DISCRETE EVENT SIMULATION OF DISTRIBUTED SYSTEMS USING STOCHASTIC PETRI NETS

Y. Narahari, K. Suryanarayanan and N.V. Subba Reddy

Department of Computer Science and Automation
Indian Institute of Science
Bangalore 560 012, INDIA

ABSTRACT

Stochastic Petri Nets (SPNs) have recently emerged as a principal performance modelling tool for distributed systems such as multiprocessors, local area networks, and automated manufacturing systems. Since the use of SPNs as an analytical tool is based on the generation of the entire state space, the technique becomes intractable for large systems. In such cases, discrete event simulation is the preferred tool for performance evaluation. In this paper, we show how SPNs can be used as a simulation model. We present several efficient algorithms based on SPNs, for conducting discrete event simulations of distributed systems.

1. INTRODUCTION

Performance modelling and evaluation constitute an important aspect of the design of distributed systems such as multiprocessors, local area networks, and automated manufacturing systems. Performance models are mainly of two types: analytical and simulation. Analytical models such as Markov chains, queueing networks, and stochastic Petri nets (SPNs) are excellent for a quick, approximate evaluation of performance but become intractable if a detailed evaluation of a large system is required. Simulation is typically used in such contexts.

Imagine the following scenario: A performance model, based on SPNs, has been constructed for a given distributed system. We find that this model is intractable (that is, very difficult to analyze) Any simplifications in the SPN model that would make it tractable are unacceptable and simulation is the only alternative. In this situation, it will be congenial to use the same SPN model and carry out a simulation of the SPN model. With this as the motivation, we show in this paper that SPNs indeed lead to an elegant model of discrete event simulation.

In Section 2, we present an informal introduction to SPNs with an illustrative example. In Section 3, we first discuss how an SPN embeds all information necessary for a discrete event simulation. Next, we outline some important SPN-based algorithms that can be used in discrete event simulation. Finally we show the basic steps in simulation output analysis using SPNs. In section 4, we present some conclusions.

2. STOCHASTIC PETRI NETS

Petri nets [1] have emerged as a prominent modelling tool of concurrent systems. A class of timed Petri nets called generalized stochastic Petri nets (GSPNs) [2] are well suited for performance modelling. In the framework of GSPNs and Extended stochastic Petri nets [3], several features of distributed systems such as concurrency, non-determinism, and synchronization can be captured in an elegant way.

Informally, any SPN comprises a set of places, a set of transitions, a set of arcs, an initial marking, and an assignment of random variables to transitions. In the SPN-representation of a distributed system, places represent logical conditions or resources in the system; transitions represent events or activities; arcs represent interdependencies among places and transitions; initial marking refers to the initial state of the system; and the random variables model the durations of various activities in the system. The evolution of an SPN in time constitutes a stochastic process called the marking process of the SPN. The use of SPNs as an analytical model is based on a steady-state analysis of the marking process. In this paper, we look at situations where the SPN model is huge enough to make this steady-state analysis intractable and simulation is the only alternative.

We first illustrate the construction of an SPN model of a simple queueing network model. Figure 1 depicts a queueing network model popularly called the closed central server network model [4]. This model has three queues, Q_0, Q_1 and Q_2, corresponding to three resources. This model has been used to describe a multiprogrammed operating system with one CPU and two input/output devices [4]. Another interpretation for this model is that of an automated manufacturing system comprising an AGV (automated guided vehicle) and two machines M_1 and M_2 [5]. We shall use the latter interpretation in this paper.

Figure 2 shows a GSPN model for the queueing network under study. The circles in the diagram are called places and represent conditions or resources in the system. There are 10 places in the model. Their interpretation is given in Table 1. The horizontal bars t_1, t_3, t_4, t_5, t_6, and t_7 are called immediate transitions and represent logical changes in the system. The rectangular bars t_2, t_8, and t_9 are called exponential transitions and represent timed activities such as processing by machines and transportation by AGV. These three transitions are associated with exponential random numbers with rates μ_0, μ_1, and μ_2 respectively. The interpretation of both types of transitions are also given in Table 1. The presence of black dots (called 'tokens') inside the places p_2, p_9, and p_{10} indicates that the AGV is free, a part is getting processed by M_1, and the other part is getting processed by M_2. This is the initial state or initial marking of the system (the words 'state' and 'marking' are used interchangeably in the sequel). Note that the immediate transitions t_3, t_4, and t_5 are conflicting in the sense that only one of them fires at any point of time, disabling the others. The probability with which each can fire is specified by the probability distribution $\{q_0, q_1, q_2\}$ defined on this set of transitions.

