Sādhanā, Vol. 11, Parts 1 & 2, October 1987, pp. 187–208. © Printed in India.

Performance modelling of a fault-tolerant real-time multiprocessor using stochastic Petri nets

Y NARAHARI and N VISWANADHAM

Department of Computer Science & Automation, Indian Institute of Science, Bangalore 560 012, India

Abstract. The fault-tolerant multiprocessor (FTMP) is a bus-based multiprocessor architecture with real-time and fault-tolerance features and is used in critical aerospace applications. A preliminary performance evaluation is of crucial importance in the design of such systems. In this paper, we review stochastic Petri nets (SPN) and develop SPN-based performance models for FTMP. These performance models enable efficient computation of important performance measures such as processing power, bus contention, bus utilization, and waiting times.

Keywords. Real-time control; fault-tolerant multiprocessors; performance modelling; queueing networks; stochastic Petri nets.

1. Introduction

The principal contribution of this paper is in conducting the performance evaluation of FTMP, a fault-tolerant real-time multiprocessor for air traffic control, by constructing stochastic Petri net (SPN) based performance models. This paper also purports to be a survey on the SPN modelling technique for performance evaluation.

Performance evaluation is an indispensable part of the design of computer systems. There are mainly three techniques which have been extensively used in performance evaluation: simulation, queueing networks, and stochastic Petri nets. Simulation is a powerful tool but requires enormous computation to yield accurate performance estimates. Queueing networks and SPN are analytical modelling tools and are much more efficient than simulation for approximate performance prediction. Efficient computational methods are available for a class of queueing networks called product form queueing networks (PFQN) but the use of PFQN entails several restrictive assumptions to be made on the system being modelled. SPN are in this sense more versatile than PFQN. The focus in this paper is on SPN based performance evaluation of multiprocessors, which is now an important topic of research (Marsan *et al* 1986). In particular, we consider the FTMP system.

FTMP (fault-tolerant multiprocessor) (Hopkins *et al* 1978) is a highly reliable, fault-tolerant, real-time multiprocessor which embodies many innovations of the current research in fault-tolerant computing and distributed computing. FTMP is a

bus-based architecture and has been proposed as the central computer for civil transport aircraft applications. An engineering prototype of FTMP was developed by the Charles Stark Draper laboratory and installed at the NASA Langley Research Center. It is designed to have a failure rate of the order of 10^{-10} failures per hour on ten-hour flights where no airborne maintenance is available. In view of the critical applications for which FTMP is used, a preliminary evaluation of its performance and reliability assumes tremendous importance. Shin *et al* (1985) have addressed some vital performance issues of FTMP using a closed queueing network (CQN) model and have derived useful performance measures such as processing power, bus utilization, bus contention, and waiting times, under specified work load conditions. In this paper, we develop performance models of FTMP based on generalized stochastic Petri nets (GSPN). We show how the GSPN models overcome some of the inadequacies of the CQN model and lead to a more realistic and accurate description of the FTMP architecture.

We first present in § 2, a comprehensive survey of stochastic Petri nets to make the paper self-contained. In § 3, we briefly review the FTMP architecture and the CQN model of FTMP. In § 4, we present several GSPN models for FTMP. We also suggest many improvements over these models and raise some theoretical questions for future investigation. The GSPN models presented in this paper have been analysed using a software package developed by us at the Indian Institute of Science. This package is coded in Pascal and runs on a DEC-1090 system.

2. Stochastic Petri nets

Stochastic Petri nets (SPN) have recently emerged as a powerful modelling and performance evaluation tool for concurrent systems. SPN are an outgrowth of timed Petri nets (TPN) which in turn have evolved from the classical Petri nets. In this section, we present an introduction to the performance evaluation methodology based on SPN. We first present essential details of classical Petri nets. Petri nets were first proposed by Carl Adam Petri (Petri 1966).

2.1 *Petri nets*

Definition 1: *Petri net.* A Petri net is a quadruple (P, T, A, M_0) where

$P = \{p_1, p_2, \ldots, p_n\}$ is a set of places,
$T = \{t_1, t_2, \ldots, t_m\}$ is a set of transitions,
$A \subseteq (P \times T) \cup (T \times P)$ is a set of arcs,
$M_0 : P \rightarrow \mathcal{N}$ is a mapping called initial marking that associates zero or more tokens to each place. $\mathcal{N}$ is the set of all non-negative integers.

Further, $m \geq 0$, $n \geq 0$, $m+n \geq 1$, and $P \cap T = \emptyset$.

In a Petri net model of a given system, places represent conditions, resources or buffers, while transitions represent events (activities) or event epochs. Tokens signify the truth value of conditions or individual resources or individual buffers. Arcs indicate the various types of dependencies between places and transitions. Initial marking represents the initial state of the system.

Generally, places are represented by circles, transitions by bars, and tokens by black dots or integer labels. Arcs are shown by directed arcs.

Definition 2: *Input places and output places.* If t is any transition of a Petri net (P, T, A, M_0), we define the set of input places of t by

$$IP(t) = \{p \in P: (p,t) \in A\}$$

and the set of output places of t by

$$OP(t) = \{p \in P: (t,p) \in A\}.$$

Definition 3: *Enabling and firing of transitions.* Let $M: P \rightarrow \mathcal{N}$ be a marking of a Petri net (P, T, A, M_0). A transition $t \in T$ is said to be enabled in marking M iff

$$M(P) \geq 1 \ \forall \ p \in IP(t).$$

A transition t enabled in a marking M can 'fire'. When t fires, the marking of the Petri net changes to M' where M' is given by

$$\begin{aligned} M'(p) &= M(p) - 1, \ p \in IP(t), \\ &= M(p) + 1, \ p \in OP(t), \\ &= M(p), \text{ otherwise.} \end{aligned}$$

We say in this case that M' is immediately reachable from M and we write $M \xrightarrow{t} M'$.

We assume in this paper that the firing of two different transitions in a given marking cannot lead to the same marking. That is,

$$M \xrightarrow{t_1} M' \text{ and } M \xrightarrow{t_2} M' \Rightarrow t_1 = t_2.$$

Definition 4: *Reachability set.* Let (P, T, A, M_0) be a Petri net. A marking M' is said to be reachable from M if there exists a sequence of transitions by firing which we can obtain M' from M. Reachability of markings is a reflexive and transitive relation and the transitive closure of this relation is called the reachability set of the Petri net.

The reachability set is denoted by $R[M_0]$ and comprises all markings reachable from M_0 in zero or more steps.

Definition 5: Reachability graph. Let $R[M_0]$ be the reachability set of a Petri net (P, T, A, M_0). The reachability graph is a directed graph (V, E) where $V = R[M_0]$ is the set of vertices and E is the set of directed arcs defined by

$(M_1, M_2) \in E$ iff there exists a transition 't' such that $M_1 \xrightarrow{t} M_2$.

Definition 6: *Conflicting and concurrent transitions.* In a Petri net (P, T, A, M_0), given any two transitions t_1 and t_2, we say t_1 and t_2 are conflicting if $IP(t_1) \cap IP(t_2) \neq \emptyset$. Otherwise, they are said to be concurrent.

The above definition can be generalized to more than two transitions in the natural way. Petri nets capture non-determinism through conflicting transitions and concurrency through concurrent transitions and can also represent elegantly the co-existence of non-determinism and concurrency.

2.2 *Stochastic Petri nets*

Classical Petri nets are useful in investigating qualitative properties of concurrent systems such as boundedness, existence/absence of deadlocks, and mutual exclusion. However they cannot be used for quantitative performance evaluation. By introducing time into the definition of a Petri net, several researchers have proposed timed Petri nets (TPN) for performance evaluation of concurrent systems (see Ramchandani 1973, Sifakis 1977, and Ramamoorthy & Ho 1980). Stochastic

Petri nets (SPN) are a recently proposed special class of timed Petri nets in which transitions or places are associated with stochastic time durations. SPN with timed places are equivalent to SPN with timed transitions and in this paper, we shall consider the latter class of SPN.

