Modeling Performance of Distributed Programs by Stochastic Decision Free Petri Nets

Mohammad Ghodsi *
Computer Engineering Department
Sharif University of Technology

Abstract

In this paper, we are concerned with performace modeling of synchronization delays in a distributed program that consists of a number of processes that interact via message passing only. A class of Timed Petri Nets called Stochastic Decision Free Petri Nets (SDFPN) is used to model such distributed programs with deterministic control flow. We propose an exact solution technique 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.
Keywords: performance evaluation, stochastic Petri nets, Markov model, non-Markovian solution

1 Introduction

In this paper, we are concerned with performace modeling of communication delays in a distributed program that consists of a number of processes that interact only via deterministic message passing primitives. The performace of such programs are mainly affected by two kinds of delays: (a) those due to resource contention, (e.g., contention for CPU, disks, communication processors, etc.), and (b) the delays caused by synchronization and communication between processes. The main emphasis of the paper is on second aspect, since contention can be adequately handled by classical queueing network models. It is assumed throughout the paper that all processes are cyclic and we allow dynamic process creation and or destruction in a controlled way.

Many techniques have been proposed for modeling synchronization and communication delays. One of the early models is Petri Nets (PN) [15]. Petri nets have been augmented with a notion of time and used for performance evaluation. The "time" parameter has been assigned to places [20] (Timed-Place PN's) as well as transitions [17] (Timed-Transition PN's.) Time can be deterministic [18] or stochastic [1]. In Stochastic Petri Nets (SPN), the resulting models can be cast as Markovian models and solved accordingly. In [12], exponentially distributed transition firing times was used for the first time. Arbitrary distributions was allowed first in [4] 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, however, complex models remain intractable. Recently, a restricted class of SPN's was characterized to have product-form solutions [21]. This has led to development of algorithms for this class of SPN's, similar to those for product-form queueing networks [11].

There has been other approaches to modeling synchronization and communication delays. Kant proposes a combination of classical queueing networks and Petri nets [9]. Heuristic modeling and solution techniques based upon product form queueing network have also been used in many instances [6]. 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, we use a restricted class of timed-place stochastic Petri nets called Stochactic Decision Free Petri Nets (SDFPN). The times that are assigned to places can have arbitrary distributions.

*The author can be reached at `ghodsi@sharif.ac.ir` and his homepage is `http://www.sharif.ac.ir/~ghodsi`.

These represents 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. We assume that the basic message passing is *rendezvous*, i.e., the sender must wait until the receiver is ready. SDFPN can model many complex distributed programs with deterministic flow graphs.

We propose an exact solution technique for computation of the performance measures of SDFPN's 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 worse case and is comparable to that for the Markovian based solution. However, this approach promises a good approximate algorithm which is also presented.

There has been several works on analysis of SDFPN's. Li and Woodside, for example, propose a state reduction technique and an iterative approximate solution for stochastic marked graphs [10]. In [2], the authors find various bounds for asymptotic cycle time and throughput of SDFPN's. Recently, Sereno proposed an iterative approximate solution based on Mean Value Analysis (MVA) [8] for SDFPN's [19].

The organization of this papers is as follows. In section 2, we review some definitions and terminologies. Exact and approximate solution techniques for DFSPN with some examples are discussed in section 3. Due to lack of space, proofs for some lemmas have not been presented. In section 4, the experimental validation of the approximate solution techniques is discuused.

2 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 the sets of places, transitions, and arcs respectively, and $\mathcal{M}_0$ is the initial marking. Let,

$$\begin{aligned} \mathcal{P} &= \{p_1, p_2, \ldots, p_m\} \\ \mathcal{T} &= \{t_1, t_2, \ldots, t_n\} \\ \mathcal{E} &\subset \{\mathcal{P} \times \mathcal{T}\} \cup \{\mathcal{T} \times \mathcal{P}\} \\ \mathcal{M}_0 &= \{m_{01}, m_{02}, \ldots, m_{0m}\} \end{aligned}$$

For any transition $t \in \mathcal{T}$, $^*t$ and t^* are defined respectively as the set of input places 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 $\mathcal{M} \in \mathcal{R}(\mathcal{M}_0)$ 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 $\mathcal{M} \in \mathcal{R}(\mathcal{M}_0)$ every place has at most k tokens. A 1-bounded PN is called *safe*.

