ANALYSIS OF MEMORY INTERFERENCE IN BUFFERED MULTI-PROCESSOR SYSTEMS IN PRESENCE OF HOT SPOTS AND FAVORITE MEMORIES

THESIS

Presented to the Graduate Council of the
University of North Texas in Partial
Fulfilment of the Requirements

For the Degree of

MASTER OF SCIENCE

By

Sanjoy Kumar Sen, B.E.
Denton, Texas
August, 1995

ANALYSIS OF MEMORY INTERFERENCE IN BUFFERED MULTI-PROCESSOR SYSTEMS IN PRESENCE OF HOT SPOTS AND FAVORITE MEMORIES

THESIS

Presented to the Graduate Council of the
University of North Texas in Partial
Fulfilment of the Requirements

For the Degree of

MASTER OF SCIENCE

By

Sanjoy Kumar Sen, B.E.
Denton, Texas
August, 1995

Sen, Sanjoy Kumar, Analysis of Memory Interference in Buffered Multiprocessor Systems in Presence of Hot Spots and Favorite Memories. Master of Science (Computer Science), August 1995, 19 tables, 16 illustrations, bibliography.

In this thesis, a discrete Markov chain model for analyzing memory interference in multiprocessors, is presented. Each memory module is either hot or favorite or neither of these two. Since the state space of such a Markov model for an $N \times M$ system, where N and M are respectively the number of processors and memory modules, becomes exorbitant for large N and M, we restrict our analytical solutions to $2 \times M$ and $N \times 2$ systems. The general case is analyzed using simulation. In all cases, the effective memory bandwidth, mean memory-queue-length and mean-waiting-time for a memory request are derived. A heuristic is presented, using a probabilistic model, which finds the number of hot modules beyond which there is hardly any bandwidth change.

ACKNOWLEDGEMENTS

I am immensely grateful to my advisor, Dr. Sajal K. Das, without whose unceasing support, encouragement and guidance this thesis could not have been written. The several months of fruitful discussions I had with him, helped me get a better understanding of the subject and also imparted the spirit of perseverance. My association with him had been a very rewarding experience which I will always treasure.

I am grateful to the members of my supervising committee, Dr. Weiping Shi and Dr. Farhad Sharokhi for their review and helpful suggestions. I am indebted to the Chair of the Department of Computer Sciences, Dr. Ken Godwin, as well as my advisor, Dr. Sajal K. Das, for the financial support during the period of my studies and for providing all neccessary computing facilities. Special thanks are due to Mr. Falguni Sarkar for his invaluable suggestions during the course of this work. I also thank all my friends in UNT for the wonderful time I had.

Finally, I express my deepest gratitude to my father, mother and sister, without whose endless affection, support and inspiration from a long distance, I could not imagine completing this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Processor memory interconnection in shared memory multiprocessor systems plays a vital role in characterizing the system performance. A shared memory multiprocessor consists of a set of processors $\{P_0, P_1, ...P_{N-1}\}$ and a set of memory modules $\{M_0, M_1, ...M_{M-1}\}$. The processor- memory communications are established either through a set of switches (*e.g.* crossbar or multistage interconnection network) or through global shared (multi) bus. In this work, both types of connections between the processors and memories will be called under the generic name *interconnection network.* A logical diagram of such a shared memory system is shown in Figure 1.1.

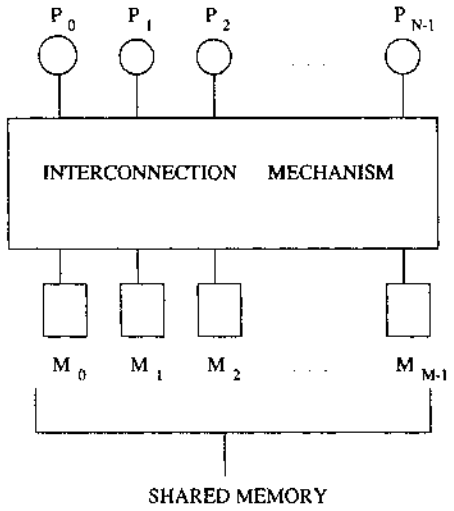


Figure 1.1: Logical diagram of Processor Memory Interconnection

The interconnection mechanism should allow efficient resource sharing among the processors. Conflict arises when more than one processor attempt to access memory modules using the same path or switching module in the network. The primary goal of any interconnection network is to minimize this conflict, leading to higher acceptance rate of memory requests by the processors. The effective memory *bandwidth*, BW,

is defined as the average number of busy memory modules in each memory cycle. Of course, the BW is also dependent on the request rate of the processors. Other performance indices of such a network include processor and memory utilization, expected queue length, fault tolerence, cost etc.

At one end of the bandwidth spectrum are the single *shared bus* systems. Although they provide extremely low transfer rates and minimal fault tolerence, they are very cost effective. However the scalability of such a system is poor in the sense that the number of processors that can be supported on a single bus is rather limited without degrading the performance. At the other extreme are the *cross-bar* systems which provide complete connection among processor-memory pairs. But the total cost of the cross-bar switches increases rapidly as the number of processor and/or memory modules increase. *Multistage interconnection networks* (MINs) have been developed as a resort to trade-off between the high BW and fault-tolerence of crossbar switches and the low cost of single bus systems. MINs provide a unique path between each processor-memory pair, reducing the number of switching modules considerably than a cross-bar system connecting the same number of processors and memories. Also a natural extension of the single bus system, with the same goal, is the *multiple bus interconnection networks*, called MBIN. For details on MBINs, refer to [19,20,21,22].