Definition 7: *Stochastic Petri net.* An SPN is a quintuple (P,T,A,M_0,F) where (P,T,A,M_0) is a Petri net and F is a mapping with domain $R[M_0]\times T$. In each $M\in R[M_0]$, F associates with each transition $t\in T$, a firing time which is a continuous random variable. The firing times are all independently distributed.

Note from the above definition that the random variable associated with a transition is in general marking-dependent. In an SPN, when a transition t is enabled in a marking M, the tokens remain in the input places of t during the firing time of t. At the end of the firing time, a token is removed from each input place of t and a token is deposited in each output place of t. When a transition t gets enabled, we say 't starts firing' and when the firing time has elapsed, we say 't finishes firing' or just say 't fires'. It is however possible that t gets disabled sometime before finishing firing due to the firing of a conflicting transition. Also, no two concurrently enabled transitions of an SPN can finish firing simultaneously.

SPN were first proposed by Natkin (1980) and Molloy (1981) in independent proposals. SPN and their extensions have been used in the performance study of multiprocessors (Marsan *et al* 1984; Holliday & Vernon 1985; Marsan *et al* 1986), priority queueing disciplines in time-shared computer systems (Balbo *et al* 1986; Zuberek 1985), fault-tolerant computer systems (Meyer *et al* 1985; Dugan *et al* 1984, pp. 507–519), and several other concurrent systems.

Definition 8: *Marking process of SPN.* Let (P,T,A,M_0) be an SPN. Let $X(t)$ represent the marking of the SPN at time $t\geq 0$ and let $X(0)=M_0$. Then $\{X(t),t\geq 0\}$, is a stochastic process called the marking process of the SPN.

The basic philosophy underlying the use of various classes of SPN in performance evaluation is the equivalence of their marking process to a Markov or semi-Markov process with discrete state space. The typical steps in SPN-based performance evaluation include: (1) modelling the given system by an SPN (2) deriving the stochastic process underlying the SPN model and computing steady-state distribution of the stochastic process, and (3) obtaining the required performance measures from the solution of the stochastic process. All steps in the SPN based performance evaluation can be automated and this factor is one of the key advantages of the SPN. Also, SPN are graphical models of concurrency, randomness, and synchronization and SPN models can be constructed in a natural way from a description of the system components and interactions. Moreover, there are certain features called non-product form features which are captured nicely by SPN but cannot be represented by efficiently solvable queueing networks (Balbo *et al* 1986). These advantages have established SPN as a principal modelling tool for performance evaluation. Several software packages have been developed for performance evaluation using SPN (*IEEE* 1985).

In this paper, we survey two widely used classes of SPN–Exponential timed Petri nets (ETPN) proposed by Natkin (1980) and Molloy (1981) and Generalized stochastic Petri nets (GSPN) developed by Marsan *et al* (1984). In the sequel, the definitions and results presented are taken from the key papers of Natkin (1980),

Molloy (1981), and Marsan *et al* (1984, 1985). We have made some minor changes in notation to suit the style of the paper. Also, we use the title 'Exponential timed Petri nets' because it is more suggestive.

2.3 *Exponential timed Petri nets*

Definition 9: *Exponential timed Petri nets* (*ETPN*). An ETPN is an SPN in which the firing time of each transition in every marking is an exponential random variable. Formally, an ETPN is a quintuple (P, T, A, M_0, F) where (P, T, A, M_0) is a Petri net and F: $R[M_0] \times T \rightarrow \mathcal{R}$ is a function that associates with each $(M,t) \in R[M_0] \times T$, a real number ($\mathcal{R}$ is the set of all real numbers). $F(M,t)$ for each $M \in R[M_0]$ and $t \in T$ is to be interpreted as the rate of exponential random variable associated with transition t.

The reachability set of an ETPN is the same as that of the corresponding Petri net. However the reachability graphs of the two are different in one respect: if M_1 and M_2 are two markings such that M_2 is reached by the concurrent firing of two transitions t_1 and t_2, then in the case of a classical Petri net, there will be an arc from M_1 to M_2 in the reachability graph while such an arc will not exist in the case of an ETPN.

2.3a *Analysis of an ETPN*: Let $\{X(t), t \geq 0\}$ be the marking process of the ETPN (P, T, A, M_0, F). The marking process of an ETPN is a continuous time Markov chain (CTMC). Hence the theory of Markov chains can be used to analyse an ETPN. The CTMC associated with an ETPN has the following characteristics.
(1) The state space of the CTMC is precisely the reachability set of the ETPN.
(2) The state transition probability matrix (TPM) of the embedded Markov chain (EMC) of the CTMC is computed as follows.

Let M_i, $M_j \in R[M_0]$ and suppose p_{ij} is the element of TPM corresponding to M_i and M_j. If M_j is not immediately reachable from M_i, then $p_{ij} = 0$. Otherwise, let t_{i_1}, $t_{i_2}, \ldots, t_{i_{n_i}}$ be the transitions enabled in M_i and let

$$M_i \xrightarrow{t_{i_k}} M_j \text{ for some } k \in \{1,2,\ldots,n_i\}.$$

Then

$$p_{ij} = \lambda_{i_k} \Big/ \sum_{l=1}^{n_i} \lambda_{i_l},$$

where λ_{i_l} is the firing rate of the transition t_{i_l} in the marking M_i.
(3) The sojourn time of each marking M (amount of time the marking process stays in M) is an exponentially distributed random variable with rate equal to the sum of the rates of those of the enabled transitions in M which lead to a marking other than M on firing.

By solving the above CTMC using standard techniques (Ross 1983), one can obtain the steady state probability distribution of the marking process. These steady state probabilities can be used in the computation of generic performance measures such as (1) probability that a given place has a token, (2) mean number of tokens in a given place, and (3) mean number of firings of a transition in unit time.

2.4 *Generalized stochastic Petri nets*

Definition 10: *Generalized stochastic Petri nets* (*GSPN*). A GSPN is a quintuple (P, T, A, M_0, F) where (P, T, A, M_0) is a Petri net, T is partitioned into two sets T_I and T_E, and F is a firing time function which gives the rate of the exponential random variable associated with each $t \in T_E$ in each reachable marking of the GSPN.

The elements of T_I are called immediate transitions and they have a firing time equal to zero in all markings. The elements of T_E, which we call exponential transitions, have an exponentially distributed firing time in each marking of the GSPN. In the graphical representation of a GSPN, a horizontal line represents an immediate transition and a rectangular bar represents an exponential transition.

2.4a *Firing rules for a GSPN*: GSPN markings are of two types: those in which at least one immediate transition is enabled are called vanishing markings (so called because immediate transitions fire in zero time) and those in which only exponential transitions are enabled are called tangible markings. The firing of transitions in a tangible marking is on the same lines as in ETPN. In a vanishing marking, the following firing rules are followed.

1. The enabled exponential transitions are not fired and only the enabled immediate transitions are allowed to fire.
2. If two or more concurrent immediate transitions are enabled, all of them fire simultaneously.
3. If some of the enabled immediate transitions are conflicting, only one of them is allowed to fire at a time, according to predefined probability distribution. Such distributions are called random switches and they contribute significantly to the modelling power of GSPN (Marsan *et al* 1984).

2.4b *Reachability graph of a GSPN*: The reachability set of a GSPN (P, T, A, M_0, F) is a subset of that of the underlying Petri net. This is because of the following reasons.

(1) Immediate transitions are given priority over exponential transitions.
(2) Concurrently enabled immediate transitions fire simultaneously and this eliminates several intermediate states.

The reachability graph of a GSPN can be constructed from its reachability set in the usual way.

2.4c *Analysis of a GSPN*: Marsan *et al* (1984) have shown that the marking process of a GSPN is a stochastic point process with discrete state space. The marking process of the GSPN is assured of having a steady state probability distribution under the following conditions.

1. The underlying Petri net is bounded (i.e., the number of reachable markings is finite).
2. The underlying Petri net is proper (i.e., the initial marking is reachable from all reachable markings).
3. The firing rates of the exponential transitions do not depend on the time of observation.