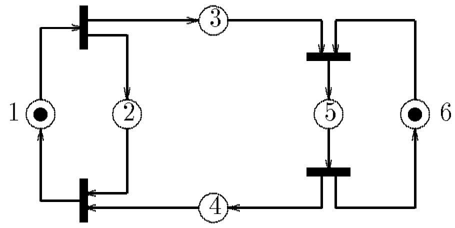

Figure 1: A simple communication protocol.

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 $\mathcal{P}$[1]. A is an $n \times m$ matrix with elements of 0, 1, and -1 (we don't allow multiple arcs.) Also let M, an $m \times 1$ column vector, represents a marking $\mathcal{M}$. Then, if $\mathcal{M}_0 \xrightarrow{\sigma} \mathcal{M}$, there exists 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[2].

A Decision Free Petri Net (DFPN), also called a *marked graph*[3], 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* if for all $p \in \mathcal{P}$, if $|p^*| > 1$ then, $\forall t \in p^*, ^*t = \{p\}$. In words, if p is a conflict input place for a set of transitions, it should be the their only input place. A *loop* in a DFPN is a directed cycle in its underlying marked graph. We will use the notation $t \in \ell$ and $p \in \ell$ to show that transition t and place p are in the loop ℓ.

For a DFPN Σ the following early results are known and we use them in our anlysis [3, 13]:

1. The token count in any loop is invariant under any transition firings.

2. The maximum number of tokens at $p \in \mathcal{P}$, in any $\mathcal{M} \in \mathcal{R}(\mathcal{M}_0)$, is the minimum total number of tokens in all the loops that includes p.

[1] Self loop places are not represented in A. This can be taken care of by adding an extra transition and place after those places.

[2] Note that there exits F's that result in feasible but unreachable markings $\mathcal{M}$.

3. Σ is live iff it does not have a loop with token count of zero.

4. Rank of the incidence matrix A is $n-1$.

5. A safe and live DFPN with 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 we consider timed-place PN's since it is more natural for modeling processes. The results presented in this paper also hold for timed-transition PN's or a combination of both.

A Stochastic Decision Free Petri Net (SDFPN) is a DFPN with randomly distributed time associated with its places. Let $X_i(k), i=1\dots m, k=1,2,\dots$ 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 we call *computation delay.* After this time the token becomes available and, only then, it can enable the output transition $t \in p^*$. The same token may have to wait for the arrival of other tokens at t. We call this waiting time *synchronization delay* and denote it by a random variable $D_i(k)$. 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 $C_t(k)$. We assume that for a given i, $X_i(k), k=1,2,\dots$ are independent and identically distributed and represented by a continuous time random variable X_i. $X_i, i=1,\dots,m$ 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. DFPN's can only model deterministic synchronizations. Thus, we have to use a more powerful PN such as free choice PN to model non-deterministic behavior like time-out, bad message, lost of acknowledgements, and so on.

3 Solution Techniques

3.1 Exact Algorithm

In this section, we will provide an algorithm 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 $\ell_1, \ell_2, \dots, \ell_s$ be the loops containing t such that $p_i \in {}^*t$ is included in ℓ_i, $i=1,\dots s$. Take two such