In a processor-memory system, contention may occur either in the interconnection mechanism (e.g. bus or MIN switches) itself or while accessing an actual memory module. By some arbitration policy (usually random), one of the conflicting requests is granted, while others are either rejected (the case of a *non-buffered* system), or queued for future processing (the case of a *buffered* system).

In the past two decades, the performance of processor-memory interconnections has been studied widely using various analytical models, some of which are *exact* [1,2,4,5,7] while others are *approximate* [3,8,9]. An exact analysis assumes that a rejected memory request from a processor is not discarded but is queued up until it is serviced by the requested memory module (buffered systems). The exact models

use Petri nets [25], Markov chains [1,2], queuing networks [22] and so on. An approximate analysis, on the other hand, assumes that the rejected requests are discarded (non-buffered systems). This suggests using probabilistic approach to analyze the performance of the system [3,8].

The memory reference patterns of the processors can be either *uniform* or *non-uniform*. Memory reference pattern is said to be uniform if all the processors have the same probability of accessing any memory module. This is a valid assumption if address interleaving on the low order address bits are used. Without the address interleaving, however, the memory reference pattern will be *non-uniform* in most cases and will depend on the locality of reference.

1.1. Non-Uniform Memory Access Patterns

The non-uniform memory access patterns can be either intra-cycle or inter-cycle. In the *intra-cycle* case, the distribution of the memory requests by the processors in a particular memory cycle is not uniform. There are four types of intra-class non-uniformity. In the first type, each processor may have a *favorite* memory module(s) which it accesses more frequently than the others [8]. This is quite logical considering the spatial locality principle of memory references where address space is sequential in each memory module.

In the second type, all the processors might access a particular memory module or a class of memory modules more frequently than the others. This might capture the effect of the processors sharing a common variable or a sysnchronization primitive (e.g. barrier synchronization). The particular memory module or the class of memory modules accessed more frequently is called *hot spot(s)* [16].

The third type of non-uniformity capturing the temporal locality principle of memory reference is defined as follows [2]. If the nth request of a processor is for memory module M_i, then the $(n+1)th$ request will be for module M_i with probability α and for module M_j (where $j \neq i$) with probability $\frac{(1-\alpha)}{(M-1)}$, where M is the total number of

memory modules in the system. This assumes that $\alpha > \frac{1}{M}$, i.e. the probability that the next memory request of a processor is to the same memory module is greater than the probability in case of uniform memory reference pattern and also greater than the probability that the next reference is to a different memory module. If $\alpha = \frac{1}{M}$, the model reduces to the uniform reference model as a special case.

A fourth type of non-uniformity, also called *spatial* non-uniformity, is defined when the events that a particular memory module has at least one request are not independent [20].

The *inter-cycle* non-uniformity of memory references occurs when the event that a processor requests a particular memory module is dependent on whether the same processor's memory request is satisfied in the previous cycle. If a processor's request to a particular memory module cannot be satisfied in the current memory cycle due to the fact that the particular module is already busy, then in the next cycle, the same request might be placed to the same memory module or it might be randomly distributed to any of the other available modules. In the former case we have to associate a buffer with each memory module [1,2,4], to store the rejected requests for the next cycle, or there should be some mechanism (sort of 'feedback') to remember the destinations of the rejected memory references and again place it to the same destination in the next cycle [7].

If a memory module is classified as any one of the following types, H (hot), NH (non-hot), F (favorite) and NF (non-favorite), then the various combinations of memory modules is given as $\{H, NH\} \times \{F, NF\}$, which when expanded gives the following set of combinations of memory modules: $\{(H, F), (H, NF), (NH, F), (NH, NF)\}$.

The system where the memory modules are of the type (NH, NF) is analyzed in [1,4,5,6]. The concept of hot memory modules is introduced by Pfister and Norton [16]. In the context of analyzing contention in the multistage interconnection network, they proposed a scheme to avoid tree saturation, a phenomenon occurring in systems consisting of modules of type (H, NF). The analysis of systems consisting of modules

of type (NH, F) is carried out in [2,8]. In this work, we will deal with the problem of contention in accessing the shared memory modules, after the request has successfully passed through the interconnection mechanism. See the taxonomy given at the end of Chapter 2 for a glimpse of the related work in this area.

1.2. Contribution of this Thesis

We carry out an analysis of an $N \times M$ system with memory modules of type (H, NF) and (H, F) in a unified manner. Our unified scheme gives analytic solutions for bandwidth, mean queue length and mean waiting time for a memory request by the processors, for $2 \times M$ and $N \times 2$ system. As our model is restricted to systems having only 2 processors in all cases excepting the $N \times 2$ system, the number of hot memory modules is limited to 1 in the analysis of systems with memory modules of type (H, F). For $N \times 2$ system, one (H, NF) and another (NH, NF) type memory modules are considered. The reason for not choosing favorite memory in the later case is that, we are not able to distinguish between a hot and a favorite memory in case of only one memory module. The general case of $N \times M$ system is analyzed using simulation studies, and the results are compared to the analytic solutions in the special cases.

