Exact Performance Estimates for Multiprocessor Memory and Bus Interference

by

Mark A. Holliday
Mary K. Vernon

Computer Sciences Technical Report #594

May 1985

Exact Performance Estimates for Multiprocessor Memory and Bus Interference

Mark A. Holliday and Mary K. Vernon

Computer Sciences Department
University of Wisconsin — Madison
Madison, WI 53706

Abstract

Exact results are given for the processing power in a multibus multiprocessor with constant memory cycle times and geometric interrequest times. Both uniform and non-uniform memory accesses are considered. Such results have not previously been obtained. In order to derive these results we use a method of introducing time into Petri Nets, called Generalized Timed Petri Nets (GTPN), that we have developed. We describe the GTPN and how it is applied to the multiprocessor interference question. We reach several new conclusions. A commonly used definition of processing power can lead to substantial underestimation of the true processing power of the system. If the real system has a constant memory access time and any number of buses, then assuming an exponential access time can lead to substantial errors when estimating processing power probability distributions. In multibus systems with only a few buses a critical memory interrequest time exists. Performance close to that with a crossbar is attainable when the interrequest time is larger than the critical value. Obtaining these results illustrates the advantages, for moderate size state spaces, of the GTPN over simulation with respect to both model design and running time.

This research was partially supported by the National Science Foundation under grants DCR-8402680 and MCS81-05904.

1. Introduction.

Due to their potential for increased performance through parallelism, multiprocessor systems (systems with more than one processor and with shared memory) have been studied for many years. In this paper we are interested in MIMD multiprocessors with multiple independent memory modules and with a single-stage multibus interconnection network. A key issue in such a system is the extent to which contention for the shared memory modules and the buses causes performance degradation. An extensive literature has developed addressing this issue with stochastic models[1][2][3]. Until now, however, no one has used exact solution methods to derive performance measures for any model that contains four important and realistic properties. These properties are:

1) both bus and memory contention are considered

2) the amount of time spent actively accessing memory per request is a constant

3) variable and non-zero processing time between memory requests

4) a not necessarily uniform distribution of accesses across the memory modules

We have developed a model which can be used to derive performance estimates for a system containing these properties. The solution method is based on the global state transition diagram (i.e. discrete time Markov Chains). Previous researchers[3][4] have concluded that this approach is computationally infeasible except for very small systems. We demonstrate that this approach is feasible for useful size systems if the Markov Chain is properly formulated. In particular, we derive results for several models (with up to 16 processors and 16 memories) with the four properties listed above.

To aid in the definition and generation of the Markov Chain we used a high level description based on Petri Nets. Petri Nets are a graph model of computation [5]. Modifying Petri Nets so that time is represented has recently been an active research area. We have generalized the method of representing time that has been suggested by Zuberek [6] and Razouk and Phelps [7]. We call our version of Petri Nets, Generalized Timed Petri Nets (GTPNs) [8,9]. The GTPN analyzer automatically generates the associated discrete-time Markov Chain and calculates the requested performance estimates.

We do not mean to underestimate the importance of approximate solution techniques and simulation. As more functionality is encoded in a model, the Markov Chain generated using the GTPN analyzer can still become prohibitively large. Our point is that the GTPN allows symmetries to be discovered and used to reduce the model complexity. Thus, with careful model formulation, the Markov Chain does not become prohibitively large as soon as with earlier models. Because of the smaller state spaces, there is more opportunity for exact results.

By using the GTPN model we reached several new conclusions about the effect of memory and bus interference in multiprocessors. First, a widely used definition of the *processing power* of a multiprocessor does not accurately reflect the true increase in power of a multiprocessor over a single processor. Second, if the real system has a constant memory access time and any number of buses, then to assume that it has an exponentially distributed access time can lead to large errors in the estimation of the probability distributions of processing power. We introduce a new definition of processing power that does accurately reflect the increase in power. Third, with respect to speedup, we have identified the phenomenon of critical memory interrequest times in multibus systems. As long as the mean interrequest time is longer than the critical value, only a very few buses are needed to attain nearly the same performance as a crossbar.

In Section 2 we describe the behavior of the multiprocessors we study and review the relevant previous work in stochastic modeling of these systems. In Section 3 we briefly describe the GTPN model. The GTPN is defined in detail and compared with Stochastic Petri Net (SPN) models in [8,9]. In Section 4 we present our multiprocessor model and results. Section 5 summarizes the important contributions of our work and suggests some directions for future research.

2. Background

2.1. Multiprocessor Characteristics

Figure 2.1 illustrates the multiprocessor systems we consider. The shared memory is divided into independent modules, each of which permits only one access at a time. The processors are connected to the memory modules through a single-stage multibus (in contrast to multiple-stage networks such as banyan networks).

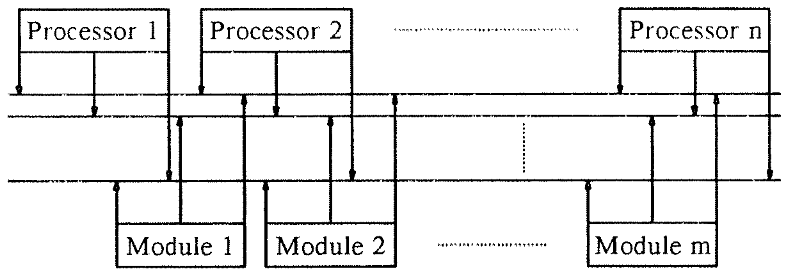

Figure 2.1. The Multiprocessor System

The process associated with each processor can be in three states: running on its processor, waiting for a memory module, or accessing a memory module. The processing time between memory requests is the processor's *interrequest time.* In much of the literature we reference below, the interrequest time is assumed to be a geometric random variable. In this case, the parameter of that random variable is the *memory request probability*(MRP). The MRP is the probability that an actively executing processor will generate a request in the next memory cycle. A process with an outstanding request blocks until it obtains an arbitrary bus and the desired memory module. The amount of time spent actively accessing memory per request is the *memory access time.* When the memory access time is a constant, time is said to be divided into *cycles* and the memory access time is sometimes called the *memory cycle time.* The distribution of accesses across the memory modules by a processor is that processor's *memory access probabilities.*

Memory utilization is the fraction of time that a memory module is being accessed by some process. *Processor utilization* is the fraction of time that a processor has its associated process running on it (versus accessing a memory or waiting for a memory). *Processor productivity* is the probability that a typical process is doing productive work (executing on its processor or accessing a memory, not waiting for a memory). Memory utilization summed over all memories is the expected number of busy memory modules or the *effective memory bandwidth.* Processor utilization summed over all processors is sometimes called *processing power*[10,11,12,13]. Effective memory bandwidth and processing power (as defined above) are the main performance estimates obtained in the studies cited below.