Marsan *et al* (1984) have presented an efficient solution technique for GSPN satisfying the above conditions. We outline the various steps in this procedure. The GSPN models presented in this paper satisfy all these conditions and can be analysed using the following steps.

1. The reachability set of the GSPN is determined.
2. The embedded Markov chain (EMC) of the GSPN has precisely the same states as in the reachability set. The transition probability matrix (TPM) of the EMC is computed. The entries in the TPM corresponding to vanishing states are computed using the predefined random switches and the entries corresponding to tangible states are obtained as in the case of ETPN.
3. The EMC comprises both vanishing and tangible states. Since the marking process stays for zero time in each vanishing marking, we do not require any information about the vanishing states. A reduced embedded Markov chain (REMC) that comprises only tangible states is derived from the EMC. The TPM of the REMC can be efficiently computed using a technique developed in Marsan *et al* (1984). Let $\{M_0, M_1, \ldots, M_t\}$ be the set of all tangible states. The TPM of the REMC will then be of order $(t+1)$. Let P denote this TPM.
4. The mean sojourn times $m_0, m_1, \ldots, m_t$ of the tangible states $M_0, M_1, \ldots, M_t$, respectively, are computed as in the case of ETPN.
5. If $\Pi = (\pi_0, \pi_1, \ldots, \pi_t)$, then the solution of the equations

$$\pi_0 + \pi_1 + \ldots + \pi_t = 1,$$

$$\Pi P = \Pi,$$

yields the stationary probability distribution of the REMC.
6. If $p_0, p_1, \ldots, p_t$ are the steady state probabilities of the tangible states of the marking process of the GSPN, then

$$p_i = \pi_i m_i / \sum_{j=0}^{t} \pi_j m_j, \quad i = 0,1,2,\ldots,t.$$

7. Various performance measures of the system modelled by the GSPN are determined using the above probabilities.

2.5 *A GSPN example*

To illustrate the various steps in the analysis of a GSPN, we present the GSPN model of a simple two-processor system where each processor acts as a standby for the other. In this system, there are two processors P_1 and P_2. When a job is waiting in the system and both the processors are up, the job is assigned to P_1. If P_1 fails while processing the job, P_2 takes up the processing of the job and simultaneously, P_1 gets repaired. When P_2 is processing, it may break down at which time P_1 may be down or in working condition. In the latter case, P_1 will process the job. A GSPN model that represents the interactions in this system is shown in figure 1. For this GSPN,

$$P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$$

$$T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8\}; \quad T_I = \{t_1, t_2\};$$

$$T_E = \{t_3, t_4, t_5, t_6, t_7, t_8\}$$

$$M_0 = (0011000)$$

$$F(M, t_{i+2}) = r_i \; \forall \; M \in R[M_0], \text{ and } i = 1,2,3,4,5,6,$$

where each r_i is a real number that represents the exponential firing rate of transition t_i.

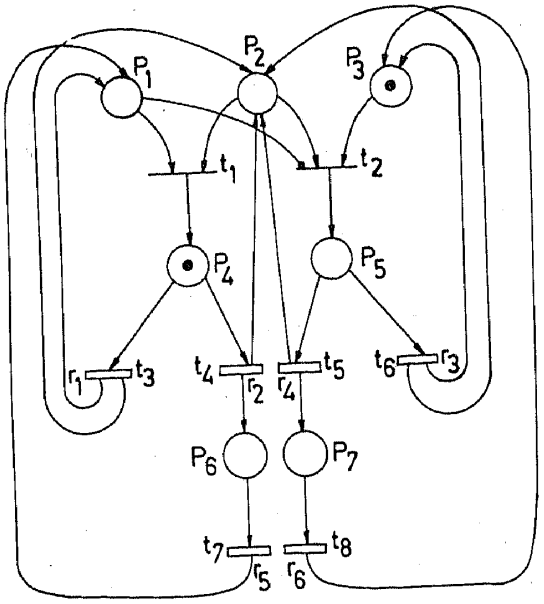

Figure 1. GSPN model of a two-processor system in which each processor acts as the standby for the other.

The function A is represented in the diagram by the directed arcs. There is a special arc between p_1 and t_2, called the inhibitor arc. The effect of this arc is that t_2 will fire only if there is a token each in p_2 and p_3 and no token in p_1. This effectively gives a way of giving priority to t_1 over t_2. It is clear that t_2 can fire only if t_1 is not enabled. The physical interpretation of the places and transitions of this GSPN model is shown in table 1.

We now explain the various steps in the analysis of the GSPN model.

Step 1. By tracing the evolution of this GSPN from the initial marking, we can generate the reachability set. There are five tangible states M_0,M_1,M_2,M_3,M_4 and three vanishing states M_5,M_6,M_7. The reachability graph is shown in figure 2. The markings are described therein. Single circles represent vanishing states and double circles represent tangible states.

Step 2. The EMC of the marking process of the GSPN is also shown in figure 2. The transition probabilities are labelled on the directed arcs. If there is no directed arc between two states, say from M_i to M_j, the meaning is the $(i,j)^{\text{th}}$ entry in the TPM is zero.

Table 1. Interpretation of the places and transitions of the GSPN model of figure 1.

p_1 : P_1 available	t_1 : P_1 starts executing the job
p_2 : Job ready	t_2 : P_2 starts executing the job
p_3 : P_2 available	t_3 : P_1 finishes executing the job
p_4 : P_1 executing the job	t_4 : P_1 fails while executing the job
p_5 : P_2 executing the job	t_5 : P_2 fails while executing the job
p_6 : P_1 undergoing repair	t_6 : P_2 finishes executing the job
p_7 : P_2 undergoing repair	t_7 : Repair of P_1
	t_8 : Repair of P_1

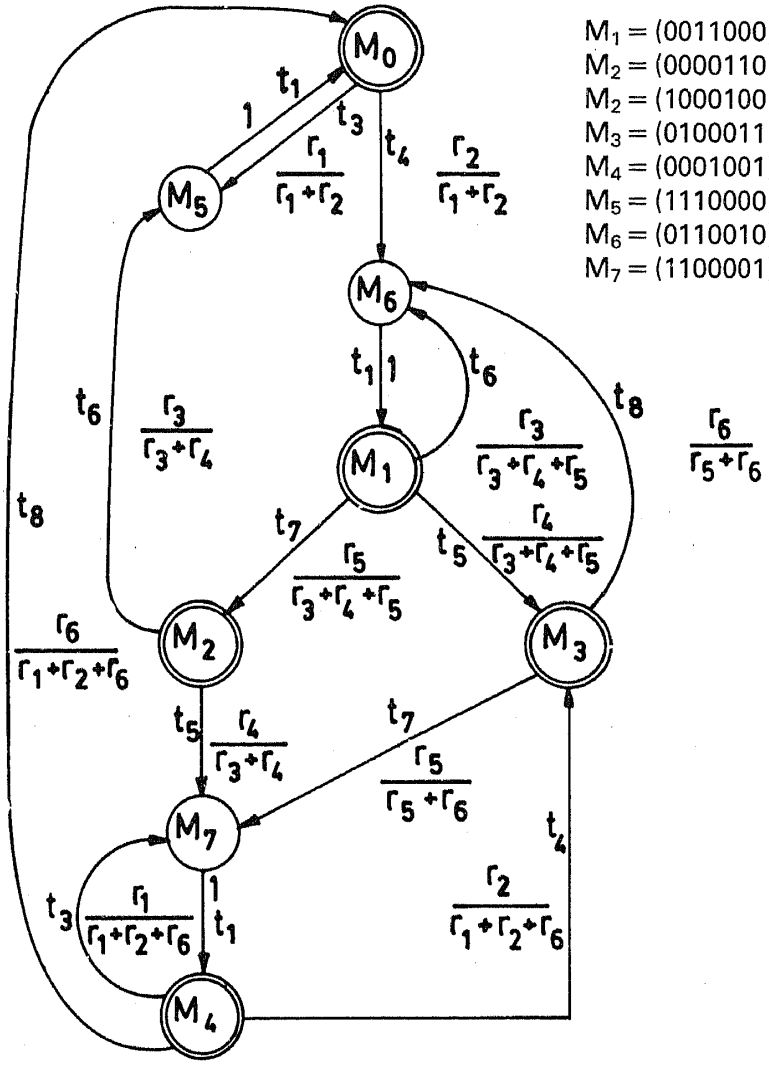

Figure 2. Embedded Markov chain and the transition probabilities for the GSPN model of the two-processor system. The states with double circles are tangible states and the rest are vanishing states.