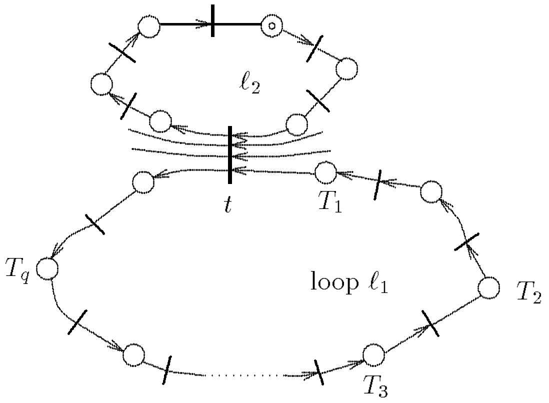

Figure 2: Different loops going through transition t

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 we may consider them "colored" and denote them as $T_1, T_2, \cdots, T_q$. Let us focus on the time period between two consecutive firings of t caused by 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 time that t is fired again by T_1. For notational simplicity, it is assumed that we start counting token arrivals at transitions from the time transition t is enabled first. Then we have,

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

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

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

We also have,

$$D_1(k)D_2(k)\dots D_s(k) = 0 \qquad k=1,2,\dots \qquad (3)$$

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

The following lemma helps 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) have a rational Laplace transform, then the synchronization delays $D_i(k)$ have Unique Limiting Distributions (ULD) (when $k \rightarrow \infty$).*

Proof: Please refer to the complete paper [5]. $\square$

Notice that, practically, almost all distributions can be approximated well by a distribution with rational Laplace transform. One exception is the deterministic time distribution. In the rest of this paper we assume 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, we conclude that the cycle time distribution of any transition in the net converges. In this lemma we show 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 S be the set of all the states of the equaivalent 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 $S_i \subset S$ be the set of states in which the loop token of ℓ is at the input place of the transition t_i. Notice that S_i's are disjoint sets. For this lemma, we consentrate on a subset of the Markov chain with states in $\cup_{i=1}^{r} S_i$ (see Figure ??.)

Let P_{ij} defines $P(s_i, s_j)$, $s_i \in S_u$ and $s_j \in S_v$, be the probability that the model changes state from s_i and eventually reaches s_j as a result of a sequence of firings γ such that γ includes exactly one firing of transition t_u. It is clear that $v = u \text{mod} r + 1$, and that no other tansitions in ℓ appears in γ. Also, let Z_{ij} be a random variable denoting the amount of delay that takes for the system to change state from s_i to s_j. Z_{ij} can be computed from the limiting solution to the state diagram and can be in general represented in the from of branching Erlang.

Now, to compute the cycle time distribution for a transition $t_i \in \ell$, we first find the probability of being at any state s given that $s \in S_i$. Then from the transition function P_{ij} defined above, we can find the probabilities of all possible cycles that starting from any state $s \in S_i$, it takes back the model to any state in S_i. We represent the cycle time for each possible cycle by a branching Erlang distribution with Z_{ij}'s as its components. C_{t_1} can now be computed easily.

Because ℓ is a one-token loop, it is clear that the set of cycles we find for t_i is the same as those for any other transition in ℓ. Therefore, it is not hard to see that if we following the same process for other transitions, the limiting distribution for their cycle times would be the same. Since Σ is a union of one-token loops, we conclude that the limiting cycle time distributions of all transitions are the same. $\square$

We know 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 $k \lim \infty$. (We also use D_i for $D_i(k), k \lim \infty$).

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: Follows from Corollary 1 and equations (1) and (2) which we wrote for the general case. $\square$

The number of distinct loops in a DFPN with m places and n transitions, and hence the number of equations of form (4), can be exponentially large. However, we only need exactly $m - n + 1$ linearly independent loop equations. Let L, an m column vector, represent a loop in the net; its entries are 1's for the places in the loop and zero otherwise. Then, $AL^T = 0$. Since the rank of A is $n-1$ we have exactly $m-n+1$ linearly independent L's representing a set of loops called *fundamental loops*. 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 we rearrange the transition and place numbers such that A_{12} is nonsingular 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 $\mu = m - n + 1$ [13]. (Here '|' represents martix concatenation.)

