbitrary direction is chosen, then the total delay on the loop would be the sum of the delays on arcs which have the same direction minus the total delay on the arc having the opposite direction. The total number of tokens in such a loop is computed similarly:

$$d_1+d_2+d_3=d_8+d_9+d_6+d_4+d_5=d_7+d_1+d_12,$$

$$d_5 + d_8 - d_7 = 0,$$

$$d_4 + d_5 - d_2 - d_3 = 0,$$

$$d_9 + d_{10} - d_{11} = 0,$$

$$D_1D_6=D_4D_{12}=D_5D_3=D_7D_8=D_{10}D_{11}=D_2=0.$$

Approximate Solution

In order to present the approximate solution, first the exact results are presented for a simple SDFPN Σ_1 with only one transition t and m self loop places $p_i, i = 1 \ldots m$. The cycle time distribution for Σ_1 is $C = d_1 = \ldots = d_m$. Following the exact algorithm, there are m cases. In the case k, $D_k^{(k)} = 0$ and $D_j^{(k)} = X_k - X_j$, $\forall j \neq i$. Therefore:

$$\begin{aligned}
&P\{D_i \le d\} \\
&= \sum_{k=1}^{m} P\{0 < X_k - X_i \le d, X_k - X_j > 0, \forall j \neq i,k\} \\
&= \sum_{k=1}^{m} \int_{x_i=0}^{\infty} \int_{x_k=0}^{x_i+d} \left[\prod_{j\neq i,k} \int_{x_j=0}^{x_k} f_{X_j}(x_j)dx_j \right] \\
&\quad \times f_{X_k}(x_k) f_{X_i}(x_i)dx_i \\
&= \int_{x_i=0}^{\infty} \left[\int_{x_k=0}^{x_i+d} \sum_{\substack{k=1 \\ k\neq i}}^{m} \prod_{j\neq i,k} F_{X_j}(x_k) f_{X_k}(x_k)dx_k \right] \\
&\quad \times f_{X_i}(x_i)dx_i \\
&= \int_0^{\infty} \left[\prod_{j\neq i} F_{X_j}(x+d) \right] f_{X_i}(x)dx \ . \qquad (8)
\end{aligned}$$

Equation 8 can also be derived directly from the fact that $C = \max\{X_i, i = 1 \ldots m\}$ and thus,

$$F_C(t) = \prod_{i=1}^{m} F_{X_i}(t). \qquad (9)$$

It is assumed that: (a) It is adequate to work with mean and variance values, and (b) All distributions can be approximated by simple Erlang (ED) or Branching Erlang (BED) depending on their coefficient of variations. Here, only the fitted ED is considered. The formulae will be more complex for BED. In fact, the result would be more accurate if Coaxian approximation is used for the distributions because the distributions of Ds generally have a mass at zero which is not captured by ED or BED.

Let the fitted ED for $X_i, i = 1 \ldots m$, have k_i stages, each with mean service rate of μ_i. First, the mean and variance of C_i are found as the cycle time distribution of the net Σ_1 without place p_i. That is, $F_{C_i}(t) = \prod_{j=1,j\neq i}^{m} F_{X_j}(t)$. The distribution is, then, fitted, by an ED used in Equation 8, to find the mean and variance of D_i. First, $m = 2$ is considered and a recursive formula is used to compute for $m > 2$. Assuming that $F_C(t) = F_{X_1}(t)F_{X_2}(t)$, the first two moments of C can be computed as (p is defined as $3-i$):

$$E[C] = \sum_{i=1}^{2} \frac{k_i}{\mu_i} \left[1 - \sum_{j=0}^{k_p-1} \binom{k_i+j}{j} \left(\frac{\mu_i}{\mu_1+\mu_2} \right)^{k_i+1} \left(\frac{\mu_p}{\mu_1+\mu_2} \right)^{j} \right], \qquad (10)$$

$$E[C^2] = \sum_{i=1}^{2} \frac{k_i(k_i+1)}{\mu_i^2}\Big[1 - \sum_{j=0}^{k_p-1}\binom{k_i+1+j}{j} \times \left(\frac{\mu_i}{\mu_1+\mu_2}\right)^{k_i+2}\left(\frac{\mu_p}{\mu_1+\mu_2}\right)^{j}\Big]. \quad (11)$$

C is approximated by an ED. Now, Equations 10 and 11 can be applied recursively to compute the fitted ED for C_i. That is, if $G(1,t) = F_{X_{i_1}}(t)$ is defined and

$$G(n,t) = G(n-1,t)F_{X_{i_j}}, \quad j = 2\ldots m-1, i_j \neq i. \quad (12)$$

then $G(m-1,t) = F_{C_i}(t)$.

Similar formulae can be derived for $E[D_i]$ and $E[D_i^2]$ using the fitted ED of C_i in Equation 8. Notice that if the only interest were the mean of D_i, the recursion 12 could be continued one more time and the mean value of the net cycle time, C could be found, then $E[D_i] = E[C] - E[X_i]$.