Step 3. The REMC which comprises only the tangible states is shown in figure 3. By careful observation of figures 2 and 3, how the REMC is obtained from the EMC becomes clear. The non-zero entries of the TPM of the REMC are shown as labels on the directed arcs of figure 3.

Step 4. The mean sojourn times of the tangible states are given by

$$m_0 = 1/(r_1+r_2);\ m_1 = 1/(r_3+r_4+r_5);$$

$$m_2 = 1/(r_3+r_4);\ m_3 = 1/(r_5+r_6);$$

$$m_4 = 1/(r_1+r_2+r_6).$$

Steps 5, 6, and 7 are now easy to work out and this completes the analysis of this GSPN model.

3. Queueing network model of FTMP

In the case of FTMP, the performance measures of interest would be: real-time performance of the system, degree of fault-tolerance attained by the system, processing power, utilization of the system bus and the processors, and contention for various resources. Shin *et al* (1985) have constructed a closed queueing network (CQN) model for the FTMP and have obtained several of these performance measures by solving the CQN model. In this section, we give essential details of the FTMP and review the CQN model.

3.1 *FTMP architecture*

There are four major components in FTMP hardware: processing clusters (PC), input/output (I/O) links, system bus, and system memory. A block diagram of the FTMP hardware is shown in figure 4. There are m PC. Each PC consists of one or more pairs of a processor and a local (cache) memory. Each PC is identical in the sense of having the same number of processor/memory pairs. All pairs in a cluster work independently on a single task at any given time to improve the reliability of the system. The system bus is a time-shared bus which is redundant for the sake of reliability. Only one cluster transmits and receives data over all copies of the bus at a time. The system memory is a collection of dynamic RAM. These are redundant with the restriction that only one system memory location may be addressed at a given time. I/O links are components that enable data to be transmitted to or from external devices such as sensors, actuators, and displays.

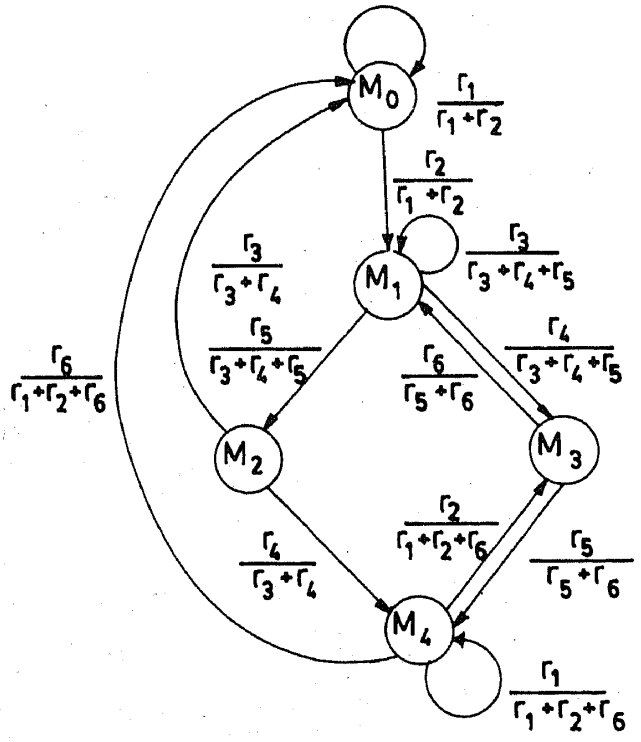

Figure 3. Reduced embedded Markov chain and the transition probabilities for the GSPN model of the two-processor system.

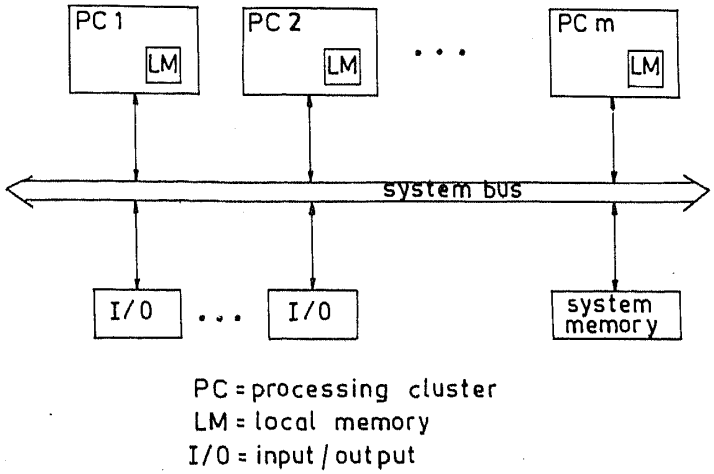

Figure 4. FMTP system architecture (figure taken from Shin *et al* 1985).

3.2 *FTMP workload*

It is the workload which decides the performance of a computer system and it is no different in the case of FTMP. Since FTMP is used in a real-time environment for air traffic control, the workload it handles comprises a fixed set of tasks that belong to several job classes. The tasks in each job class have the same frequency of initiation and are despatched at regular intervals to handle repetitive functions such as flight control, configuration control, fault detection, fault recovery, and system displays.

Different job-classes are assigned different priorities based on their frequency of initiation. A job-class gets priority over another if it has greater frequency of initiation than the latter. Consequently, a cluster working on or about to work on a task gets priority for using system resources over a cluster working on or about to work on a task that is initiated less frequently compared to the first.

3.3 *Operating rules*

All tasks to be executed by the system are stored in the system memory. An idle cluster wishing to process a task, say of class i, has to first gain control of the system bus. The idle cluster waits until the bus is free and proceeds to participate in a polling sequence, which is a decision process to determine the cluster with highest priority. For this, each cluster transmits its priority number over the system bus and a voting mechanism decides the cluster which will gain control of the system bus. If a cluster fails in a polling sequence, it waits until the bus is free again and initiates another polling sequence.

A cluster that succeeds in a polling sequence reads the task queue for a specific job class from system memory and determines the next task to be executed, based on a FCFS policy. The cluster then reads in the selected task and all data required for processing the task. This data may be obtained from I/O link reads or more system memory reads. After obtaining all information necessary for internally executing the task, the cluster updates the task queue in the system memory and releases the bus. When a cluster completes a task, it will again request bus control and transmit its results to relevant addresses. It then determines which job class to process next and then proceeds as before. At any given time, all the clusters could be processing tasks simultaneously, resulting in peak performance. There is

degradation in performance when a cluster becomes idle, waiting for control of the system bus.

3.4 *Engineering prototype of FTMP*

The performance evaluation studies carried out in Shin *et al* (1985) and in this paper are on an engineering prototype of FTMP built by the Charles Stark Draper Laboratory and installed at the NASA AIRLAB at the Langley research centre. A block diagram of the hardware architecture of the prototype is shown in figure 5. The prototype comprises ten identical line replaceable units (LRU). Each LRU contains a processor/cache memory module, a shared 16 K word memory, an I/O port, a clock generator, and related peripheral support and control circuitry. A processing cluster is a processor triad and triple modular redundancy is used. Upto three processor triads can be in operation simultaneously, utilizing nine of the ten processor/cache modules. The tenth module serves as a spare. With three triads operating simultaneously, the system functions as a 3-processor multiprocessor.

Upto three memory triads can be formed from nine memory modules, with the tenth module used as a spare. Each memory triad corresponds to a single 16 K work region of the system memory and thus we have 48 K words of contiguous shared memory when three memory triads are operating simultaneously.