1. for each $k = 1 \ldots N$ cases, do the following steps:
   (a) Let $I_k = \{k_1, k_2, \ldots, k_n\}$ be the set of indices such that $D_j = 0$ for $j \in I_k$. 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 the jth, $j \in I_k$ elements. Solution to this set of equation 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 occurance 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, $D_i, i = 1 \ldots n$,
$$\begin{aligned} P\{D_i <= d\} &= \sum_{k=1}^{N} P\{D_i^{(k)} <= d, \text{case} = k\} \\ &= \sum_{k=1}^{N} P\{D_i^{(k)} <= d, D_j^{(k)} > 0, j \in \bar{I}_k\} \end{aligned}$$

Algorithm 1. Exact Algorithm.

The set of equations of form (4) written for a set of fundamental loops B can be shown by
$$B(D + X) = KC \tag{5}$$
where $X[i] = X_i$ and $D[i] = D_i$ and $K[i]$ is the number of tokens in the loop represented by the ith row of B. We can eliminate C in (5) and thereby get $(m - n)$ equations of the form
$$QD = -QX = Y. \tag{6}$$

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 of form (3), i.e.,
$$\prod_{p_i \in {}^*t} D_i = 0 \qquad \forall t \in T, \tag{7}$$
we have m equations to be solved for m unknown D's. We should not, however, lose the sight of the fact that X and D are all random variables.

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 equations (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 $p_{i_k} \in {}^*t_k$. We are now ready to present an exhustive algorithm for the exact solution of any SDFPN.

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 the algorithm are disjoint. It remains to show that the probability of a case k for which $Q^{(k)}$ is singular is in fact zero. Then it follows that $\sum_{k=1}^{N} P\{\text{case} = k\} = 1$ 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 we rearrange the indices such that $I_k = \{m{-}n{+}1, m{-}n{+}2, \ldots, m\}$. Then $Q = \begin{bmatrix} Q^{(k)} & Q_2^{(k)} \end{bmatrix}$ where $Q_2^{(k)}$ is an $(m - n) \times n$ non-zero matrix. The set of equations can then be written as,
$$Q^{(k)} D^{(k)} = -\begin{bmatrix} Q^{(k)} Q_2^{(k)} \end{bmatrix} \begin{bmatrix} X_1^{(k)} \\ X_2^{(k)} \end{bmatrix}.$$

By multiplying both sides of the above equation by b^T we get,
$$0 = \begin{bmatrix} 0 & b^T Q_2^{(k)} \end{bmatrix} \begin{bmatrix} X_1^{(k)} \\ X_2^{(k)} \end{bmatrix}$$
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 our assumption. Therefore $P\{\text{case} = k\} = 0$. $\square$

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

Lemma 4 *The above algorithm also applies to any bounded SDFPN provided that the unsafe places have delays with exponential distributions.*

Proof: Please refer to the complete paper [5]. □

The complexity of this algorithm is exponential which is comparable to that of Markovian global balance solution techniques used in most SPN's. 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.

3.2 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 3 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,

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

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

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

$$\begin{aligned} D_2 &= X_3 + X_5 \\ D_6 &= X_1 + X_3 - X_6 \\ P\{k=1\} &= P\{X_3+X_5>0, X_1+X_3-X_6>0\} \\ &= P\{X_1 + X_3 > X_6\} \\ &= 1 - \mu_1\mu_3/(\mu_1+\mu_6)(\mu_3+\mu_6). \end{aligned}$$

For $k = 2$, $D_4 = D_6 = 0$we have,

$$\begin{aligned} D_2 &= X_5 + X_6 - X_1 \\ D_3 &= X_6 - X_1 - X_3 \\ P\{k=2\} &= P\{X_5 + X_6 > X_1, X_6 > X_1 + X_3\} \\ &= P\{X_6 > X_1 + X_3\} = 1 - P\{k=1\} \end{aligned}$$