The dynamic evolution of a GSPN model constitutes a stochastic process, called the marking process. It has been shown in the case of GSPNs, that the marking process is a semi-Markov process [2]. Haas and Shedler [6,7] have shown that the marking process of a general

CH2766 – 4/89/0000 - 0622

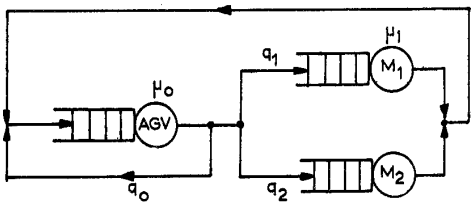


Figure 1. A closed Queueuing Network Model.

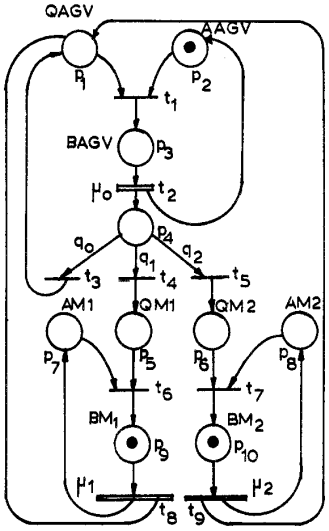


Figure 2. GSPN Model of the Queueing Network in Fig. 1.

Places:

1	: Queue of parts waiting for the AGV
2	: AGV available
3	: AGV transporting a part
4	: A part that has just been transported by the AGV
5	: Queue of parts waiting for M1
6	: Queue of parts waiting for M2
7	: M1 available
8	: M2 available
9	: M1 processing a part
10	: M2 processing a part

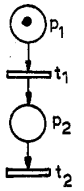
Immediate Transitions:

1	: AGV starts transporting a part
3	: Finished part gets unloaded
4	: Part joins the queue for M1
5	: Part joins the queue for M2
6	: M1 starts processing a part
7	: M2 starts processing a part

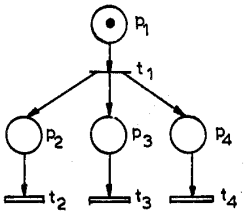
Exponential Transitions:

2	: μ_o	: Part transfer by the AGV
8	: μ_1	: Processing by M1
9	: μ_2	: Processing by M2

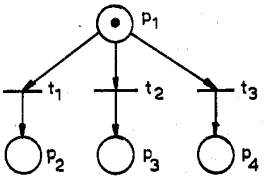
Table 1. Description of the GSPN model



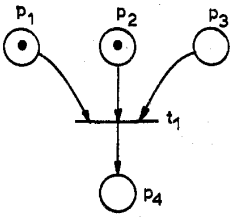
3(a) Sequential Execution



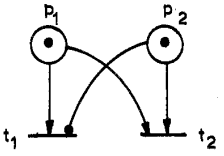
3(b) Concurrency



3(c) Conflict



3(d) Synchronization



3(e) Priorities

Figure 3. Petri Net Representation

class of SPNs is a GSMP (generalized semi-Markov process).

3. DISCRETE EVENT SIMULATION USING SPNs

3.1 SPN as a Simulation Model

The usual model for the underlying stochastic process of a discrete event simulation is a generalized semi-markov process (GSMP) [6]. The marking process of an SPN model is also a GSMP. Thus the description power of an SPN model is the same as that of a discrete event simulation. The places of an SPN model represent the resources or conditions in the physical system while transitions represent events or activities in the system. The transitions of an SPN naturally map onto the events of a discrete event simulation. The set of all reachable markings of an SPN gives the state space modelled by a discrete event simulation.