Communications between the processors and the shared memory are through three serial system buses: a data transmit bus, a data receive bus, and a polling bus for resolving bus contention. The bus system has redundancy, but from the programmer's viewpoint, there is only one system bus. All information processing and transmission is conducted in triplicate so that local voters in each module can correct errors.

The software is divided into five groups–executive software, facilities software, acceptance test/diagnostic software, applications software, and support software. Depending on the frequency of initiation, these various tasks fall into three job classes with the nominal frequencies 25 Hz (job class 1), 12·5 Hz (job class 2) and 3·125 Hz (job class 3). There is a dispatch algorithm (which is part of the executive software) which initiates these tasks at the corresponding frequencies. For using system resources, tasks of job class 1 get priority over those of job class 2 and job class 3 and tasks of job class 2 get priority over those of job class 3.

3.5 *Queueing network model of FTMP*

The model proposed by Shin *et al* (1985) for FTMP is a closed queueing network (CQN). In figure 6, we show the CQN model for the FTMP prototype. This model has

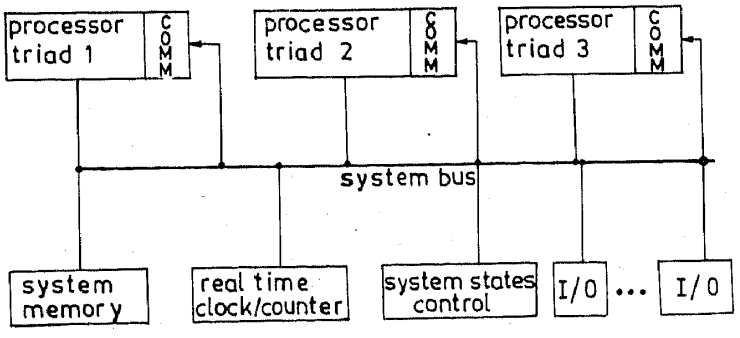

Figure 5. Architecture of the engineering prototype of FTMP (from Shin *et al* 1985).

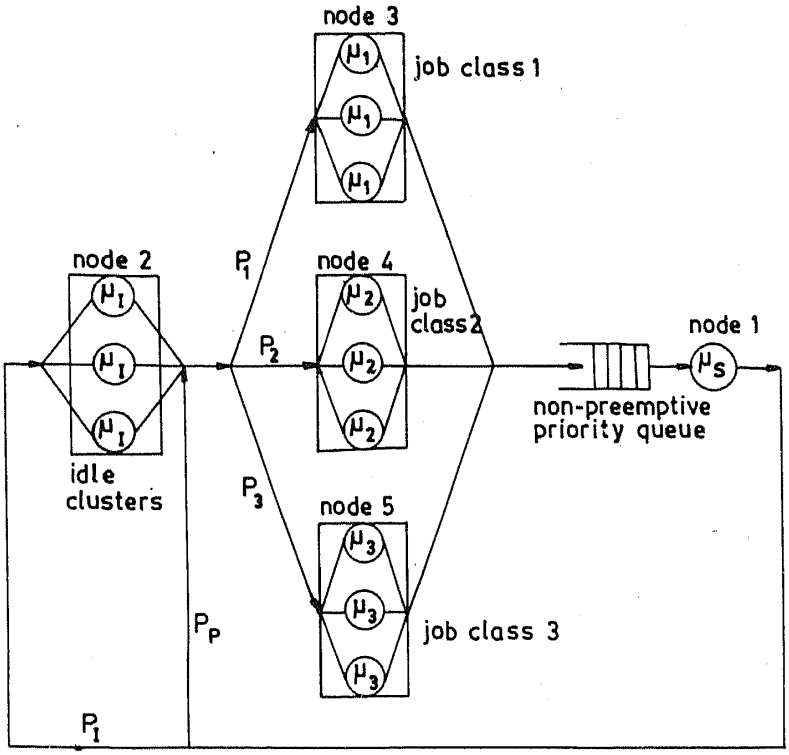

Figure 6. Closed queueing network model of FTMP, developed by Shin *et al* (1985).

five nodes and each node represents a customer that needs service. The tokens that move about the CQN model are the exponential servers moving from customer to customer. Various tasks are the customers while the processing clusters and the system bus are the servers. Since the number of servers remains the same between any two successive server failures, this queueing network becomes a CQN.

3.5a *Assumptions in the CQN model*: 1) Processing clusters and the system bus do not fail. This assumption effectively means that failures and reconfigurations in the system will result in a change in the number of tokens in the CQN.
2) Processing time of each task is an exponentially distributed random variable. The processing rate of a task of job class i ($i = 1,2,3$) is μ_i.
3) Idle time of each cluster is exponentially distributed with rate μ_I.
4) Bus transmission time of all tasks is exponentially distributed with rate μ_s.
5) Number of tasks in each job class is at least one.

3.5b *Nodes of the CQN model*: Node 1 represents transmission activity over the system bus. It consists of a non-preemptive priority queue and an exponential transmission server. A token at this node represents a cluster that is either waiting to transmit on the system bus or currently transmitting. A non-preemptive priority queue is used to enforce the priorities among clusters based on the job classes they are processing. Clusters processing tasks of the same job class transmit on a FCFS basis.

Node 2 represents idle clusters. This is a multiserver node with three servers (same as the number of clusters). A node of this type indicates that all the clusters may be served at this node without a queue forming. The sojourn time in this idle

state is exponentially distributed with rate μ_I. The rate at which clusters leave this node is $k \cdot \mu_I$ where k is the number of tokens being served by the node.

Nodes 3, 4, and 5 represent the processing of the job classes 1, 2, and 3 respectively. Each of these nodes, like node 2, is a multiserver node with three servers. The rate at which clusters leave the node $i+2$ ($i = 1,2,3$) is $k \cdot \mu_i$ where k is the number of tokens being served by the node.

3.5c *Branch probabilities*: When a cluster completes transmission on the system, it either drops into the idle state or continues processing. The probability of the former event is P_I and that of the latter is P_p, with $P_I + P_p = 1$. When a cluster is to enter a processing state, there is a probability P_i of its getting assigned to a task of class i where $i = 1,2,3$. Also, $P_1 + P_2 + P_3 = 1$. Typically, $P_i > P_j$ when $i < j$. Typical values for these branch probabilities as well as the exponential service rates are given in the NASA report of Shin *et al* (1985). These parameters were obtained through analysing experimental data of the FTMP prototype and also by making reasonable assumptions.

3.5d *Analysis of the CQN model*: The system state is a 5-tuple $(a_1, a_2, a_3, a_4, a_5)$ where $a_i \in \{0,1,2,3\}$ is the total number of tokens representing clusters at node i, $i = 1,2,3,4,5$. In the CQN model of the FTMP prototype, there are 35 system states [Shin *et al* 1985]. These 35 states are shown in table 2. These states constitute those of a continuous time Markov chain. Since it is possible for a token in this model to move from one node to any other node, the Markov chain is irreducible. Also, since there is a non-zero probability that a token leaving a node will return to that node, the Markov chain is recurrent. Thus we have a finite, irreducible, recurrent Markov chain and we can therefore compute the steady state probability distribution using classical techniques (Ross 1983).

The following performance measures have been obtained, using the steady state probabilities.

Table 2. System states of the CQN model of FTMP.

State	a_1	a_2	a_3	a_4	a_5	State	a_1	a_2	a_3	a_4	a_5
0	3	0	0	0	0	18	0	2	0	0	1
1	2	1	0	0	0	19	0	1	2	0	0
2	2	0	1	0	0	20	0	1	1	1	0
3	2	0	0	1	0	21	0	1	1	0	1
4	2	0	0	0	1	22	0	1	0	2	0
5	1	2	0	0	0	23	0	1	0	1	1
6	1	1	1	0	0	24	0	1	0	0	2
7	1	1	0	1	0	25	0	0	3	0	0
8	1	1	0	0	1	26	0	0	2	1	0
9	1	0	2	0	0	27	0	0	2	0	1
10	1	0	1	1	0	28	0	0	1	2	0
11	1	0	1	0	1	29	0	0	1	1	1
12	1	0	0	2	0	30	0	0	1	0	2
13	1	0	0	1	1	31	0	0	0	3	0
14	1	0	0	1	1	32	0	0	0	2	1
15	0	3	0	0	0	33	0	0	0	1	2
16	0	2	1	0	0	34	0	0	0	0	3
17	0	2	0	1	0						

1. Probability that a cluster is idle.
2. Probability that there is bus contention.
3. Average queueing time of a customer of class i ($i = 1,2,3$).
4. Mean queueing time for a typical customer.