2.2. Previous Results: Bhandarkar

Bhandarkar[1] is an important early work. He uses discrete time Markov Chains to obtain performance estimates for up to a 16 processor/16 memory system. The key assumptions of his model are:

1) all of the processors are statistically identical
2) the memory access time (memory cycle time)is a constant
3) MRP is 1, i.e. a process is never actively executing on its own processor. When its current request is satisfied, it always generates another request at the start of the next cycle.
4) the interconnection network is a crossbar
5) requests made by each processor have an independent and equal probability of being directed to any one of the modules, i.e. *uniform memory access probabilities.*

In spite of these simplifying assumptions, the state space of his Markov Chain grows rapidly. For example, the 16 processors/16 memories system has 300,540,195 states. Bhandarkar did not attempt to solve the steady state equations for a system with 300,540,195 states. He solves the steady state equations for a much smaller state space that takes advantage of the fact that the processors are identical. However, the only way he was able to compute the transition probabilities in the smaller state space was by building and collapsing the larger state space.

Bhandarkar considered loosening assumption 3 so that the MRP is less than one, i.e. a processor might spend some time executing between requests. This would have implied a larger state space so he did not attempt an exact solution.

Most of the research in less restrictive stochastic models of multiprocessors since Bhandarkar's work has focused on approximate solution techniques. The studies are divided into categories according to which of the above assumptions (2, 3, 4, or 5) are relaxed.

2.3. Less Restrictive Models

Bhandarkar[1], Strecker[14], and Wulf and Bell[15] deal with the basic case of a constant memory access time, uniform access probabilities, a crossbar, and a MRP of 1. One logical change is to consider to what extent performance is degraded due to having fewer buses than in a crossbar. Towsley[3] gives approximate solutions and simulation values for this case. Alternatively, a crossbar could be assumed and a MRP less than one considered. This is a reasonable change, because presumably each processor has some local memory or a cache that it is using for most of its memory activity. Baskett and Smith[16], Rau[17], Yen, Patel, and Davidson[4], and Towsley[3] give approximate solutions for this case.

More recent studies combine these two changes, i.e. they have constant access time, uniform access, a MRP less than one, and multibuses. Lang, Valero, and Alegre[18] provide simulation results. Bhuyan[19], and Mudge, Hayes, Buzzard, and Winsor[20], Towsley[3], and Goyal and Agerwala[2] give approximate solutions.

Some studies assume an exponentially distributed memory access time, an exponentially distributed interrequest time, and use continuous time Markov Chains in the solution. Approximate solutions of these models are in Bhandarkar and Fuller[21], Marsan and Gerla[11], Marsan, Balbo, and Conte[12], Marsan, Balbo, Conte, and Gregoretti[13], Önyüksel and Irani[22], and Jacobson and Lazowska[23]. Exact solutions are in Irani and Önyüksel[24], Molloy[25], and Marsan, Balbo, and Conte[10]. Several of these studies [25,13,10] are of special interest because they use a form of Petri Nets, called Stochastic Petri Nets, to derive their continuous time Markov Chains. We defer further discussion until Section 4. Mudge and Al-Sadoun [26] is an approximate solution that allows the memory access time to be any discrete time random variable that has first and second moments.

The last group of studies consider nonuniform access probabilities. All assume constant cycle time, and a crossbar. Those that only allow a MRP of one are Sethi and Deo [27] and Du and Baer[28]. The papers that allow memory request probabilities less than one are: Hoogendoorn[29], Mudge and Makrucki[30], Siomalas and Bowen[31], Towsley[3], and Bhuyan[32]. All the solutions are approximate except one of the ones given in Du and Baer. The exact solution method in Du and Baer is a modification of Bhandarkar's exact method. Perhaps this is why they only consider a crossbar and a MRP of 1.

In Section 4 we develop a GTPN model of multiprocessors which can be modified easily (primarily by changing a few parameters) to reflect the various assumptions made in the above studies. We will compare the performance estimates obtained from exact analysis of the GTPN with some of the results cited in this section. We will also use the GTPN model to obtain results not previously reported. First we introduce the GTPN.

3. The GTPN Model

This section describes the Generalized Timed Petri Net (GTPN). The GTPN belongs to the class of deterministic t-Timed Petri Nets [33,34], and removes restrictions on the net in earlier methods for analyzing performance [6,7]. The GTPN model was introduced in [8] and described more completely in [9]. The reader is referred to both references for further details.

3.1. The Net

A GTPN is a Petri net which includes: 1) a deterministic firing duration associated with each transition, 2) a mechanism for specifying next state probabilities for conflicting transitions, and 3) a set of named resources associated with each transition which are used to calculate performance estimates. Thus, the GTPN is formally defined by the following tuple: Letting S denote the set of reachable states, $\Re^+$ denote the positive reals, and $\mathcal{P}$ denote the power set, the model is formally defined as follows:

$$GTPN = (P, T, A, M_0, D, F, C, R)$$

where

$P = \{p_1, p_2, \ldots, p_n\}$	(places)
$T = \{t_1, t_2, \ldots, t_m\}$	(transitions)
$A: \{P \times T\} \cup \{T \times P\} \rightarrow \{0, 1, 2, ...\}$	(directed arcs)
$M_0: P \rightarrow \{0, 1, \ldots\}$	(initial marking)
$D: T \times S \rightarrow \Re^+ \cup \{0\}$	(firing durations)
$F: T \times S \rightarrow \Re^+ \cup \{0\}$	(firing frequencies)
$C: T \rightarrow \{yes, no\}$	(CntComb boolean flags)
$R: P \cup T \rightarrow \mathcal{P}(\{r_1, r_2, \ldots, r_k\})$	(resources)

The first four components of the tuple are identical to the constructs in an untimed Petri Net (see [35] for more details). Important properties of the remaining four components are summarized below.