An SPN model can elegantly capture the features of a distributed system, such as concurrency, communication and synchronization. Figures 3(a) - 3(e) depict Petri net representations for sequential execution, concurrency, conflict, synchronization, and priorities. The SPN model of a distributed system will typically involve all of these features. In Figure 3(a), there are two transitions t_1 and t_2, of which t_1 is enabled. When t_1 fires, a token is deposited in p_2, enabling t_2. Thus the execution of t_1 and t_2 follows a precedence relation, which indicates sequential execution. In figure 3(b), t_1 is enabled and its firing results in a token in each of the places p_2, p_3, and p_4. Now all the three transitions t_2, t_3, and t_4 are enabled and this represents three concurrent activities. The situation in figure 3(c) is that the transitions t_1, t_2, and t_3 are all enabled but only one of them can fire because of a single token in p_1. Here, the firing of t_1, for example, will disable t_2 and t_3. The above three are conflicting activities. The conflict is often resolved by associating a probability distribution to the set $\{t_1, t_2, t_3\}$. This was what we did in the case of the GSPN model of figure 2 (see transitions t_3, t_4, and t_5) in figure 2. In figure 3(d), t_1 is not enabled because p_3 does not have a token. The enabling of t_1 now awaits the arrival of a token in p_3, which amounts to synchronization. In Figure 3(e), there is a special arc called the inhibitor arc from p_2 to t_1. As a consequence, t_1 fires if there is a token in p_1 and there is no token in p_2. This provides a way of giving priority to transition t_2 over transition t_1.

3.2 SPN-based Algorithms

Here, we discuss the following two problems:

Algorithm 1: Determining the set of events that will fire next, given the current set of enabled events.
Algorithm 2: Computing the next set of enabled events, given (a) the current state and (b) the current set of events that will fire.

SPNs can be used in a natural way because the concept of transitions and firing rules in SPNs closely models what happens in a discrete event simulation.

In Algorithm 1, given the set of enabled immediate transitions in the current state, we first compute the sets of conflicting transitions and concurrent transitions among these. We select one transition in every conflict set and all concurrent transitions, to fire next. There could be two problem cases here: (a) How to select a transition if two or more conflict sets are not disjoint. (b) How to select a transition in the case of 'confusion' (a situation where concurrency and conflict are both present in a special way; see reference [8] for more details). Owing to space constraints, this algorithm is not given here. The complete algorithm may be found in [9].

In Algorithm 2, we are given the current state and the current set of transitions selected to fire next. In the SPN framework, this algorithm can be made efficient by restricting the checking of enabledness to only the output transitions of the input places and the output places of all the firing transitions. The details of this algorithm can be found in [9].

3.3 Generic Performance Measures

We discuss here the computation of performance measures of interest in SPN based simulation. Simulation is carried out for a given number of transition firings. Each time a transition fires, the state of the system changes. In terms of the Petri net model, the changes that occur when a transition fires are: (1) The input places of the transition lose tokens. (2) The output places of the transition gain tokens. (3) The clock in the simulated system advances by the firing time of the transition. (4) The total number of times this transition has fired gets incremented by 1.

In the simulation, each time a transition fires, the above changes are monitored and stored in suitably chosen data structures. At the end of the simulation, performance measures are computed by looking at the values contained in these data structures.

Data Structures

a) Global Clock: This is a global real variable with initial value zero. It gets incremented each time a timed transition fires, by an amount equal to the firing time of the transition. When an immediate transition fires, the global clock is not incremented.

b) Firings: 'Firings' is a vector with one element for every transition. For transition t_i, Firings $[t_i]$, at any given point in simulation, gives the total number of times t_j has fired from the start of the simulation.

c) Tokenloss: 'Token loss' is a vector with one element for every place. For place p_i, tokenloss $[p_i]$ gives the total number of times p_i has lost a token during the simulation so far.

d) Tokentime: 'Token time' is a two dimensional array where the first dimension corresponds to the places and the second dimension to numbers of tokens. For example, if there are 50 places in the model and the maximum number of tokens that can reside in any place is 10, we define token time [1:50, 0:10]. Token time [23,5], for example, would give the total amount of time in the simulation to far, for which places p_{23} had exactly 5 tokens.

e) Marktime: 'Mark time' is a vector with one element for every marking in the Petri net. If M_k is a marking (state), Mark time (k) would give the total time for which the system has spent in marking M_k, so far.

f) Tokenfultime: 'Tokenfultime' is a vector with one element for every place. Tokenfultime (p_i) gives the total time so far for which p_i has contained at least one token.