Since it is expected that memory requests will accumulate in the queue of a hot memory module leading to deterioration of memory bandwidth, multiple hot-spots (K many) in the multiprocessor system should lead to a definite improvement in the system performance. But it is also shown that the bandwidth (BW) of the system stops improving significantly if the number of hot memory modules is increased beyond a certain value, less than $min(N, M)$. In this work, we also present a heuristic to find this upper bound on K. The heuristic, based on an approximate probabilistic model, actually estimates an asymptotic bandwidth of the system.

1.3. Chapter summary

Chapter 2 gives the glimpses of some of the existing work in this area. Our Markov model for buffered memory multiprocessors is described in Chapter 3. A step by step approach is taken for the development of the model, and each step is precisely described in various subsections. Chapter 4 presents various simulation results for the general case of the multiprocessor architecture. An approximate probabilistic analysis for the general case of the proposed model is given in Chapter 5, which leads to a heuristic for estimating the saturation value of K beyond which the system bandwidth does not show any significant improvement. Chapter 6 finally concludes this thesis.

CHAPTER 2

PREVIOUS WORK

Processor memory interconnection networks can be broadly classified into two types depending on the connection mechanism. In one type of networks commonly known as Multistage Interconnection Networks, the interconnection is through a set of switches. In the other type, the interconnection mechanism is by a single or multiple shared bus. We discuss some of the important previous works in both types of interconnection networks.

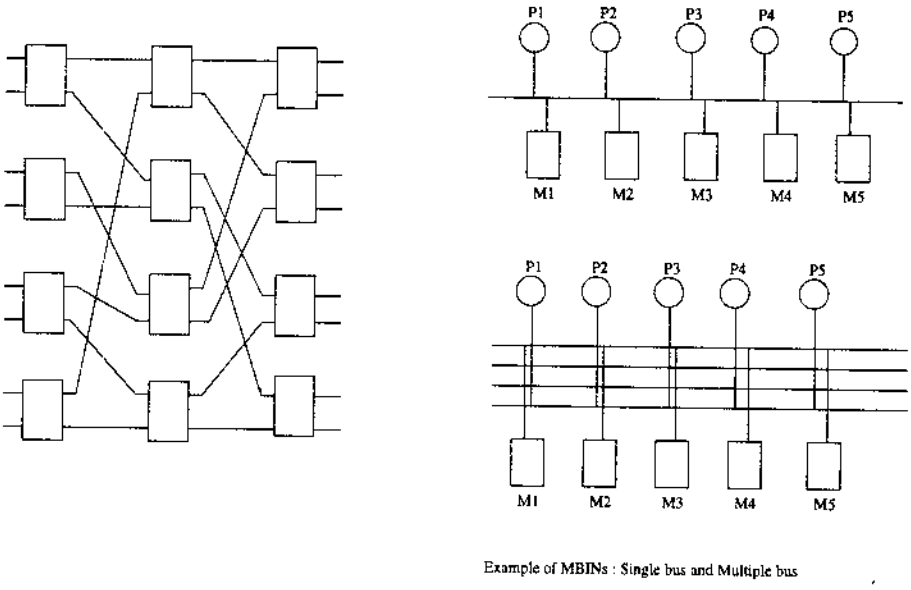


Figure 2.1: Examples of MIN and MBINs

2.1. Multistage Interconnection Networks

In this section we discuss multistage interconnection networks, popularly called MINs. A set of processors are logically connected to a set of memory modules through an interconnection network which is composed of high-speed switches. This is unlike

general interconnection networks studied for distributed systems, which have physical processor(s) sitting in each node. Each switching module may or may not have buffers to store the memory requests rejected during the current cycle. In the former case the memory requests in one cycle is independent of the requests in the previous cycle, while it is not so in case of buffered models of MIN. This latter model is, of course, more realistic. Again, within each memory cycle, the requests made by the processors may be independent which we call *uniform reference* model, or each processor may have a locality of reference which makes the memory requests non-uniform.

2.1.1. Unbuffered MINs

Uniform Reference models

Patel [3] presented a new form of processor-memory interconnection network called Delta network. A *Delta network* is an $a^n \times b^n$ switching network with n stages consisting of $a \times b$ cross-bar modules. The link pattern between the stages is such that there is a unique path of constant length from any source to any destination. A 2×2 crossbar switch has the capability of connecting the input I to either of the two outputs labelled 0 and 1 respectively, depending on the value of some control bit in I. Also the switch has the capability to arbitrate between two conflicting inputs. The network consists in construction of demultiplexer trees (also called *fan-in* trees) with every memory module as its root and the processors as leaves. Every new tree may use some of the already existing links. The path between a processor and a memory module is digit controlled such that a cross- bar module connects an input to one of its b outputs depending on a single base-b digit taken from the destination address. Patel computed the total number of cross-bar modules (of size $a \times b$) required in an $a^n \times b^n$ delta network as $a^n - b^n/a - b$ if $a \neq b$, or nb^{n-1} if $a = b$. Patel carried out analysis of bandwidth and memory request acceptance probability of a processor, in case of a crossbar system and a delta network of the same dimension. Assuming a processor request generating probability of m and that the blocked requests are

ignored, Patel[3] came up with a closed form expression for the expected bandwidth, BW in a cross-bar of size $M \times N$

$$BW = N\left[1 - N\left(1 - \frac{m}{N}\right)^M\right]$$

where M = number of processors,

N = number of memory modules.