Petri Nets are often illustrated graphically. Figure 3.1 shows an example GTPN including the initial state distribution of tokens. Each transition is labeled with, from left to right, its firing duration expression, its frequency expression, its CntComb boolean flag, and its list of resources.

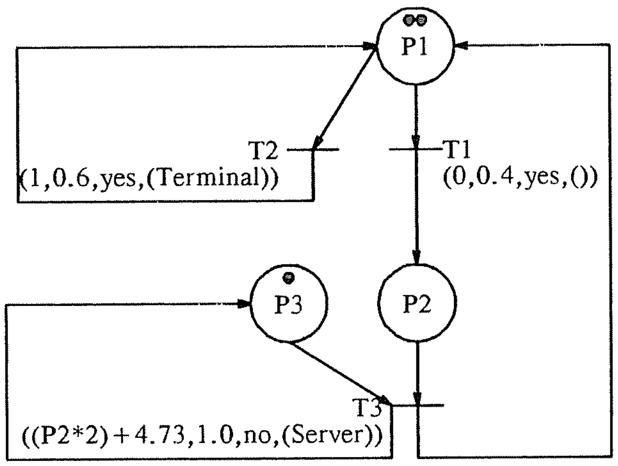

Figure 3.1. Example of a GTPN net

A transition's firing duration and frequency are expressions containing immediate values (real and integer), names of places (which represents the number of tokens in that place in the current state), names of transitions (which represents the value one if at least one firing of that transition is in progress in the current state and is otherwise zero), and arithmetic, relational, and logical operators. Thus, a transition's firing duration and frequency can be state-dependent, but for a given state they are deterministic. The CntComb boolean flag is used in computing state probabilities as explained later. The set of resources are used to compute performance measures as explained later.

3.2. Next States and Their Probabilities

The *multiplicity* of an input place is the number of arcs from that place to that transition. An output place's multiplicity is defined analogously. N *enablings* of a transition exist if each of its input places contains a number of tokens equal to at least N times its multiplicity.

An enabling of a transition can *start firing* by removing from each input place a number of tokens equal to its multiplicity. After start firing, the firing is *in progress* until *end firing.* While the firing is in progress, the time to end firing, called the *remaining firing time* (RFT), decreases from the transition's firing duration to zero. At end firing a number of tokens equal to its multiplicity is put on each output place.

A marking and the set of current RFT's defines a *state* of the net. Given a particular state, the basic rule for finding the possible next states is straightforward. Find how many enablings of each transition exist. Find the maximal sets [1] of the enablings that can start firing simultaneously. Each maximal defines a next state. The time spent in the original state is zero. The new RFT's are set to their transitions' durations. If there are no enablings, but there are some firings in progress, then the next state is generated by the end firing of all firings in progress with the smallest RFT (*Tmin*). The time-in-state value in this case is *Tmin.* If there are no enablings and no firings in progress, then the net remains in the current state forever.

The rules for assigning probabilities to next states are now given. Multiple next states can only occur when the next states are due to start firings. In this case, we need to assign a probability to each maximal set of enablings that can start firing together. Two transitions whose sets of input

[1] a set with property α is a *maximal* set with property α if it is not a proper subset of any other set with property α.

places intersect are in the same *conflict set.* The transitive closure of the property of intersecting input places defines an equivalence relation on the set of transitions. This equivalence relation partitions the set of transitions into disjoint sets called *generalized conflict sets.*

A maximal is the union of a set of independent *local maximals*, one from each generalized conflict set. The probability for a maximal is the product of the probabilities for the associated local maximals (since the local maximals are independent). To compute the probability of a local maximal, take the product of the frequencies of all the enablings (the result of evaluating the frequency expression of the enabled transition) in the maximal. Multiply this product by the number *NumComb* when appropriate as discussed below. Finally, for each maximal, normalize the product by dividing it by the sum of the products over all the maximals.

NumComb means *number of combinations* and is the number of ways tokens can be removed from input places in order to implement a local maximal. In some cases, multiplying by this value is needed in order to derive an intuitively reasonable probability. The boolean flag CntComb (Count Combinations) associated with each transition specifies whether this should be done. Only if the flag is *yes* for all transitions in the maximal, is *NumComb* used.

3.3. Resources and Reachability Graphs

Each transition has a set (possibly empty) of named resources. A named resource can be associated with more than one transition. Whenever one of those transitions is firing, the resource is in *use.* The number of those transitions firing simultaneously is the current number of *usages* of that resource.

By using the rules above for generating next states, we can determine the set of *reachable states* for a given initial state. By placing directed edges from parent states to child states, the set of reachable states can be viewed as a *reachability graph.* By building and analyzing the net's reachability graph we can find the average number of uses of a resource over time. This average, if properly implemented and interpreted, can be used to obtain a variety of meaningful performance estimates.

3.4. An Example

The example in Figure 3.1 models users at terminals who, with a geometric think time, generate requests for a server. There is one token on place P1 for each user. Transitions T1 and T2 implement the think time. Note that the GTPN can represent geometric, as well as constant, holding times. Transition T3 implements a load-dependent server with a firing duration that depends on the number of tokens on P2.

In Figure 3.2 and Table 3.1 we show the reachability graph for the net in Figure 3.1 assuming there is only one user. The labels on the edges of the graph are the next state probabilities. The labels on the vertices of the graph are the values for time-in-state. The marking vectors are shown in the table. The RFT sets are shown as a list of pairs with one pair per in progress firing of a transition. The first component of each pair is the name of the transition. The second component is the remaining firing time. The resources used and their number of uses are also shown in the table.