The approximate algorithm is based on the iterative use of the exact solutions for all simple nets Σ_t that are considered for each transition t. These nets are similar to Σ_1 and can be exactly solved. At each iteration, it is assumed that the approximate values for the means and variances of all synchronization delays (Ds) are available. Σ_1 is presented by opening all transitions, except t and allowing the tokens to flow freely. Tokens, however, will face Xs and Ds as two different delays. All loops around t can be easily found, using the method of stages and from the values found in the previous iteration, the mean values and variances of the delays that are encountered by each single token in loops around t can also be obtained. Now, Equations 10, 11 and 12 are used to find new mean values and variances for D_ts, synchronization delays at t. This is done for all transitions and this step is repeated until all values converge. A more formal description of the approximate algorithm is shown in Algorithm 2.

It is difficult to formally argue about the convergence of the above procedure. However, the experimental validation shows that the algorithm converges for reasonable parameter values.

The complexity of the approximate algorithm is $O(Knm^2)$ which is quite fast compared to the exact algorithm (K is the number of iterations.)

EXPERIMENTAL VALIDATION OF APPROXIMATE ALGORITHM

A series of experiments was run to evaluate the accuracy of the approximate solution presented in the paper. Four SDFPNs were chosen to test the algorithm, each with different characteristics, one of which is shown in Figure 3. The number of places were between 10 and 25. For each net, several test data with different delays were created. Branching Erlang distribution was used to fit all the distributions with different coefficients of variation. The total number of tests performed was 20. The convergence criteria had a relative error of less than 0.001. A program was

1. From the basic loop equations of $Qd = 0$, find $r = |{}^*t_i|$ simultaneous equations of the following form for each transition t_i:

   $$g_{i_1}(X,D) + D_{i_1} = g_{i_2}(X,D) + D_{i_2} = \ldots = g_{i_r}(X,D) + D_{i_r},$$

   where $p_{i_j} \in {}^*t_i$ and for some zero/one values of as and bs:

   $$g_{i_j}(X,D) = \sum_{k=1}^{m} a_{ik}X_k + \sum_{\substack{k=1\\k\neq j}}^{m} b_{ik}D_k.$$

   Such equations surely exist but it is better to choose those for which the g functions have the fewest terms possible.
2. Initially assume that $E[D_i] = Var[D_i] = 0$ for $i = 1\ldots n$.
3. Repeat the following steps until mean values for all Ds converge:
   (a) For each transition t_i, compute the means and variances of g_{i_j} as:

   $$E[g_{i_j}] = \sum_{k=1}^{m} a_{ik}E[X_k] + \sum_{\substack{k=1\\k\neq j}}^{m} b_{ik}E[D_k],$$

   $$Var[g_{i_j}] = \sum_{k=1}^{m} a_{ik}Var[X_k] + \sum_{\substack{k=1\\k\neq j}}^{m} b_{ik}Var[D_k].$$

   (b) Treat g_{i_j}s as the Xs for a single transition t in simple net Σ_1. Use Equations 10, 11 and 12 to compute the new values for $E[D_i]$ and $Var[D_i]$.

Algorithm 2. Approximate algorithm.

written to symbolically compute good loop equations for each transition of the input net and calculate the approximation results. Each input net was also automatically converted to a simulation language that is used by PAWS package [21]. The results of the mean values of the cycle times and the synchronization delays of the approximate algorithm and simulation were compared.

As expected, the largest errors occurred when most delays were near the threshold. The largest error for mean synchronization delays was 16% but that for standard deviation was 27%. The largest errors and standard deviation for the mean values of the cycle times were 7% and 12%, respectively. The mean error for synchronization delays of all these tests was found to be 9%. For the mean cycle times this was 4%. These ranges of error are acceptable for most purposes.

No convergence problems were observed in any verification runs. The maximum number of iterations needed was 17. However, on average, 9.6 iterations were sufficient for convergence.

Although the approximate algorithms have not been verified with a complicated real-world system, it seems that these experiments are enough to expect the same performance for even complicated systems that can be modeled by SDFPNs. Many experiments conducted by the author show that small models tend to become more unstable and are prone to more errors compared to complex models with many synchronization steps.

The main source of error in this approximation is the dependencies among the delays. It is assumed that Ds and Xs are independent when computing the variance of gs. Moreover, in Equations 8, 10 and 11 independent Xs are assumed. These are not true in general for gs. The approximate algorithm was made possible, however, by these simplifying assumptions.