A simple explanation for the model is that $(1 - \frac{m}{N})^M$ represents the probability that none of the M processors requests for a particular memory module in a particular memory cycle. So probability that there is at least one memory request for that memory module is obtained by subtracting this from 1. Multiplying by the number of memory modules, N, gives the expected number of busy memory modules per cycle, assuming the memory requests are uniformly distributed over all the memory modules in a cycle.

Let the probability of acceptance of a memory request P_A,
be defined as the ratio of the expected bandwidth to the expected number of requests generated per cycle. Then

$$P_A = \frac{BW}{mM} = \frac{N}{mM} - \frac{N}{mM}\left(1 - \frac{m}{N}\right)^M$$

Applying the previous result for a $M \times N$ crossbar to any $a \times b$ crossbar module of the Delta network, the expected number of requests that it passes per cycle is obtained by setting $M = a$ and $N = b$. Thus $BW = b - b\left(1 - \frac{m}{b}\right)^a$ Dividing the above expression by the number of output lines in any of the $a \times b$ modules yields the rate of requests on any of the b output lines as $1 - \left(1 - \frac{m}{b}\right)^a$ If m_i is defined as the request rate on an output line in stage i , then the BW of the Delta network is given as $BW = b^n m_n$ where $m_i = 1 - \left(1 - \frac{m_{i-1}}{b}\right)^a$ and $m_0 = m$
Thus

$$P_A = \frac{b^n m_n}{a_n m}$$

Since Patel could not find a closed form solution for the BW of a Delta network, he could only graphically compare the two parameters BW and P_A of Crossbar and Delta networks for various sizes. Kruskal and Snir [27] provided asymptotic solutions for Patel's recurrence relation for the BW of delta network. Kumar and Jump [9] showed that the approximate solution obtained by Kruskal and Snir for the throughput of unbuffered delta networks is a strict upper bound. They also derived a strict lower bound on the performance of the delta network. If the output rate of the switches in stage i is denoted by x_i then the following upper and lower bounds for x_i is obtained.

$$\frac{2B}{\frac{2B}{x_0}+(B-1)i} \geq x_i \geq \frac{2B}{\frac{2B}{x_0}+(B-1)i+2BE_i}$$

where, $E_i = \frac{B^2-B+2}{4B^2} \times log_e \left[\frac{i \times (B-1)}{\frac{2B}{x_0}-2B+2} + 1 \right]$.

Streker [6] developing a set of simple approximate models for a system with N processors and M memory modules had arrived at the same formula for BW. The basic assumption remained the same - memory requests at every cycle are independent and uniformly distributed. Another way to look at it is that the rejected requests at the end of every memory cycle are discarded, and are reassigned randomly among all memory modules in the next cycle. Ravi [28] presented a very similar model for the case where the request rate is 1, and his results has been shown to be exactly the same. Briggs and Davidson [29] have extended Streker's model to incorporate cases like memory cycle times greater than unity, memory access times different from memory cycle times etc.

Non-uniform reference model

For non-uniform reference, the basic assumptions are changed. Now it is assumed that the rejected requests are not simply discarded but are resubmitted. Also in some cases memory requests were assumed to be non-uniform in each memory cycle. Yen, Patel and Davidson [7] proposed a new model, called the *rate adjusted probability* model, which sought to adjust the memory request rate (assumed same for all the processors)

for the new assumption that the rejected requests are not simply discarded but are resubmitted as new and independent requests, thereby increasing the effective request rate. The resubmission follows a uniform distribution. One of the major drawbacks of this model is that it over-estimates BW, because multiple rejected requests to the same memory module will get evenly distributed to all the memory modules in the next cycle, thereby increasing their chance of acceptance. If P_A is the probability that a request is accepted then $\frac{1}{P_A}$ is the expected number of rejections (blocked cycles) plus the one accept cycle. If ψ is the static request rate and α is the adjusted request rate (also called *dynamic* request rate) then α is given by

$$\alpha = \frac{\frac{\psi}{P_A}}{1-\psi+\frac{\psi}{P_A}}$$

and bandwidth is given as

$$BW = M\left[1-\left(1-\frac{\alpha}{M}\right)^N\right]$$

The probability of acceptance P_A is $\frac{BW}{N\alpha}$.

Bhuyan [8] introduced the concept of *favorite memory* modifying the *uniform distribution* pattern of memory requests assumed in the earlier works. Each processor P_i has a *favorite* memory module M_i with which it communicates more than any other memory module. A factor m which is the probability that P_i accesses M_i provided P_i generates a memory request is assumed to be known *apriori* . The remaining processors' requests are uniformly distributed among the $(N-1)$ remaining modules. Obviously $m > \frac{1}{N}$. Bhuyan carried detailed analysis of favorite memory cases for crossbar and Delta network. If p_0 is the probability that a processor generates a request, then $p_0 m$ is the rate of request to its favourite memory. Therefore, the rate of request by P_i to the non-favorite memories is $p_0(1-m)$, which is *uniformly distributed* among the $(N-1)$ memory modules. Also probability that P_i does not request M_i is $1-p_0 m$. Hence for an $N \times N$ crossbar, the probability that M_i will not be requested by the $(N-1)$ processors (excluding P_i) is $\left(1-p_0\frac{(1-m)}{(N-1)}\right)^{N-1}$
Hence probability that none of the N processors requests memory module M_i is

$$(1 - p_0 m)\left(1 - p_0 \tfrac{(1-m)}{(N-1)}\right)^{N-1}$$

The probability that there will be at least one request for memory module M_i is

$$1 - (1 - p_0 m)\left(1 - p_0 \tfrac{(1-m)}{(N-1)}\right)^{N-1}$$

Hence the bandwidth is obtained as

$$BW = N\left[1 - (1 - p_0 m)\left(1 - p_0 \tfrac{(1-m)}{(N-1)}\right)^{N-1}\right]$$