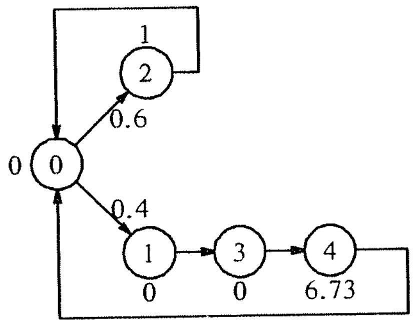

Figure 3.2. Reachability Graph for example

Table 3.1. Reachable States for example

States	Marking			RFT Set	Resources
	P1	P2	P3		
0	1	0	1	{}	{}
1	0	0	1	{(T1,0.0)}	{}
2	0	0	1	{(T2,1.0)}	{Terminal(1)}
3	0	1	1	{}	{}
4	0	0	0	{(T3,6.73)}	{Server(1)}

4. Multiprocessor Analysis

We have analyzed the performance of mutiprocessor systems using the GTPN. In this section we define our performance measure of speedup, describe the GTPN model used, and summarize the results of our experiments. We modeled several systems that have been studied previously for the sake of comparison. First, we conducted four sets of validation experiments. Second, we compared our results with those of Marsan, Balbo,and Conte[10] which assume exponential access time. The important measure of *speedup* has not been studied in the stochastic modeling literature cited in Section 2. Consequently, our third set of experiments look at speedup for some representative systems. Finally, we examine the performance of a particular class of non-uniform access probabilities called *favorite memory.*

4.1. Measures and the GTPN net

Recall that previous evaluations of multiprocessors using stochastic models have studied effective memory bandwidth and a measure of processing power defined as: processor utilization summed over all processors. We are more interested in a different measure of processing power: processor productivity summed over all processors. To avoid confusion in the discussion below, we will call our measure *speedup*, since it is the same as the speedup measure used in the non-stochastic literature on multiprocessors. We will use the term processing power in the sense defined in previous studies. We argue that speedup is a more important single measure of system performance because the goal of multiprocessing is speeding up a program, not achieving high memory or processor utilization. Effective memory bandwidth and processing power are also easily computed for our GTPN models as we show below.

The GTPN model used in the analysis assuming uniform access probabilities and a multibus is shown in Figure 4.1 and Table 4.1. The net for the non-uniform access case is a slight modification of this one. The net shown is for a system with three processors, two memories, and one bus (P2). It is the model in Marsan, Balbo, and Conte[10], modified to support discrete time.

The tokens are shown in the initial state. The tokens in P5 represent free memories. The tokens

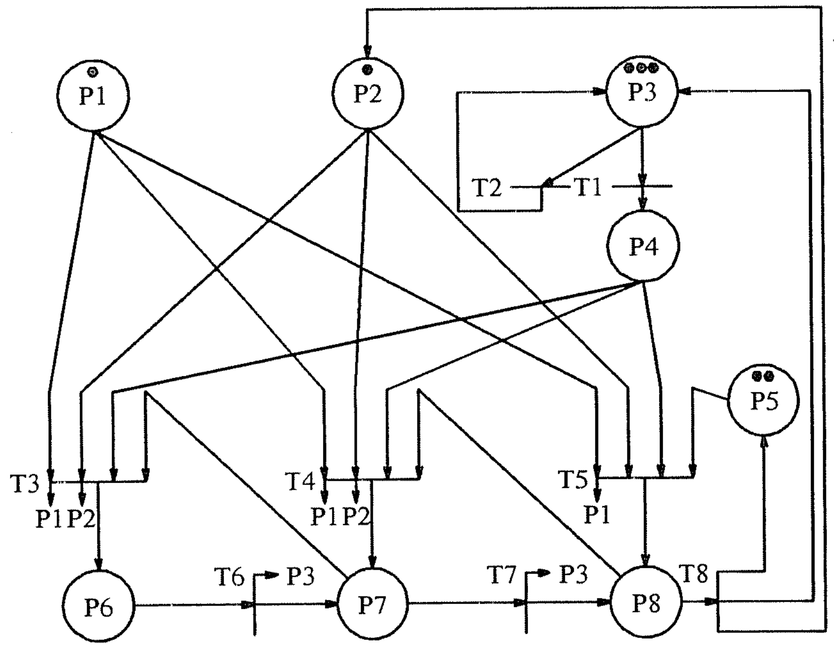

Figure 4.1. GTPN net for a 3 processor/2 memory/1 bus system. Uniform access.

Table 4.1. The attributes of each transition in Figure 4.1.

Transition	Duration	Frequency	Cnt Combs	Resources
T1	0.0	MRP	yes	(Spd,PP)
T2	1.0	1 - MRP	yes	()
T3	0.0	P7/3	no	()
T4	0.0	P8/3	no	()
T5	0.0	P5/3	no	()
T6	1.0	(((P2 = 0) | (P4 = 0) | (P5 = 0)) &	no	(Spd, MemBdwth)
		(P3 = 0) & (T1 = 0)) * 1.0		
T7	1.0	same as T6	no	(Spd, MemBwdth)
T8	1.0	same as T6	no	(Spd, MemBwdth)

in P3 represent processors that are active locally. The tokens in P2 represent free buses. Transitions $T1$ and $T2$ implement a geometric processing time between memory requests. (Note that the GTPN can represent geometric, as well as constant, holding times.) With probability equal to the memory request probability, each token in P3 moves to P4. A token in P4 represents a processor making a memory request. The places along the bottom represent lengths of memory queues. Because the memory modules are statistically identical we are just interested in the possible combinations of queue lengths. For example, a token in the leftmost place signifies that all three processors are waiting for the same memory module. In this case, there can be no other tokens along the bottom places. We want each token on P4 to have its memory request uniformly distributed among the memory modules. If only one transition can start firing at a time, then the frequency expressions for *T*3, *T*4, and *T*5 ensure uniformity. The place *P*1 is used to enforce that only one token on P4 at a time moves to the bottom row (with zero delay). As tokens move across the bottom, processors have their memory requests granted and return to P3. The last processor to use a memory module (T8) returns the bus token to P2.

The frequency expressions for the transitions along the bottom enforce that none of the transitions along the bottom row start firing until all possible tokens on P4 are moved into the memory subsystem. In these frequency expressions a vertical bar represents a logical or. When the number of tokens on *P*2 or *P*4 or *P*5 is zero and the number of tokens on *P*3 is zero, and there are no firings of *T*1, the expression evaluates to one, otherwise it evaluates to zero. The Cnt Combinations column of Table 4.1 contains the value of the flag that determines how probabilites for maximals are to be calculated for each transition that may appear in a maximal. Three resources are used to derive performance measures. *Spd* generates speedup. *PP* generates processing power. *MemBdwdth* generates the effective memory bandwidth.

4.2. Model Validations

We first consider four previous studies that assume constant cycle time and uniform access, in order to validate our model. Bhandarkar[1] gives exact numerical results for the effective memory bandwidth up to a 8 processor/8 memory/crossbar and memory request probability (MRP) of 1. In Table 4.2 we present his numbers and the results from our GTPN; they agree.