CONCLUSIONS

In this paper, an exact solution technique has been presented for any safe and live stochastic decision free Petri net which can be used to model deterministic distributed programs as well as other systems. The algorithm does not do reachability analysis and avoids the use of classical Markovian modeling techniques and, therefore, neither suffers from state space explosion nor demands exponential distributions. It allows arbitrary distributed time attached to places and/or transitions. The net behavior is represented by simple equations which can easily be written using the fundamental loops. The exact algorithm also applies to bounded SDFPNs provided that the unsafe places have delays with exponential distributions. However, the complexity of the algorithm for computing the distributions or even the mean values for synchronization delays is exponential which is comparable to that of other existing approaches. Finding the regions in which each of the N cases can occur is also an expensive procedure which is called upon once in an iteration. Identifying the "impossible" cases beforehand and use of a good region finding algorithm can reduce the complexity of the algorithm to some extent.

In spite of its complexity, the algorithm promises good approximate techniques, one of which was presented in the previous section. The proposed approximation gives reasonably accurate results without incurring exponential complexity, however, it is difficult to analyze the introduced errors.

In order to compute actual delays suffered by processes, resource contention and interprocess communication must be considered simultaneously. This can be done by solving the contention and communication models iteratively, one feeding the other. Note that a process waiting for rendezvous does not compete for system resources, therefore, communication delay can be incorporated in the contention model via a delay station. Further work on this is underway.

REFERENCES

1. Petri, C.A. "Communication with automata", Technical Report RADC-TR-65-377, New York, 1966. original in German; Kommunication mit automaten, Univ. of Bonn (1962).
2. Sifakis, J. "Use of Petri nets and preliminary performance evaluation", in *Measuring, Modeling and Evaluating Computer Systems*, Beimer and Gelenbe, Eds., pp 75-93, North-Holland (1977).
3. Ramachandani, C. "Analysis of asynchronous concurrent systems by timed Petri nets", PhD Thesis, Lab. of Computer Science, MIT, Cambridge, MA, (1974). (tech report# TR-120).
4. Ramamoorthy, C.V. and Ho, G.S. "Performance evaluation of asynchronous concurrent systems using Petri nets", *IEEE Transactions on Software Engineering*, pp 440-449 (1980).
5. Vernon, M. and Holliday, M. "Performance analysis of multiprocessor cache consistency protocols using generalized Petri nets", in *ACM SIGMETRICS Conference*, pp 9-17 (May 1986).
6. Ajmone-Marson, M., Balbo, G. and Conte, G. "A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems", *ACM Transactions on Computer Systems*, pp 93-122 (1984).
7. Molloy, M. "Performance analysis using stochastic Petri nets", *IEEE Transactions on Computer*, pp 913-917 (Sept. 1982).
8. Peterson, J.L., *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Englewood Cliffs, NJ (1981).
9. Dugan, J.B., Trivedi, K.S., Geist, R.M. and Nocola, V.F. "Extended stochastic Petri nets: Applications and

analysis", in *Performance'84*, E. Gelenbe, Ed., pp 75-83, North-Holland (1984).

10. Taylor, P.G. and Henderson, W. "Generalization of queuing network product form solution to stochastic Petri nets", *IEEE Transaction on Software Engineering*, **17**(2), pp 108-116 (Feb. 1991).
11. Balbo, B. and Sereno, M. "Computational algorithms for product form solution stochastic Petri nets", in *Proc. Fifth Int'l Workshop on Petri Nets and Performance Models*, Toulouse France (1993).
12. Kant, K. "Modeling interprocess communication in distributed programs", *International Workshop on Petri Nets and Performance Models, Madison, WI*, pp 75-83 (1987).
13. Heidelberger, P. and Trivedi, K. "Queuing network models for programs with internal concurrency", *IEEE Transactions on Computer*, pp 73-82 (1983).
14. Lazawoska, E.D., et al., *Quantitative System Performance*, Prentice-Hall, Englewood Cliffs, NJ (1984).
15. Lin, Y. and Woodside, C.M. "Iterative decomposition and aggregation of stochastic marked graph Petri nets", in *12th International Conference on Applications and Theory of Petri Nets*, pp 257-275, Gjern, Denmark (1991).
16. Baccelli, F. and Liu, Z. "Comparison properties of stochastic decision free Petri nets", *IEEE Transactions on Automatic Control*, **37**(12), pp 1905-1920 (1992).
17. Kant, K., *Introduction to Computer System Performance Evaluation*, McGraw-Hill Inc. (1992).
18. Sereno, M. "Approximate mean value analysis for stochastic marked graphs", *IEEE Transactions on Software Engineering*, **22**(9), pp 654-664 (1996).
19. Commoner, F. and Holt, A.W. "Marked directed graphs", *Journal of Computer and System Science*, pp 511-523 (1971).
20. Murata, T. "Circuit theoric analysis and synthesis of marked graphs", *IEEE Transactions on Circuits and Systems*, pp 400-405 (1977).
21. Information Research Associates, *PAWS: Performance Analyst's Workbench System*, Austin, Texas (1986).