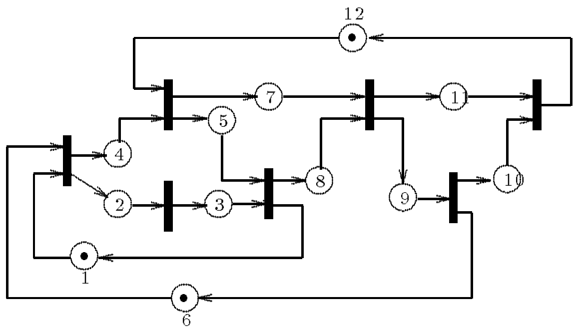

Figure 3: Another example

This means that we don't have to consider other two cases. (The probability of both cases indeed turns out to be zero)

The distribution of D's can be computed using,

$$\begin{aligned} P\{D_2 \le d\} &= \\ & P\{0 < X_5 + X_6 - X_1 \le d, X_6 > X_1 + X_3\} \\ & + P\{0 < X_3 + X_5 \le d, X_1 + X_3 > X_6\} \\ P\{D_3 \le d\} &= \\ & P\{0 < X_6 - X_1 - X_3 \le d, X_5 + X_6 > X_1\} \\ P\{D_6 \le d\} &= \\ & P\{0 < X_1 + X_3 - X_6 \le d, X_2 + X_3 > 0\} \end{aligned}$$

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

$$\begin{aligned} E[D_3] &= \frac{\mu_1\mu_3}{\mu_6(\mu_1+\mu_6)(\mu_3+\mu_6)} \\ E[D_6] &= \frac{1}{\mu_1} + \frac{1}{\mu_3} - \frac{1}{\mu_6} + \frac{\mu_1\mu_3}{(\mu_1+\mu_6)(\mu_3+mu_6)} \\ E[D_2] &= \frac{1}{\mu_3} + \frac{1}{\mu_5} + \frac{\mu_1\mu_3}{\mu_6(\mu_1+\mu_6)(\mu_3+\mu_6)} \end{aligned}$$

Notice that, in this example, only one of the mean values need 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 we can write the following equations. Notice that we may write equations for any undirected loop as well with a bit of care. We choose an arbitrary direction, 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.

$$\begin{aligned}
d_1 + d_2 + d_3 &= d_8 + d_9 + d_6 + d_4 + d_5 \\
&= d_7 + d_1 + d_1 2, \\
d_5 + d_8 - d_7 &= 0, \\
d_4 + d_5 - d_2 - d_3 &= 0, \\
d_9 + d_{10} - d_{11} &= 0, \\
D_1 D_6 = D_4 D_{12} &= D_5 D_3 = D_7 D_8 \\
&= D_{10} D_{11} = D_2 = 0.
\end{aligned}$$

3.3 Approximate Solution

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

$$\begin{aligned}
P\{D_i \leq d\} &= \sum_{k=1}^{m} P\{0 < X_k - X_i \leq d, X_k - X_j > 0, \forall j \neq i, k\} \\
&= \sum_{k=1}^{m} \int_0^\infty \int_0^{x_i+d} \left[\prod_{j \neq i,k} \int_0^{x_k} f_{X_j}(x_j) dx_j \right] f_{X_k}(x_k) f_{X_i}(x_i) dx_i \\
&= \int_0^\infty \left[\int_0^{x_i+d} \sum_{\substack{k=1 \\ k \neq i}}^{m} \prod_{j \neq i,k} F_{X_j}(x_k) f_{X_k}(x_k) dx_k \right] f_{X_i}(x_i) dx_i \\
&= \int_0^\infty \left[\prod_{j \neq i} F_{X_j}(x+d) \right] f_{X_i}(x) dx \qquad (9)
\end{aligned}$$

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

$$F_C(t) = \prod_{i=1}^{m} F_{X_i}(t). \qquad (10)$$

We assume 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 we only consider the fitted ED. The formulae will be more complex for BED. In fact, the result would be more accurate if we use Coaxian approximation for the distributions because the distributions of D's generally have a mass at zero which is not captured by ED or BED.