The GTPN model for 16 processors and 16 memories yielded a Markov Chain with 8115 states. Bhandarkar's approach yielded a Markov Chain with 300,540,195 states. As mentioned in section 2.1, Bhandarkar needed this large state space as a means of indirectly reaching a smaller state space. The GTPN allows us to describe the system such that we can directly derive the smaller state space. This explains the difference in state space sizes. A similar direct method was used by the GSPN[10]. We feel that it is quite likely that deriving the smaller state space without the

Table 4.2. Effective memory bandwidth. Crossbar. MRP = 1.

Processors	Memories	Bhandarkar	GTPN
2	2	1.5000	1.5000
4	4	2.6210	2.6210
6	6	3.7809	3.7809
8	8	4.9471	4.9471
10	10	-----	6.1150
12	12	-----	7.2835
14	14	-----	8.4527
16	16	-----	9.6225

Table 4.3. Effective memory bandwidth. 16 processor/16 memories/Multibus. MRP = 1.

Buses	Towsley Approx	Towsley Sim	Exact
8	7.93	7.96	7.977
9	8.73	8.80	8.825
10	9.27	9.34	9.357
11	9.53	9.58	9.566
16	9.62	9.66	9.623

GTPN is possible. Our point is that the GTPN aids in seeing and expressing the symmetry which allows the direct derivation.

We use the Power Method, an iterative sparse matrix algorithm, to solve for our results. The iterations terminate when the sum over all states of the absolute value of the difference between the last two iterations, is less than the convergence criterion. With our default convergence criterion, 5×10^{-5}, all of our values agreed with Bhandarkar's except in the 8 processor/8 memory case, where we reached 4.9469. We repeated the analysis with a smaller convergence criterion, 5×10^{-6}, and reached Bhandarkar's value. We have used our default convergence criterion in all of the other experiments reported in this paper. The default should be accurate to at least three digits. We round all of our remaining results to three digits. This should also be sufficiently accurate since all of the remaining comparisons are to simulations and approximate solutions.

Towsley[3] gives approximate solutions and simulation values for effective memory bandwidth for a 16 processor/16 memory system with a MRP of one and a multibus interconnect. Our exact results are compared with his results in Table 4.3. Our values are in each case within the 90% confidence interval of his simulation, and provide further evidence that the approximate analysis is accurate. A reasonable conclusion is that up to 6 buses can be removed from the crossbar with only a small performance degradation. Lang, Valero, and Alegre is another study that arrives at a similar conclusion.

Table 4.4. Effective memory bandwidth. 8 processor/8 memory. MRP = 0.5.

Buses	Lang Sim	Bhuyan Approx	Mudge Approx	Exact
1	1.00	1.00	0.98	1.000
2	2.00	1.97	1.88	2.000
3	2.87	2.79	2.57	2.898
4	3.33	3.27	2.99	3.352
5	3.45	3.44	3.16	3.458
6	3.47	3.47	3.22	3.469
7	3.47	3.47	3.23	3.469
8	3.47	3.47	3.23	3.469

Table 4.5. Effective memory bandwidth. 16 processor/16 memory.

Buses	Mean IRT	Goyal Sim	Goyal Approx	Exact
1	8	1.00	0.9997	0.9998
	16	0.8584	0.8719	0.8588
	32	0.4794	0.4811	0.4745
2	4	2.0000	2.0000	1.9983
	8	1.6616	1.6655	1.6560
	16	0.9316	0.9331	0.9365
	32	0.4852	0.4847	0.4793

We now present two validations that involve a MRP of less than one and a multibus. First, we consider a 8 processor/8 memory system with a MRP of 0.5. Effective memory bandwidth is the measure reported in previous papers. Table 4.4 gives the simulation values in Lang, Valero, and Alegre[18], the approximate values of Bhuyan[19], the approximate values of Mudge, Hayes, Buzzard, and Winsor[20], and our exact results. Our exact values are within the 99% confidence intervals of the simulation results in all cases, and provide still better values for evaluation of approximate results. Note that, again, the number of buses can be reduced substantially from a crossbar with minimal effect on performance.

Second, we consider a 16 processor/16 memory system with one or two buses. In Table 4.5 we show Goyal and Agerwala's[2] values and ours. In this table we adopt their convention of using the mean interrequest time (mean $IRT = \frac{1}{MRP} - 1$) instead of the memory request probability. Our values and theirs agree within the range of statistical error.

4.3. Comparison with Exponential Memory Access Time Models

Recall from Section 2 that several studies have assumed an exponential memory access time and have derived exact performance estimates using continuous time Markov Chains. There are two possible reasons for the exponential assumption. One, is that the multiprocessor under study has an exponential memory access time. Two, is that the multiprocessor under study has a constant memory access time, but that assuming an exponential memory access time is a reasonable approximation which yields models that can be solved exactly. In this paper we are interested in the stochastic modeling of multiprocessors with constant memory access times. Consequently, we are interested in the second reason. One would expect from queueing theory (as noted in Marsan and Gerla [11]) that the model with constant access time will give higher predictions for speedup. We conducted several experiments to see how large a difference the assumption of exponential access time makes.

Marsan, Balbo, and Conte[10] gives exact results for a 12 processor/2 bus system. They vary the number of memories and the *load*. They assume that the interrequest time is exponentially distributed with rate λ and the memory access time is exponentially distributed with rate μ. The load is the ratio, ρ, of λ to μ. Our approach can be compared to theirs. We assume a constant memory access time and an interrequest time which is, strictly speaking, a modified geometric random variable. The important step is to make our models as similar as possible, so that only the difference in modeling the memory access time is observed. In particular, we need to represent the interrequest time distribution accurately.

In the limit, as the time length of a trial goes to zero, a modified geometric random variable

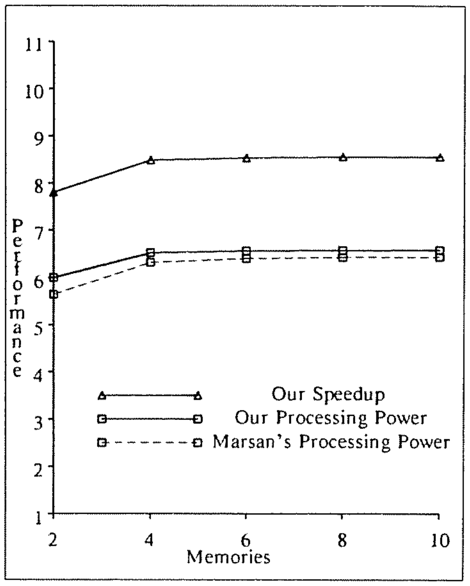

Figure 4.2. Expo. access time vs. constant cycle time. 12 processors/2 buses. Load is 0.3. MRP = 0.231.