Bhuyan extended the above analysis to favourite memory cases for $M \times N$ crossbars with $M > N$ and $M < N$:

For the case $M > N$, $BW = N\left\{1 - (1 - p_0 m)\left(1 - p_0 \tfrac{1-m}{N-1}\right)^{N-1}\left(1 - \tfrac{p_0}{N}\right)^{M-N}\right\}$

For the case $M < N$, $BW = N - M(1 - p_0 m)\left(1 - p_0 \tfrac{1-m}{N-1}\right)^{M-1} - (N - M)\left(1 - p_0 \tfrac{1-m}{N-1}\right)^{M}$

For Delta networks the rate of request on an output line in the ith stage is

$$p_i = 1 - \left(1 - \tfrac{p_{i-1}}{b}\right)^{a}$$

where p_0 is the probability of generation of a request by a processor. The analysis is restricted to only $N \times N$ case of delta network with n stages of $a \times a$ crossbar modules with N/a such modules per stage. A processor is connected to its favourite memory module when all switches are in a straight connection. If p_0 denote the request rate of a proces sor and q_i denote the probability that there is a favourite request to a switch at the $(i+1)th$ stage, then $p_i = 1 - (1 - p_{i-1}q_{i-1})\left(1 - p_{i-1}\tfrac{1-q_{i-1}}{a-1}\right)^{a-1}$. The rate of request at the final stage p_n decides the bandwidth. Hence, $BW_f = p_n \times N$.

2.1.2. Buffered MINs

With buffered MINs we now have the provision of saving rejected request-packets for resubmission to the same memory module in the next cycle. This increases the system performance considerably, especially when the traffic pattern is non-uniform. A packet is not lost when path conflict occurs with more than one processor trying to send their requests through the same port of a switching module. Also a packet can

leave a buffer in one switching module only when the buffer at the succeeding stage is able to accept it. Next we consider several performance evaluation of buffered MINs with the help of various models.

Skinner and Asher [5] were the first to use Markov chain models to analyse memory interference in multiprocessor systems with memory queues. The basic assumption was that all processors and all the memory modules are identical. A multiprocessor system with n processors and m memory modules, where, at each memory cycle a set of processors are allocated a set of memory modules is similar to an occupancy problem with n balls and m urns, a common combinatorial problem. If state of the Markov chain is defined by the m-tuple $(k_1, k_2, k_3, \ldots, k_m)$, where $\sum_i k_i = n$ and k_i denotes the number of processors waiting in the queue of the i^{th} memory module including the currently active processor. The number of possible states in such case is equal to the number of different ways n balls can be placed in m urns which is given by $\binom{n+m-1}{n}$. Since the number of states becomes prohibitively large for large values of n and m, Skinner and Asher's study was limited to a small number of processors and memory modules, and they failed to generalise it for larger number of processors and memory modules.

Bhandarkar [1] reduced the number of states noting that many of the states of the Markov chain model were identical owing to the assumption that all the processors are identical. For example in a four processor and three memory modules system, the states denoted by the following 3 tuples are identical : $(1,2,1),(1,1,2),(2,1,1)$. Bhandarkar defined a *partial* state as a transition state between the current state and the next state, with the assumption that all the currently active processors will terminate their memory access at the end of the current cycle. So st the end of the current cycle the system enters a partial state where the number of processors in each memory queue decreases by one. If the current state is $(k_1, k_2 \ldots k_m)$ then the partial state at the end of the cycle is $(j_1, j_2, \ldots j_m)$ where $j_i = k_i - 1$ if $k_i > 0$ or $= 0$, otherwise. A new state $(l_1, l_2, \ldots, l_m)$ is said to be reachable from $(k_1, k_2, \ldots, k_m)$ if

$l_i \geq k_i$. Probability of a state transition is $\frac{x!}{d_1!d_2!...d_m!}\left(\frac{1}{m}\right)^x$ where $d_i = l_i - j_i$ and $\sum_i d_i = x$. Since the number of states in this case too, became unmanageably large for $m, n > 3$, Bhandarkar wrote a program to compute the state transition matrix of the Markov chain.

Sethi and Deo [2] proposed a discrete Markov chain model for analysing memory interference in multiprocessor systems for non-uniform memory access pattern. They assumed that if the n^{th} request of a processor is for memory module i, then the $(n+1)^{th}$ request will be for module i with probablity α and for module $j(j \neq i)$ with probability $\frac{(1-\alpha)}{(m-1)}$. Thus requests to memory modules except the previous one are assumed to be uniformly distributed. If $\alpha = \frac{1}{m}$, where m is the number of memory modules, the model reduces to the uniform reference model. The performance measure used by Sethi and Deo for analysis of a $p \times m$ system (p processors and m memory modules) is Average Number of Busy Memory Modules (ANBM's). This is the average number of memory modules which are busy during a memory cycle. Following Bhandarkar's approach to use the discrete Markov chain technique, the ANBM's for a $2 \times m$ and $p \times 2$ are computed from the state transition probabilities. The ANBM for a $2 \times m$ system is given by

$$ANBM = \frac{m(2m+\alpha-3)}{m(m+\alpha-1)-1}$$

The ANBM for a $p \times 2$ system having two memory modules and $p \geq 2$ processors is given by

$$ANBM = \frac{2(p+\alpha-1)}{p+2\alpha-1}$$

Similar analysis was carried out for a $3 \times m$ system with three procesors and $m \geq 3$ memory modules, but with $\alpha = 1$. The ANBM was obtained as $3 - \frac{6}{m+2}$.