Also, variations of these measures due to change in system parameters such as the exponential service rates and branch probabilities have also been investigated.

4. Stochastic Petri net models of FTMP

The CQN model of FTMP, reviewed in the previous section is not adequate for the following reasons.

1. The bus transmission time for all tasks is assumed to be identically distributed. A more realistic assumption would be to consider different exponential distributions for the bus transmission times of different job classes.
2. The CQN model does not capture the exact sequence of operations in the FTMP since idle clusters are shown in the model to start processing a task straightaway. In the actual system, an idle cluster first obtains all the programs and data corresponding to the next task from the system memory and I/O links and then only starts processing the task.
3. The CQN model is valid only when there are no processor and bus failures. In the event of a failure of a processing cluster or bus, the number of tokens in the CQN model comes down by one and we have to solve this new CQN separately. Further, reconfiguration activity is not represented in the model.
4. After finishing the processing of a task, a cluster goes through an exponentially distributed idle period. In the actual system, however, the idle time of a cluster is decided by various factors such as the rate at which jobs are initiated, the number of resources in the system, etc.

In this section, we show how compact performance models can be built for FTMP using GSPN. In the report of Shin *et al* (1985), a GSPN model has been presented for FTMP, which is very huge, cumbersome, and intractable. The GSPN models that we present in this paper are however easy to understand and solve. We first develop a GSPN model which is an exact replica of the CQN model.

4.1 *GSPN model 1*

Figure 7 shows a GSPN model of FTMP, whose layout follows closely that of the CQN model of figure 6. The interpretation of the places and transitions of this model, the firing rates of the exponentially timed transitions, and the specifications of the random switches in this model are all given in table 3. Note that several of the firing rates are marking dependent, as in case of the CQN model. The random switches aid us in making decisions at various points and use the branch probabilities of the CQN model. The place p_1 of the model represents node 2 of the CQN model; places p_3,p_4,p_5 represent nodes 3,4,5 respectively; and, places p_7 and p_8 represent node 1. The initial marking of this model is (300001000) which corresponds to the state when all three clusters are idle and the system bus is free.

The reachability set consists of 35 tangible states and 40 vanishing states. The former states correspond precisely to the 35 system states of the CQN model. Also, the performance results obtained from these two models are identical.

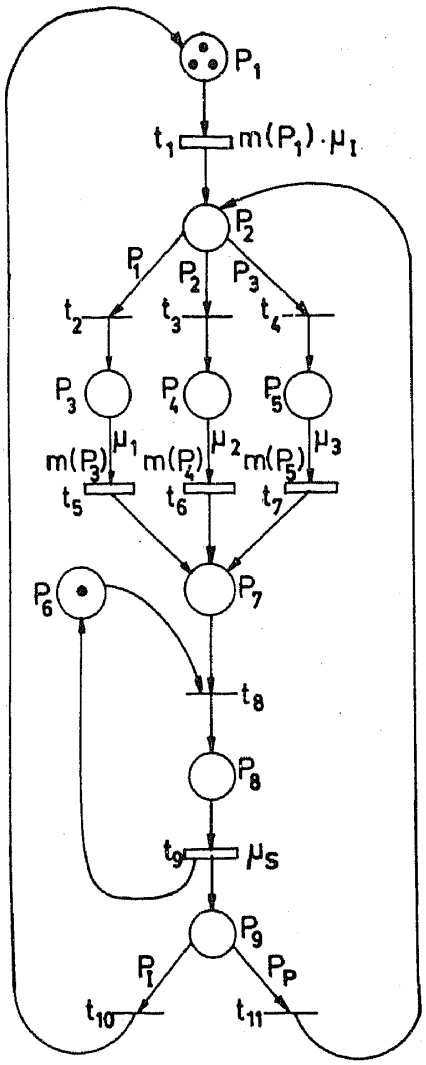

Figure 7. A GSPN model of FTMP, modelling exactly the same features as the closed queueing network model of Shin *et al* (1985).

Table 3. Description of GSPN model 1.

p_1 : Idle clusters	t_1 : Idling phase of clusters; rate = $m(p_1)\cdot\mu_I$
p_2 : Clusters ready to process tasks	t_2 : Cluster chooses to process a task of class 1
p_3 : Clusters processing tasks of class 1	t_3 : Cluster chooses to process a task of class 2
p_4 : Clusters processing tasks of class 2	t_4 : Cluster chooses to process a task of class 3
p_5 : Clusters processing tasks of class 3	t_5 : Processing of tasks of class 1; rate = $m(p_3)\cdot\mu_1$
p_6 : Bus idle	t_6 : Processing of tasks of class 2; rate = $m(p_4)\cdot\mu_2$
p_7 : Clusters waiting to transmit on the bus	t_7 : Processing of tasks of class 3; rate = $m(p_5)\cdot\mu_3$
p_8 : Bus transmission in progress	t_8 : Cluster starts bus transmission
p_9 : Cluster that has just finished a bus transmission	t_9 : Bus transmission activity; rate = μ_S
	t_{10} : Cluster becomes idle after bus transmission
	t_{11} : Cluster resumes processing after bus transmission

Random switches

1) Transitions t_2, t_3, t_4 with probabilities P_1, P_2, and P_3 respectively.
2) Transitions t_{10} and t_{11} with corresponding probabilities P_I and P_p.

4.2 *GSPN model 2*

In the GSPN model 1, we did not explicitly represent the priorities assigned to the clusters based on the job classes they are working on nor did we model the polling sequence. In GSPN model 2, which is pictured in figure 8, we capture the above features. The place p_7 of GSPN model 1 is now replaced by three places p_7,p_8,p_9, to model explicitly clusters working on three different job classes. Also we have used inhibitor arcs to give priority to transition t_8 over t_9 and t_{10} and to transition t_9 over t_{10}. The places p_1–p_6 and the transitions t_1–t_7 in the two GSPN models have the same significance. The interpretation of the remaining places and transitions is given in table 4. The reachability set of GSPN model 2 comprises 48 tangible states and 75 vanishing states. Owing to space constraints, we are not giving a list of these states. We can however find a mapping (that is not one-to-one) which associates these 48 states to the 35 states obtained in the case of the previous model. By analysing this GSPN model and computing its steady state distribution, we can directly obtain performance measures such as mean bus waiting times of clusters processing different job classes. This is in addition to the performance measures that can be computed using the earlier models.

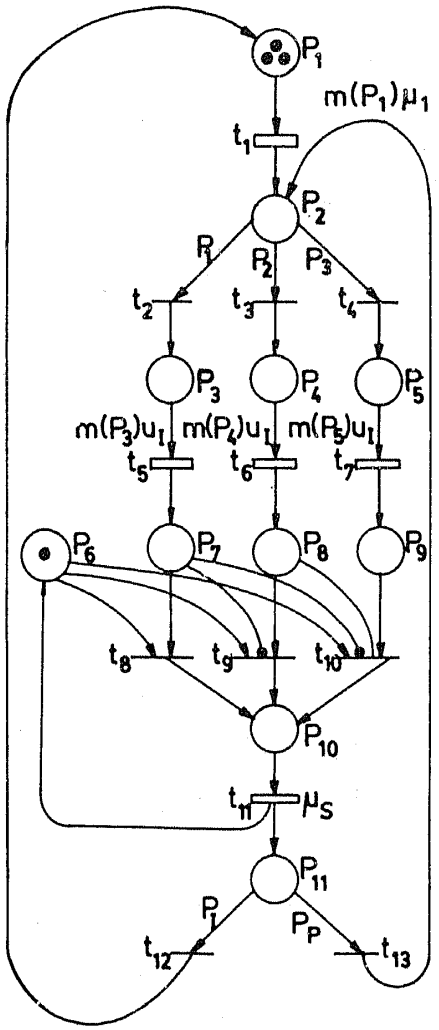

Figure 8. A GSPN model of FTMP, showing explicitly the priorities enforced in the polling mechanism.