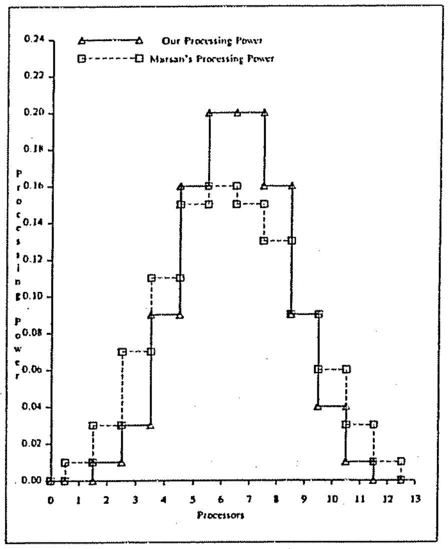

Figure 4.3. Probability Distributions for the 6 memory case of Figure 4.2.

is identical with the exponential random variable with same mean. Consequently, if trials are "reasonably frequent", then a modified geometric random variable is a good approximation to the exponential random variable with the same mean. We can approximate the exponential memory interrequest time arbitrarily closely in our GTPN model, by decreasing the duration of transition T2 and adjusting the frequency expressions for transitions T1 and T2 appropriately. Furthermore, for a selected duration of transition T2 (greater than zero), the variance of the modified geometric distribution is larger than the variance of the exponential distribution we are approximating. The increased contention due to this larger variability will result in lower estimates of processing power than if the exponential interrequest time were represented exactly. Since we expect (and observe) that constant memory access time model will have a higher processing power than the exponential memory access time, the differences we observe due to approximating the interrequest time will be conservative. (We verified this experimentally.)

In each of the experiments we conducted, a memory access time of 1 and an interrequest time of $1/\rho$ approximated using a duration of 1 for T2, yields a reasonably accurate representation. We note that this selection of parameters was used successfully in the validation against Goyal and Agerwala (in Table 4.5). Goyal and Agerwala's simulation assumed that the interrequest time is exponentially distributed.

Figure 4.2 shows their estimates of processing power and ours, as the number of memories is varied, for a 12 processor/2 bus system with load of 0.3. The difference in estimates is 6% at 2 memories and decreases to 2% at 10 memories. Thus, the exponential access time assumption underestimates but provides a good approximation of the expected value for processing power. Our estimates for *speedup* are also shown. These results are qualitatively similar, but substantially larger than the processing power results. If we are really interested in speedup, processing power is a poor approximation.

Though expected values are important, the nature of the probability distribution of processing power is useful in characterizing multiprocessor behavior. In Figure 4.3 we show the constant access time and approximating exponential access time probability distributions for a 12 processor/2 bus/6 memory system with load of 0.3. The distributions are substantially different. As one might expect, the distribution assuming an exponential access time has a higher variance.

Marsan and Chiola [36], concurrently with our work, have introduced deterministic firing times into the GSPN under certain restricted conditions. Those restricted conditions imply that their multiprocessor models can only allow one bus. They reach conclusions similar to ours for the one bus case. Thus, our work may be viewed as a generalization of theirs to the case of an arbitrary number of buses.

4.4. Critical Memory Request Probability

We now describe the analyses we conducted that are not comparisons with previous studies. Our first set of experiments measured *speedup* for a 10 processor/10 memory system. The memory request probability is varied from 0.1 to 1.0. The number of buses is 1, 2, 3, 4, and 10. Our results in Figure 4.4 suggest an important conclusion about the effect of the number of buses on speedup. When the number of buses is small, a *critical memory request probability* appears to exist. The horizontal line drawn at speedup = 8.75 indicates approximately where this critical MRP lies on each curve. Below that probability, speedup is close to that with a crossbar(even for just two buses). Above that probability, speedup rapidly decreases and is equal to the number of buses in the limiting case. This rapid decrease is clearly due to the lack of buses. The drop is more gradual as the number of buses increases and is to a larger and larger extent due to memory contention instead of bus contention. We note that a functional relationship may exist between the number of processors, memory modules, and buses, and the critical MRP. Further study is required to determine whether this is true.

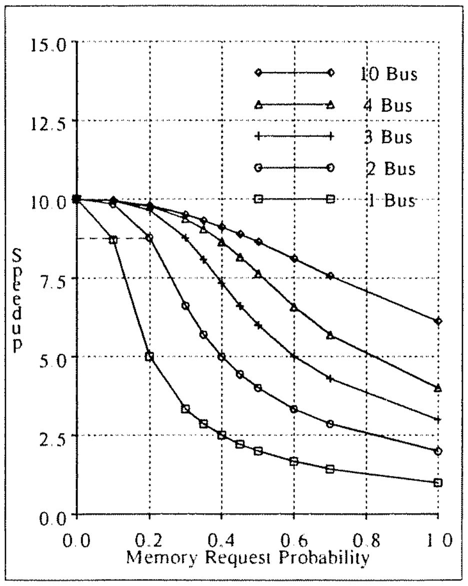

Figure 4.4. Measure is speedup. 10 processor/10 memory. Uniform access.

Our results are more specific than the conclusion reached by Lang, Valero, and Alegre. With respect to the measure of effective memory bandwidth, they concluded that good performance is possible with the number of buses equal to one half the number of processors with a MRP of 0.5. Note that their conclusion is supported by Figure 4.4. Furthermore, we conclude that as long as the memory request probability stays below the critical value, only a few buses are needed to have close to the performance of a crossbar.

4.5. Non-Uniform Access Probabilities

All of the experiments above assume uniform access probabilities. Many authors have argued that this assumption is reasonable because the memory modules presumably are interleaved by the low-order bits of the memory addresses. Rau's [37] trace driven simulations, however, show that, at least in some cases, even with memory module interleaving, accesses are not uniform. Consequently, several studies have considered the non-uniform case. One version of non-uniformity that is of interest is called *favorite memory.* In favorite memory, there is one memory module, say module i, that is accessed with a different frequency than the other modules by all processes. Module i has probability, α, of being accessed while the probability of each other module being accessed is uniformly distributed over $1 - \alpha$, for all processors.