Yen, Patel and Davidson[7] proposed a *steady state flow model* for multi- processor systems with memory queues. Let N and M be the number of processors and memory modules, ψ be the request rate and f be the processor utilization. Then Nf processors are active (either in accessing memory or in some internal computation) and $N(1-f)$

processors are blocked. The Nf active processors issue $Nf\psi$ memory requests each cycle and $Nf(1-\psi)$ processors are blocked. Since all the $Nf\psi$ requests are accepted, the steady state number of active processors remains fixed at Nf. To compute the number (r) of non-empty queues in the system, a successive approximation method is applied. In the first step all the $N(1-f)$ queued requests are distributed uniformly among M queues to give

$$\frac{r'}{M} = 1 - \left(1 - \frac{1-f}{M}\right)^N$$

This model uses a better estimate assuming that each of the M queues has probability $\frac{r'}{M}$ of making an access request. Then

$$\frac{r}{M} = 1 - \left(1 - \frac{\frac{r'}{M}}{M}\right)^M$$

is the estimated probability that there is a request for a particular memory. The probability that none of the N processors has a request for that memory is $\left(1 - \frac{f\psi}{M}\right)^N$ Thus the effective memory bandwidth of the system is given as

$$BW = M\left[1 - \left(1 - \frac{f\psi}{M}\right)^N \left(1 - \frac{r}{M}\right)\right] = Nf\psi$$

Pfister and Norton [16] introduced a separate class of memory-access non-uniformity called *hot spot* of higher access rate superimposed on otherwise uniform traffic. Hot spots capture the effect of all the processors continually sharing a common variable, which resides in a particular memory module called the *hot* memory. An interesting phenomenon called *tree saturation* can be observed for MINs with finite buffers. When the queue at the root of the *fan-in* tree (formed by the links connecting all the processors to a single memory module in the MIN) becomes full, the queues in the previous stage can no longer send packets to them and become full themselves leading to similar conditions to all the queues in other levels of the tree. Eventually the entire tree consists of full queues. Pfister and Norton introduced the concept of *combining* messages to similar destination at every node of the network. In particular comparisons are performed only between a pair of messages stored in the output queues

of each switching node (only blocked messages), and if the messages are combined the information is stored in a *wait buffer.* A combined message can combine again in a later node. When a reply to a combined message reaches a node where it was combined, multiplies replies are generated and the messages are routed to individual destinations based on the information stored in the *wait buffer.*

Pairwise combining shows definite improvement over non-combining in systems having hot spots. Ideally one would like to combine as many combinable messages as resided in the buffer of a switch module. But this makes the combining process complicated and also creates congestion in the wait buffer. Simulating pairwise combining in a network consisting of many stages, Lee, Kruskal and Kuck [17] observed that *tree saturation* still takes place. With infinite buffer length the congestion takes place only at the root, although the situation improved with finite buffers because messages spend more time waiting near the base of the tree thereby increasing the probability of combining. Lee and Kuck suggested a bounded combining scheme with three-way combining as the best solution noting that combinable requests coming out of the switches are slightly more than two combinables.

Jenq [10] formulated two models for analyzing single-buffered banyan network composed of 2×2 switching elements. The basic assumption behind the first model is that the buffers in the same switching element (SE) are independent of one another and traffic pattern is uniform. He also developed another model that does not include the independence assumption, and the results from them nearly matched. So it was concluded that the independence assumption is realistic. However a major drawback of this model is that blocked states were not considered.

Yoon, Lee and Liu [12] extended Jenq's model to the more general case of networks composed of $n \times m$ SEs. They also expanded the single-buffer model to multi-buffer model. Basically, this model has two states - one state in which there is no request in the buffer and another state in which there is a single request in. Since all the buffers are assumed to be identical and independent, these two states are sufficient

to model the system behavior. But this model still did not incorporate the blocked state. Theimer, Rathgeb and Huber [14] improved upon the previous two models by discarding the assumption of independence between buffers in a SE. Instead they exhaustively enumerated the possible states in the 2×2 SE. The model has nine states and it takes the blocked state into consideration. Although it shows significant improvement over the previous two models, the number of states become prohibitively large for arbitrary larger sizes of SE or number of buffers.

Although the performance of MINs under uniform traffic model is interesting to note, it does not represent the traffic pattern of a realistic system. Kim and Leon-Garcia [18] carried out the analysis of buffered Banyan networks, consisting of 2×2 switching elements (SE, crossbar elements), under non-uniform traffic pattern. The two important performance metrices of any interconnection network are throughput and delay. *Throughput* is define d as the number of output packets per clock cycle that exit from each output port. *Delay* is defined as the time taken by a packet to reach the destination port starting at the input port. For non-uniform traffic pattern these parameters are going to change from module to module and hence the maximum delay and the maximum throughput at a destination port which has the worst congestion is considered. The buffered Banyan network is modeled as a Markov chain and analysed using a network decomposition technique. First, individual SE's are modelled and then the relationship among the switching elements are depicted. In fact, each SE is represented by the state of its buffers, which are as before. Buffers with single elements are considered.