Let the fitted ED for $X_i, i = 1 \dots m$ have k_i stages, each with mean service rate of μ_i. We first find the mean and variance of C_i which is the cycle time distribution of the net Σ_1 without place p_i. That is, $F_{C_i}(t) = \prod_{\substack{j=1 \\ j \neq i}}^{m} F_{X_j}(t)$. Its distribution is then fitted by an ED which is used in equation (8) to find the mean and variance of D_i. Assume that $F_C(t) = F_{X_1}(t) F_{X_2}(t)$. Then first two moments of C can be computed as (p is defined as $3 - i$),

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

$$E[C^2] = \qquad (12)$$
$$\sum_{i=1}^{2} \frac{k_i(k_i+1)}{\mu_i^2} \left[1 - \sum_{j=0}^{k_p - 1} \binom{k_i + 1 + j}{j} \left(\frac{\mu_i}{\mu_1 + \mu_2} \right)^{k_i + 2} \left(\frac{\mu_p}{\mu_1 + \mu_2} \right)^j \right]$$

from which we approximate C's distribution by an ED. We can now apply equations (10) and (11) recursively to compute the fitted ED for C_i. That is, if we define $G(1, t) = F_{X_{i_1}}(t)$, and

$$G(n, t) = G(n-1, t) F_{X_{i_j}}, \quad j = 2 \dots m-1, i_j \neq i. \qquad (13)$$

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

Similar formulas can be derived for $E[D_i]$ and $E[D_i^2]$ using the fitted ED of C_i in (8). Notice that if we are only interested in the mean of D_i we could continue the recursion (12) one more time and find the mean value of the net cycle time, C, then $E[D_i] = E[C] - E[X_i]$.

The approximate algorithm for a SDFPN 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.

4 Experimental Validation of Approximate Algorithm

A series of experiments were run to evaluate the accuracy of the approximate solution presented in the paper. Four SDFPN's were chosen to test the algorithm, each with different characteristics, one of them is shown in Figure 3. For each net, several test data with different delays were created. We have used the branching Erlang distribution to fit all the distributions with different coefficient of variations. The total number of tests performed was 20.

As expected, the largest errors occurred when most delays were near the threshold. The largest error

1. Using the set of equations $Qd = 0$, for each transition t_i find $r = |*t_i|$ simultaneous equations of form

$$g_{i_1}(X, D) + D_{i_1} = g_{i_2}(X, D) + D_{i_2} = \ldots = g_{i_r}(X, D) + D_{i_r} \tag{8}$$

where $p_{i_j} \in^* t_i$ and

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

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 $i = 1 \ldots n$.
Repeat the following step until mean values for D's converge.

   (a) For each transition t_i, compute the means and variances of g_{i_j} as

$$\begin{aligned} E[g_{i_j}] &= \sum_{i=k}^{m} a_{i_k} E[X_k] + \sum_{\substack{i=k \\ k \neq j}}^{m} b_{i_k} E[D_k] \\ Var[g_{i_j}] &= \sum_{i=k} m a_{i_k} Var[X_k] + \sum_{\substack{i=k \\ k \neq j}}^{m} b_{i_k} Var[D_k] \end{aligned}$$

   (b) Treat g_{i_j}'s as the X's for transition t as for the simple net Σ_1 and use the exact solution techniques, equations (10), (11), and (8) to compute the new values for $E[D_i]$ and $Var[D_i]$.

Algorithm 2. Approximate Algorithm.

in mean synchronization delay was 16% but that in standard deviation was 27%. The mean errors over the all these tests were found to be 9% and 15% respectively. These figures are with respect to the results of simulation done using PAWS package [7]. We believe these range of errors acceptable for most purposes.

Dependencies among the delays are the main source of error in this approximation. We assumed D's and X's to be independent when computing the variance of g's. Moreover, equations (8), (10), and (11) assume independent X's which is not true in general for g's. Nevertheless, no convergence problems were observed in all verification runs.