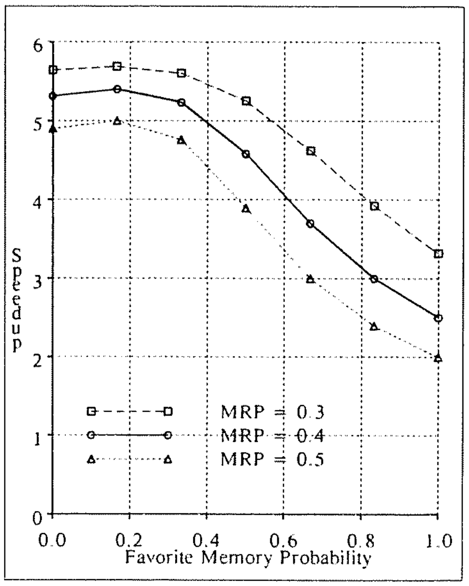

Figure 4.5. Favorite Memory nonuniform accesses. 6 processor/6 memory/3 bus.

We conducted an experiment assuming a favorite memory. We considered a system with 6 processors, 6 memories, and 3 buses. Results are given in Figure 4.5 for memory request probabilities of 0.3, 0.4, and 0.5. Each curve has seven data points for when zero, one sixth, two sixths, up to six sixths of the memory requests are directed to the favorite memory. Note that a modest favoritism (i.e. two sixths) has only a small effect on speedup. As expected, the speedup decreases as the favoritism increases and as the memory request probability increases. In addition, as the memory request probability increases the importance of favoritism increases, causing speedup to decrease more rapidly.

This favorite memory experiment illustrates the need to develop approximate solution techniques based on the GTPN. Identifying one memory module as favorite causes a significantly larger state space than when all the modules are identical. For example, this 6 processor/6 memory/3 bus system has 3384 states while a 6 processor/6 memory/3 bus system without a favorite memory has only 496 states.

5. Conclusions

We have presented exact performance estimates for models of multiprocessors for which only approximate and simulations estimates existed. These models include the important properties of constant memory access time, memory request probabilities less than one, and bus contention. One form of non-uniformity in the memory access probabilities was also treated. We derived these results by using a method of introducing time into Petri Nets, called Generalized Timed Petri Nets(GTPN), that we have developed. This method is efficient for moderate size state spaces. For example, a multiprocessor model with 12 processors, 10 memories, 2 buses, and a geometric interrequest time of 5 time units has 2026 reachable states and requires 274 seconds to build the reachability graph and analyze it for performance estimates. The results we have derived illustrate the advantages of the GTPN model in specifying instantaneous, constant, and geometric holding times in analytical system models. If constant delays are not needed, or if they satisfy restrictions in current SPN models [38,39], then the SPN models may be more advantageous due to smaller state spaces [8,9].

The previous stochastic modeling studies of multiprocessor memory and bus interference have measured effective memory bandwidth, and processing power as defined by: processor utilization times the number of processors. We suggest a better measure of processing power which is equivalent to the measure of speedup that is commonly used in other bodies of literature on multiprocessors.

Our multiprocessor performance estimates provided several important insights. One is that assuming an exponential access time for a model of a multiprocessor with constant memory access time and any number of buses causes only a small underestimation of the expected value of processing power. However, the probability distributions for processing power differ substantially. The distribution assuming an exponential access time has a higher variance.

Two, is that at low request rates only a few buses are needed to have almost the performance of a crossbar. However, when only a few buses are used, a critical request rate exists. Exceeding that critical request rate causes a dramatic collapse in performance.

Many performance questions remain with respect to multiprocessor memory and bus interference. One focus of our current research is to determine how much more information about multiprocessor memory and bus interference can be obtained by the GTPN technique.

Our modeling approach does have a major drawback. As in any approach that builds an entire state space, many interesting models cannot be studied because their state spaces are too large. Our current implementation supports state spaces with up to 40,000 states. We could raise this limit to some extent, but the basic problem remains. Approximate solutions must play an important role. One focus of our future research will be to study how GTPNs can be used to provide approximate solutions.

Acknowledgements

Many people have made helpful comments and suggestions during our research. We would like to especially thank M. Ajmone Marsan and Giovanni Chiola for the use of their GSPN tool, for their noticing some errors in our initial multiprocessor net, and for their noticing, in the restricted case of one bus, the difference in processing power probability distributions due to the exponential access time assumption.

References

[1] D.P. Bhandarkar, "Analysis of memory interference in multiprocessors," *IEEE Trans. Comput.*, vol. C-24, pp. 897-908, Sept. 1975.

[2] A. Goyal and T. Agerwala, "Performance Analysis of Future Shared Storage Systems," *IBM J. Res. Develop.*, vol.28-1, pp. 95-108, Jan. 1984.

[3] D. Towsley, "An approximate analysis of multiprocessor systems," in *Proc. 1983 ACM SigmetricsConf. on Meas. and Mod. Comput. Syst.*, Aug. 1983, pp. 207-213.

[4] D.W. Yen, J.H. Patel, and E.S. Davidson, "Memory interference in synchronous multiprocessor systems," *IEEE Trans. Comput.*, vol. C-31, pp. 1116-1121, Nov. 1982.

[5] C.A. Petri, "Kommunikation mit Automaten," *Schriften des Rheinisch- Westfalischen Institute fur Instrumentelle Mathematik an der Universitat Bonn*, Heft 2, Bonn, W. Germany, 1962; translation: C.F. Greene, Supplement 1 to Tech. Report RADC-TR-65-337, Vol. 1, Rome Air Development Center, Grifiss Air Force Base, NY 1965.

[6] W.M. Zuberek, "Timed Petri nets and preliminary performance evaluation," in *Proc. 7th Annu. Symp. Comput. Architecture*, pp. 88-96, 1980.

[7] R.R. Razouk and C.V. Phelps, "Performance Analysis Using Timed Petri Nets," in *Proc. 1984 Int Conf. on Parallel Processing*, pp. 126-129, August, 1984.