Table 4. Description of GSPN model 2.
The places p_1–p_6 and transitions t_1–t_7 have the same description as in table 3. The phrase 'cluster of type i' means a cluster working on or about to work on a task of class i (i = 1,2,3).

p_7 : Clusters of type 1 waiting to access the bus	t_8 : Cluster of type 1 starts using the bus
p_8 : Clusters of type 2 waiting to access the bus	t_9 : Cluster of type 2 starts using the bus
p_9 : Clusters of type 3 waiting to access the bus	t_{10} : Cluster of type 3 starts using the bus
p_{10} : Bus transmission in progress	t_{11} : Bus transmission activity; rate = μ_s
p_{11} : Cluster, just after finishing a bus transmission	t_{12} : Cluster becomes idle after finishing a bus transmission
	t_{13} : Cluster resumes processing after finishing a bus transmission.

Random switches
1) Transitions t_2, t_3 and t_4 with corresponding probabilities P_1,P_2 and P_3.
2) Transitions t_{12} and t_{13} with probabilities P_f and P_p respectively.

4.3 *GSPN model 3*

This GSPN model overcomes two shortcomings of the previous models. The first one is in representing the exact sequence of operations a processing cluster undergoes. In the previous models, a cluster, after the idling phase, started processing a selected task straightaway, instead of first acquiring all the programs and data required for executing the task. Secondly, the previous models did not consider the individual characteristics of tasks of different job classes with the result that their identity is not maintained throughout the model. The present model overcomes these two difficulties by including a small number of additional places and transitions. The model is shown in figure 9, and the description of the elements of the model is given in table 5. It may be noted that the processing of each job class is now separately modelled and there is no loss of identity of job classes anywhere. There are now 15 places and 19 transitions and the reachability set comprises 104 tangible states and 145 vanishing states. Using this model, we can get more detailed and accurate information about the FTMP system. For instance, in addition to all the previously mentioned performance measures, we can now compute the fraction of time the bus is used by each job class and the total processing time for each job class.

4.4 *GSPN model 4*

We now show how failures and the subsequent reconfiguration/repair activities in FTMP can be modelled using GSPN. In the FTMP, there can be failure of processors, memories, or the system bus. We shall only consider processor failures. When a processor fails, its triad will attempt to complete its current job step, which it will be able to do unless a second failure prevents it. When the job step is complete, one of the other processor triads is assigned the task of reconfiguring the injured triad. If a spare processor is available, the injured triad is connected to the appropriate bus and if no spares are available, it is retired. In the latter case, the resources of the multiprocessor are diminished by one processing unit and the two unfailed members of the retired triad are now available as spares. More details about reconfiguration and repair can be found in Hopkins *et al* (1978).

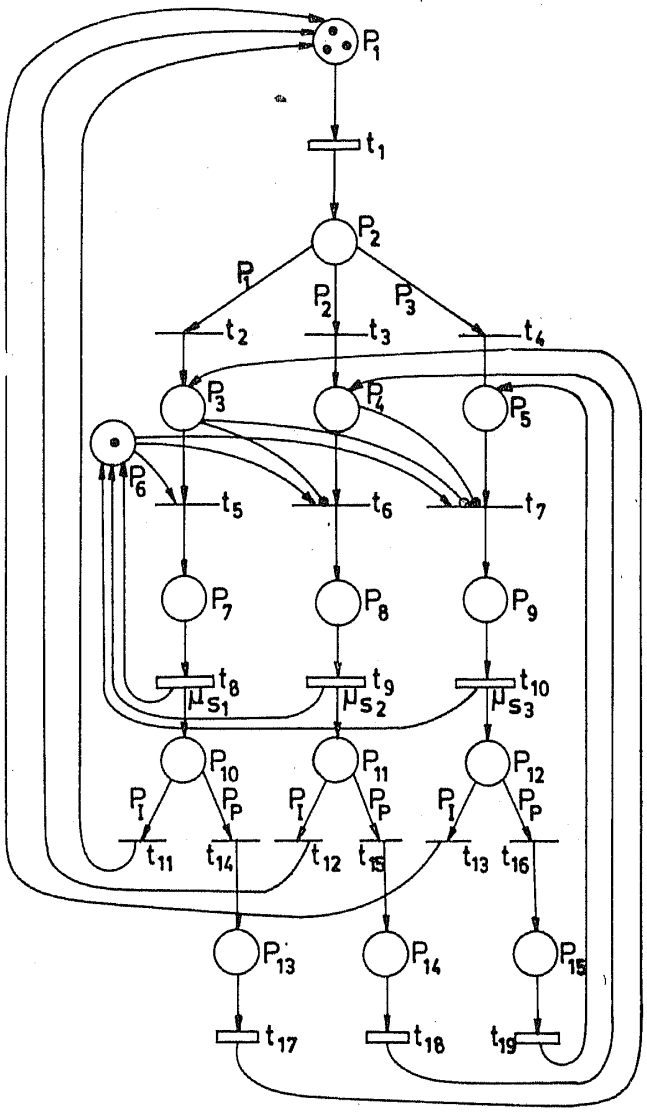

Figure 9. A GSPN model of FTMP, taking care of priorities and individual characteristics of job classes.

Figure 10 shows a GSPN that models reconfiguration and repair in the event of processor failures. The overall GSPN model of FTMP can be easily constructed from this GSPN. Table 6 gives the interpretation of the places and transitions of this GSPN. It is to be noted that multiple failures are not taken care of by this model. By solving the overall GSPN model, we can now obtain the performance measures in the presence of failures.

In respect of modelling failures, GSPN models have one definite advantage over the CQN model. The closed nature of the CQN model depends crucially on the fact that there are no failures in the system. To obtain the performance of the system in the presence of failures using CQN models, one has to obtain the performance contribution from each of the configurations and weight it by the relative time of operation. The GSPN model on the other hand, gives a direct and more accurate method of computing system performance in the presence of failures.

Table 5. Description of GSPN model 3.
The phrase 'cluster of type i' refers to a cluster that is working on or about to work on a task of class i. In this table, the index i takes the values 1,2,3 wherever mentioned.

p_1	:	Idle clusters	t_1	:	Idle phase of clusters; rate = $m(p_1)\cdot\mu_I$
p_1	:	Clusters ready to process	t_{i+1}	:	A cluster chooses to process a task of class i
p_{i+2}	:	Clusters of type i waiting to access the bus	t_{i+4}	:	A cluster of type i starts using system bus
p_6	:	Bus available	t_{i+7}	:	Bus transmission activity of cluster of type i; rate = μ_{S_i}
p_{i+6}	:	Clusters of type i using the system bus	t_{i+10}	:	A cluster of type i becomes idle after using the system bus
p_{i+9}	:	A cluster of type i that has just finished using the bus	t_{i+13}	:	A cluster of type i resumes processing its task after using the system bus
p_{i+12}	:	Clusters processing tasks of job class i	t_{i+16}	:	Processing activity by clusters of type i; rate = $m(p_{i+12})\cdot\mu_i$

Random switches
t_2, t_3, t_4 with probabilities P_1, P_2, P_3, respectively
t_{11} and t_{14} with probabilities P_I and P_p, respectively
t_{12} and t_{15} with probabilities P_I and P_p, respectively
t_{13} and t_{16} with probabilities P_I and P_p, respectively

4.5 *Other refinements*

Here, we outline how we can construct more realistic models for FTMP. We consider not only GSPN but also other recent variants of SPN proposed in the literature.