5 Conclusions

We have presented an exact solution technique for any safe and live stochastic decision free Petri nets 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 and thus does not suffer from state space explosion and does not demand 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 SDFPN's 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 for other existing approaches. Finding the regions in which each of the N cases can occur is also an expensive procedure which is called once in an iteration. Identifying the "impossible" cases beforehand and 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 section 4. The proposed approximations can give reasonably accurate results without incurring exponential complexity, however, it is difficult to analyze the error introduced by them.

In order to compute actual delays suffered by processes, we must consider resource contention and interprocess communication simultaneously. This can be done by solving the contention and communication models iteratively, one feeding the other. Note that a 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] M. Ajmone-Marson, G. Balbo, and G. Conte. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer systems*, pages 93–122, Jan 1984.

[2] F. Baccelli and Z. Liu. Comparison properties of stochastic decision free petri nets. *IEEE Transactions on Automatic Control*, 37(12):1905–1920, Dec. 1992.

[3] F. Commoner and A. W. Holt. Marked directed graphs. *Journal of Computer and System Science*, pages 511–523, 1971.

[4] J. B. Dugan, K. S. Trivedi, R. M. Geist, and V. F. Nocola. Extended stochastic Petri nets: Applications and analysis. In E. Gelenbe, editor, *performance '84*, pages 75–83. North-Holland, 1984.

[5] M. Ghodsi. Modeling performance of distributed programs by stochastic decision free Petri nets. Technical report, Computer Engineering Dept, Sharif University of Technology, Oct. 1998.

[6] P. Heidelberger and K. Trivedi. Queueing network models for programs with internal concurrency. *IEEE Transactions on Computer*, pages 73–82, Jan. 1983.

[7] Information Research Associates, Austin, Texas. *PAWS: Performance Analyst's Workbench System*, 1986.

[8] K. Kant. *Introduction to Computer System Performance Evaluation.* McGraw-Hill Inc., 1992.

[9] K. Kant. Modeling interprocess communication in distributed programs. *International workshop on Petri Nets and Performance Models, Madison, WI*, pages 75–83, Aug. 1987.

[10] Y. Lin and C.M. Woodside. Iterative decomposition and aggregation of stochastic marked graph petri nets. In *Twelfth International Conference on Applications and Theory of Petri Nets*, pages 257–275, Gjern, Denmark, June 1991.

[11] B. Balbo M. Sereno. Computational algorithms for product form solution stochastic petri nets. In *Proc. Fifth Int'l Workshop on Petri Nets and Performance Models*, Toulouse France, Oct. 1993.

[12] M. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computer*, pages 913–917, Sept. 1982.

[13] T. Murata. Circuit theoric analysis and synthesis of marked graphs. *IEEE Transactions on circuits and systems*, pages 400–405, July 1977.

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

[15] C. A. Petri. Communication with automata. Technical Report RADC-TR-65-377, New York, 1966. original in German; Kommunication mit automaten, Univ. of Bonn (1962).

[16] S. Raida. A markov model for distributed communication sequential processes. *Performance 84, Elsevier, North Holland*, pages 3–16, 1884.

[17] C. Ramachandani. *Analysis of asynchronous concurrent systems by timed Petri nets.* PhD thesis, Lab. of Computer Science, MIT, Cambridge, MA, 1974. (tech report# TR-120).

[18] C. V. Ramamoorthy and G. S. Ho. Performance evaluation of asynchronous concurrent systems using Petri nets. *IEEE Transactions on Software Engineering*, pages 440–449, Sept 1980.

[19] Matteo Sereno. Approximate mean value analysis for stochastic marked graphs. *ieeetse*, 22(9):654–664, Sept. 1996.

[20] J. Sifakis. Use of Petri nets and preliminary performance evaluation. In Beimer and Gelenbe, editors, *Measuring, Modeling and evaluating computer systems*, pages 75–93. North-Holland, 1977.

[21] P.G. Taylor W. Henderson. Generalization of queueing network product form solution to stochastic petri nets. *ieeetse*, 17(2):108–116, Feb. 1991.