[8] M.A. Holliday and M.K. Vernon, "A Generalized Timed Petri Net Model for Performance Analysis," in *Proc. Int. Workshop on Timed Petri Nets*, July 1985.

[9] M.A. Holliday and M.K. Vernon, "A Generalized Timed Petri Net Model for Performance Analysis," Tech. Rep. 593, Comp. Sci. Dept., UW-Madison, May 1985 (submitted for publication).

[10] M.A. Marsan, G. Balbo, and G. Conte, "A Class of Generalized Stochastic Petri Nets," *ACM Trans. on Computer Systems*, vol. 2, pp. 93-122, May 1984.

[11] M.A. Marsan and M.Gerla, "Markov models for multiple-bus multiprocessor systems," *IEEE Trans. Comput.*, vol. C-31, pp.239-248, Mar. 1982.

[12] M.A. Marsan, G. Balbo, and G. Conte, "Comparative performance analysis of single bus multiprocessor architectures," *IEEE Trans. Comput.*, vol. C-31, pp. 1179-1191, Dec. 1982.

[13] M.A. Marsan, G. Balbo, G. Conte, and Gregoretti, "Modeling Bus Contention and Memory Interference in a Multiprocessor System," *IEEE Trans. Comput.*, vol. C-32, pp. 60-72, Jan. 1983.

[14] W.D. Strecker, "Analysis of the instruction execution rate in certain computer structures," Ph.D. dissertation, Carnegie-Mellon Univ., Pittsburgh, PA, 1970.

[15] W.A. Wulf and C.G. Bell, "C.mmp – A mulit-mini-processor," in *Fall Joint Comput. Conf. AFIPS Conf. Proc.*, vol. 41, pt. 2, 1972, pp.765-777.

[16] F.S. Baskett and A.J. Smith, "Interference in multiprocessor computer systems with interleaved memory," *Commun. Ass. Comnput. Mach.*, vol. 19, pp. 327-334, June 1976.

[17] B.R. Rau, "Interleaved memory bandwidth in a model of multiprocessor computer systems," *IEEE Trans. Comput.*, vol. C-28, pp. 678-681, Sept. 1979.

[18] T.Lang, M. Valero, and I. Alegre, "Bandwidth of crossbar and multiple-bus connections for multiprocessors," *IEEE Trans. Comput.*, vol. C-31, pp.1227-1234, Dec. 1982.

[19] L.N. Bhuyan, "A Combinatorial Analysis of Multibus Multiprocessors," in *Proc. 1984 Int. Conf. on Parallel Processing*, Aug. 1984, pp.225-227.

[20] T.N. Mudge, J.P. Hayes, G.D. Buzzard, and D.C. Winsor, "Analysis of Multiple Bus Interconnection Networks," in *Proc. 1984 Int. Conf. on Parallel Processing*, Aug. 1984, pp. 228-232.

[21] D.P. Bhandarkar and S.H. Fuller, "Markov chain models for analyzing memory interference in multiprocessor systems," in *Proc. 1st Annu. Symp. Comp. Arch.*, pp. 1-6, Dec. 1973.

[22] I.H. Önyüksel and K.B. Irani, "A Markovian queueing network model for performance evaluation of bus-deficient multiprocessor systems," in *Proc. 1983 Int. Conf. on Parallel Processing*, Aug. 1983, pp.437-439.

[23] P.A. Jacobson and E.D. Lazowska, "Analyzing queueing networks with simultaneous resource possession," *Commun. Ass. Comput. Mach.*, vol. 25, pp.142-151, Feb. 1982.

[24] K.B. Irani and I.H. Önyüksel, "A Closed Form Solution for the Performance Analysis of Multiple-Bus Multiprocessor Systems," *IEEE Trans. Comput.*, vol. C-33, pp. 1004-1012, Nov. 1984.

[25] M.K. Molloy, "On the integration of delay and throughput measures in distributed processing models," Ph.D dissertation, Univ. California, Los Angeles, 1981.

[26] T.N. Mudge and H.B. Al-Sadoun, "Memory Interference Models with Variable Connection Time," *IEEE Trans. Comput.*, vol. C-33, pp. 1033-1038, Nov. 1984.

[27] A.S. Sethi and N.Deo, "Interference in multiprocessor systems with localized memory access probabilities," *IEEE Trans. Comput.*, vol. C-28, pp. 157-163, Feb. 1979.

[28] H.C. Du and J.L. Baer, "On the performance of interleaved memories with non-uniform access probabilities," in *Proc. 1983 Int. Conf. on Parallel Processing*, Aug. 1983, pp. 429-436.

[29] C.H. Hoogendoorn, "A general model for memory interference in multiprocessors," *IEEE Trans. Comput.*, vol. C-26, pp. 998-1005, Oct. 1977.

[30] T.N. Mudge and B.A. Makrucki, "Probabilistic analysis of a crossbar switch," in *Proc. 9th Int. Symp. on Comput. Arch.*, Apr. 1982, pp. 311-320.

[31] K.O. Siomalas and B.A. Bowen, "Performance of crossbar multiprocessor sytems," *IEEE Trans. Comput.*, vol. C-32, pp.689-695, July 1983.

[32] L.N. Bhuyan, "An Analysis of Processor-Memory Interconnection Networks," *IEEE Trans. Comput.*, vol. C-34, pp.279-283, March 1985.

[33] C. Ramchandani, "Analysis of Asynchronous Concurrent Systems by Timed Petri Nets," Ph.D. Thesis, MIT, 1974.

[34] P. Chretienne, "Les réseaux de Petri temporisés," Thèse d'état, Paris-6 University, June 1983.

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

[36] M.A. Marsan and G. Chiola, "On Petri Nets with Deterministic and Exponential Transition Firing Times," private communication, July 1985.

[37] B.R. Rau, "Program Behavior and the Performance of Interleaved Memories," *IEEE Trans. Comput.*, vol. C-28, pp.191-199, March 1979.

[38] J.B. Dugan, K.S. Trivedi, R.M. Geist, and V.F. Nicola, "Extended Stochastic Petri Nets: Applications and Analysis," *Performance 84*, pp. 507-519, Paris, France, December 1984.

[39] M.A. Marsan, G. Chiola, and G. Conte, "Generalized Stochastic Petri Net Models of Multiprocessors with Cache Memories," *1st Int. Conf. on Supercomputing Systems*, December 1985.