Performance Measures

We describe below how generic performance measures

can be defined. Specific performance measures for the given system can be computed from these.

a) Steady state probabilities of markings: Let M_k be a marking. Then the steady state probability of M_k is given by

$$SSP(M_k) = marktime(k)/globaltime$$

b) Probability that a condition is satisfied in the system: Let C be a condition (such as system working, bus busy, machine idle, so on), and let PROB(C) be the required probability, then

$$PROB(C) = \left(\sum_{M_k \epsilon S(C)} marktime(k) \right)/(globaltime)$$

The summation is over the set S(C) which is the set of all markings in which the condition is satisfied.

c) Probability that place p_i has exactly k tokens: The required probability is given by

$$PROB(P_i,k) = (tokentime[i,k])/(globaltime)$$

d) Expected number of tokens in place p_i: The required measure is given by

$$ET(p_i) = \sum_{k=1}^{\infty} k.\ PROB(p_i,k)$$

e) Average waiting time in place p_i:

$$WAIT(p_i) = (tokenfultime[i])/(tokenloss[i])$$

f) Throughputrate of a transition t_j: This is the expected number of times a transition fires per unit time and is given by

$$TR(t_j) = firings[j])/(globaltime)$$

4. CONCLUDING REMARKS

In the context of discrete event simulation, we have shown that SPNs provide a natural model because transitions represent the events and the SPN incorporates a considerable amount of information about the set of events that can occur when the process is in a particular state and about the sets of 'new events' and 'old events' when a transition fires in a particular state. Also, the marking process of an SPN is a generalized semi-Markov process which is the usual model for the underlying stochastic process of a discrete event simulation.

A software package has been developed in the language C, for discrete event simulation based on SPNs. This package incorporates all of the techniques discussed in Section 3. For details of the algorithms and the implementation, the reader is referred to [9,10].

REFERENCES

1. J.L. Peterson, 'Petri Net Theory and the Modelling of Systems', Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1981.
2. M.A. Marsan, G. Balbo, and G. Conte, 'A Class of Generalized Stochastic petri Nets for the Performance Evaluation of Multiprocessor Systems', ACM Transactions on Computer Systems, Vol. 2, No. 2, pp. 93-122, May 1984.
3. J.B. Dugan, K.S. Trivedi, R.M. Geist, and V.F. Nicola, 'Extended Stochastic Petri Nets: Applications and Analysis', Performance '84', pp. 419-441, 1984.
4. K.S. Trivedi, 'Probability and Statistics with Reliability, Queueing and Computer Science Applications', Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.
5. Y. Narahari and N. Viswanadham, 'Performance Modelling of Flexible Manufacturing Systems', To appear in: Journal of IETE, 1989.
6. P.J. Haas and G.S. Shedler, Regenerative Simulation Methods for Local Area Networks, IBM Journal of Research and Development, Vol. 29, pp. 194-205 1985.
7. P.J. Haas and G.S. Shedler, Stochastic Petri Net Representation of Discrete Event Simulations, IBM Research Report RJ 5646 (57145), July 1987.
8. M.A. Marsan, G. Balbo, G. Chiola, and G. Conte, 'GSPNs Revisited: Random Switches and Priorities', Proc. Intl. Workshop on Petri Nets and Performance Models, Madison, pp. 44-53, August 1987.
9. K. Surianarayanan, 'Petri Net based Algorithms for Discrete Event Simulation', B.E. Project Report, Dept. of CSA, I.I.Sc., April 1989.
10. N.V. Subba Reddy, 'Discrete Event Simulation using Petri Nets', M.E. Project Report, Dept. of CSA, I.I.Sc., 1989.