4.5a *FTMP dispatching*: In the FTMP, there is a dispatching program, part of the executive software, which schedules various tasks at regular intervals to handle repetitive applications. In the models discussed so far, processing clusters are assumed to have exponentially distributed idle times. This is not a reasonable assumption because the idle time of a cluster is decided by the arrival pattern of the incoming jobs as scheduled by the dispatching algorithm and by random

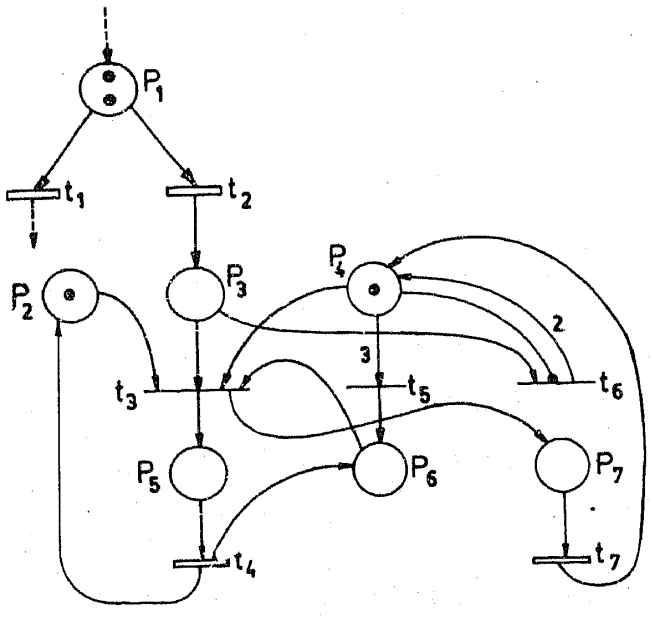

Figure 10. A GSPN representation of reconfiguration and repair in FTMP in the event of a processor failure. Note that the arcs from p_4 to t_5 and from t_6 to p_4 have multiple arcs with weights 3 and 2, respectively.

Table 6. Interpretation of the places and transitions of the GSPN model of figure 10.

p_1 : Clusters processing tasks of a given job class	t_1 : A cluster finishes processing a task
p_2 : Bus available	t_2 : A processor fails during execution of a task
p_3 : Failed clusters	t_3 : Reconfiguration starts
p_4 : Spare processors available	t_4 : Reconfiguration activity
p_5 : Reconfiguration in progress	t_5 : A processor triad is formed from three spare processors
p_6 : Idle clusters	t_6 : Spares not available and hence the two unfailed processors become spares
p_7 : Repair of a processor in progress	t_7 : Repair activity of a processor

phenomena such as process/cluster/memory/bus failures. Therefore by representing the actions of the dispatcher explicitly in the model, we obtain a more realistic model.

4.5b *Non-exponential distributions*: In a GSPN, all the timed transitions have exponential distributions associated with them. It is often more realistic and more accurate to assume non-exponential distributions. However the analysis of such an SPN becomes complex since the memoryless property of the exponential distribution cannot be used any more. In the literature, Molloy (1981), Dugan *et al* (1984), Marsan *et al* (1985), Meyer *et al* (1985), and Haas & Shedler (1986) represent the major efforts in incorporating general distributions in SPN. The FTMP performance models using non-exponential distributions can be analysed through the results of these researches.

4.5c *Deterministic and stochastic firing times*: Some work has been carried out recently in the analysis of stochastic Petri nets with three types of transitions: those which fire in zero time, those which fire in deterministic time and those which fire in an exponentially distributed time. Marsan & Chiola (1986) have presented a technique by which such an SPN can be analysed, when in each reachable marking of the SPN, at most one concurrent deterministic transition is enabled.

In a real-time environment such as the one in which FTMP is used, normally there is a fixed set of jobs executed at regular intervals. Hence the program size and the I/O data will be almost the same for a given task. It is therefore more realistic to assume deterministic durations for processing times and bus transmission times. The technique devised by Marsan & Chiola (1986) can be used for analysing the GSPN models 1, 2, and 3 by assuming deterministic bus transmission times and keeping the rest of the models unchanged. However their technique cannot be used if we assume deterministic processing times. This suggests an interesting future direction for research namely the theoretical investigation of solutions of FTMP models when deterministic and stochastic firing times coexist.

4.5d *Integrated models using queueing networks and GSPN*: In this paper, we have looked at both CQN and GSPN models of FTMP. A class of queueing networks called product from queueing networks (PFQN) is efficiently solvable but entails several restrictive assumptions to be made on the modelled system. These restrictive assumptions such as absence of blocking, absence of synchronization, and absence of priorities can be overcome by GSPN models. An interesting technique has been developed by Balbo *et al* (1986), which combines the best features of PFQN and

GSPN and obtains integrated models for the given system. Such an integrated model for FTMP is worth investigation and will lead to a model more efficient than that based on PFQN alone or GSPN alone.

5. Conclusions

Stochastic Petri nets represent a recent modelling technique for performance evaluation of computer systems. In this paper, we have presented an overview of SPN and developed elegant GSPN models for FTMP, a bus-based, real-time, fault-tolerant multiprocessor used as the central computer in air-traffic control applications. We have also brought out several advantages of the GSPN models over a queueing network based model proposed earlier by Shin *et al* (1985). The GSPN models presented have been used for computing vital performance measures of FTMP, using an analysis package developed by us.

References

Balbo G, Bruell S C, Ghanta S 1986 *IEEE Trans. Software Eng.* SE-12: 561–576

Dugan J B, Trivedi K S, Geist R M, Nicola V F 1984 in *Performance'84* (ed.) E Gelenbe (Elsewier: North-Holland)

Haas P J, Shedler G S 1986 *Performance Evaluation* 6: 189–204

Holliday M A, Vernon M K 1985 *Proc. Int. Workshop on timed Petri nets, Torino, Italy* (Silver Spring, MD: IEEE Comput. Soc. Press)

Hopkins A L Jr, Smith T B, Lala J H 1978 *Proc. IEEE* 66: 1221–1239

IEEE 1985 *Proc. of the Int. Workshop on timed Petri nets, Torino, Italy* (Silver Spring, MD: IEEE Comput. Soc. Press)

Marsan M A, Balbo G, Bobbio A, Chiola G, Conte G, Cumani A 1985 *Proc. Int. Workshop on timed Petri nets, Torino, Italy* (Silver Spring, MD: IEEE Comput. Soc. Press) pp. 80–87

Marsan M A, Balbo G, Conte G 1984 *ACM Trans. Comput. Syst.* 2: 93–122

Marsan M A, Balbo G, Conte G 1986 Performance models of multiprocessor systems (Boston: The MIT Press)

Marsan M A, Chiola G 1986 *Proc. 7th European workshop on applications and theory of Petri nets, Oxford, England* pp. 151–165

Meyer J F, Movaghar A, Sanders W H 1985 *Proc. Int. Workshop on timed Petri nets, Torino, Italy* (Silver Spring, MD: IEEE Comput. Soc. Press) pp. 106–115

Molloy M K 1981 *On the integration of delay and throughput measures in distributed processing models*, Ph D thesis, University of California, Los Angeles

Natkin S 1980 *Reseaux de Petri Stochastiques*, These de Docteur-Ingegneur, CNAM-Paris

Petri C A 1966 *Communication with automata*, Ph D thesis; Technical Report RADC-TR-65-377, New York, January 1966

Ramamoorthy C V, Ho G S 1980 *IEEE Trans. Software Eng.* SE-6: 440–449

Ramchandani C 1973 *Analysis of asynchronous concurrent systems by timed Petri nets*, Ph D thesis, Massachusetts Institute of Technology

Ross S M 1983 *Stochastic processes* (New York: John Wiley and Sons)

Shin K G, Woodbury M H, Lee Y H 1985 Modelling and measurement of fault-tolerant multiprocessors, NASA Contractor Report CR-3920, August

Sifakis J 1977 *Third Int. Workshop on modelling and performance evaluation of computer systems, Amsterdam* (Amsterdam: North-Holland)

Zuberek W M 1985 *Int. Workshop on timed Petri nets, Torino, Italy* (Silver Spring, MD: IEEE Comput. Soc. Press)