State 0 : no packet in the buffer

State 1 : single packet in buffer

For our case, the throughput of the network is defined as the expected number of packets delivered from the last stage of the network per cycle. For a switching module l in stage k, let the input ports be denoted by x and the output ports by y. Let,

B_{klx} Probability that there is a packet in buffer of port klx

P_{klxy} Probability that a packet at klx is destined to y

r_{klx} Probability that a packet at klx can advance to next stage, then

the average number of packets delivered per cycle from the port *klx* to *kly* of a SE is $P_{klxy}B_{klx}r_{klx}$.

To find B_{klx} the state-transition equations of the two state Markov chain is solved to get the steady state probabilities of states 0 and 1. B_{klx} is equivalent to the steady state probability of state 1. To find r_{klx} note that a packet at the buffer of the *klx* port is allowed to advance if the following conditions are met :

1) The packet either reaches port y without any contention or wins contention if there is any. 2) Buffer at $(k+1)l'y'$ is able to accept a packet. This is possible if there is no packet at that buffer, or if there is one it is going to leave the port in the current cycle.

If both 1) and 2) are satisfies then a packet reaches the next stage and contributes to the throughput of the network. Based on the above the following equation was derived

$$r_{klxy} = (1 - B_{klx} + B_{klx}P_{klxy} + .5B_{klx}P_{klxy}) \cdot \left(1 - B_{(k+1)l'y'} + B_{(k+1)l'y'}r_{(k+1)l'y'}\right)$$

r_{klx} can be obtained from r_{klxy} as follows

$$r_{klx} = r_{klxy}P_{klxy} + r_{klxy}P_{klxy}$$

and delay for a path is given as

$$\sum_{allstages} \frac{1}{r_{klxy}}$$

Addding to this the IBC buffer delay, we get the total delay of the network. The objective is to compute the values of r and B. Since both of them are described by recurrence relations, the solutions are by iterative method, assuming a certain *load matrix* feeding certain traffic load to the first stage.

Hsiao and Chen [13] modeled single buffered MINs with SEs of arbitrary sizes using three states. 'State 0' is when the buffer is empty, 'State n' is the normal state

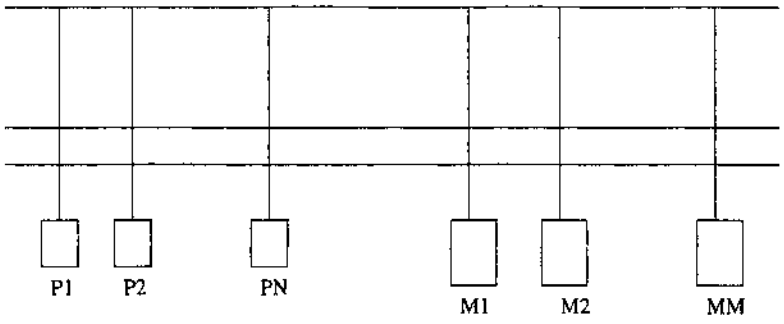
where a buffer has a packet which arrived in the previous network cycle, and 'State b' is the blocked state where the buffer has a blocked packet which had stayed there for at least one cycle due to blocking. A blocked state occurs when a packet cannot move to the next stage because the destination stage buffer is full or it has lost the contention with other packets for a same output link. Hsiao and Chen, however based their model on certain unrealistic assumptions. For example, they assumed that a blocked request cannot go directly to the next stage even though there is an empty buffer and no other packets are destined for it. This is because a blocked packet can only contend with other blocked packets and if it wins the contention it changes state from blocked to normal state. Thus this assumption overestimates packet delay of the network. Also a blocked packet is not resubmitted to the original memory module but randomly distributed.

2.2. Multiple Bus Interconnection Network

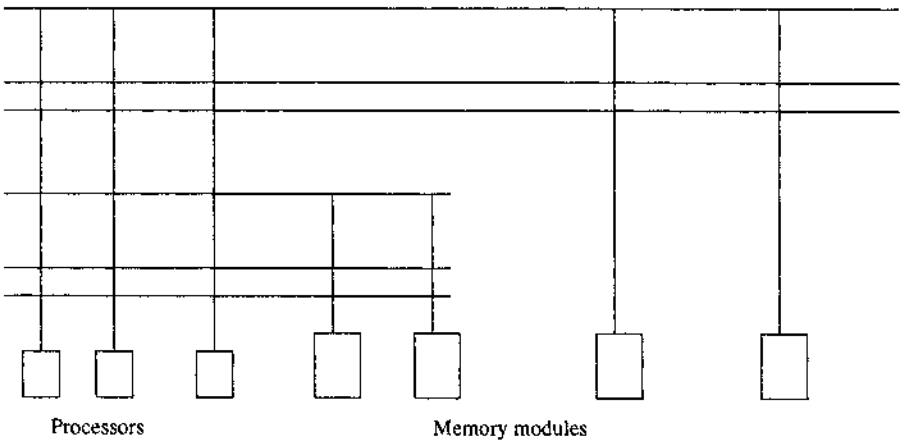
The interconnection networks discussed so far used either crossbar networks or multistage interconnection networks, which are attractive because of the high bandwidth capability and fault tolerence. But the principle disadvantage of these systems is their high cost (cost of the switching modules). Most of the commercially available systems having a single processor use a single shared bus. A logical and cost effective extension of the single bus systems are the multiple bus systems. Single bus systems are inexpensive but they lack fault tolerence and have a limited bandwidth. A multiple bus system will naturally increase the bandwidth and fault tolerence, without increasing the cost appreciatively, because we are not dealing with the expensive switching modules anyway. Another advantage of multiple bus systems over MIN's is the simple interconeection scheme and easy expansibility.

Lang [19] was the first to investigate the performance of multiple-bus systems. They divided the multiple bus systems into two categories, complete and partial, based on the interconnection scheme of the bus with the processors and memory

modules. For a typical system consisting of B buses used to connect N processors to M memory modules ($B \leq N$), in the *complete* case (fig 2.2a) every processor and memory module are connected to every bus; in the *partial* case (fig 2.2b), each memory module need to be connected to a subset of buses. Lang *et al.* showed that a complete multiple bus configuration with $B \approx \frac{N}{2}$ has almost the same bandwidth as an $N \times M$ crossbar and higher fault tolerence. Simulations were used to determine the bandwidth of both classes of system.



a) Complete Bus case



b) Partial Bus case

Figure 2.2: Examples of Multiple Bus Interconnection Networks

Mudge, Hayes, Buzzard and Winsor presented a discrete stochastic model of bandwidth for multiple bus multiprocessor systems for both the partial and complete case taking into account the various conflicts arising from memory and bus contention. Two types of assumptions are made to simplify the analysis - *i) temporal independence*, in which successive memory requests are assumed to be independent (i.e. blocked requests are discarded), and *ii) spatial independence*, in which the event that there is at least one request for a particular memory module is independent of the same event for another memory module. Later, spatial independence assumption was removed, and an iterative scheme is developed to reduce the error caused by the temporal independence.

There are two sources of conflict due to memory requests in a multiple bus system - *i) memory interference*, when a processor requests an already busy memory module, *ii) bus interference*, when a memory request cannot be completed due to non-availability of a bus. Lang *et al.* proposed a two stage arbitration scheme to resolve these conflicts. In the first step, M 1-out-of-N memory arbiters each selects one outstanding request for each memory module. In the second step, one B-out-of-M bus arbiter assigns buses to the requests selected in the first stage. Each processor is assumed to generate independent requests (Bernoulli trials) for memory modules at the beginning of each memory cycle with probability p. The memory requests are assumed to be uniformly distributed over all memory modules with probability $\frac{1}{M}$. Therefore, probability that a particular processor P_i requests memory module M_j is $\frac{p}{M}$. If E_j is the event that there is at least one request for M_j, then probability of E_j is

$$Pr[E_j] = q = 1 - (1 - \tfrac{p}{M})^N$$

If events E_j are assumed to be independent, then probability that exactly i of the M memory request arbiters output a memory request is $f(i) = \binom{M}{i} q^i (1-q)^{M-i}$. The probability that B or more of the M memory request arbiters output a memory request is $F(B) = \sum_{i=B}^{M} f(i)$. Now requests for at most B of these memory requests can be met as there are only B buses. Hence the BW of a *complete* multiple bus

system is

$$BW = BF(B) + \sum_{i=1}^{B-1} i f(i)$$

These results are next extended to the partial bus case. Let the B buses be divided into g equal groups (assume g is a factor of B). Let $m = \frac{M}{g}$ and $b = \frac{B}{g}$. Thus, $f_g(i) = \binom{m}{i} q^i (1-q)^{m-i}$, and $F_g(B) = \sum_{i=b}^{m} f_g(i)$. Hence bandwidth is given as

$$BW_g = g\left[bF_g(B) + \sum_{i=1}^{B-1} i f_g(i)\right].$$

The idea is to multiply by g the bandwidth of any one of the groups of complete multiple bus systems to get the bandwidth of the whole system.

The same model is extended by the authors to the case with spatial dependence, i.e. assuming each event E_j is dependent on E_k for $k \neq j$. The dependence is given as follows.

$Pr[E_j|E_k] = \sum_{i=1}^{N} Pr[E_j|E_k$ results from i requests$] \times Pr[E_k$ results from i requests$]$
$= \sum_{i=1}^{N} \left[1 - \left(1 - \frac{p}{M-1}\right)^{N-i}\right] \binom{N}{i} \left(\frac{p}{M}\right)^i \left(1 - \frac{p}{M}\right)^{N-i}$.

To consider the effects of *spatial* dependence, the authors defined a new function $h(i)$ denoting the exact probability that i of the M memory request arbiters output a memory request. An expression for $h(i)$ was deduced, and is given as

$$h(i) = \sum_{k=i}^{N} \binom{N}{k} p^k (1-p)^{N-k} \binom{M}{i} M^{-k} \theta$$

where θ is given as, $\theta = \sum \frac{k!}{n_1! n_2! \ldots n_i!}$ where the summation is carried out over all $n_1, \ldots n_i > 0$. An alternative expression for θ is given in [] as $\theta = i! \left\{ {k \atop i} \right\}$, where $\left\{ {k \atop i} \right\}$ denotes a Stirling number of the second kind. It is defined as the number of ways to partition a set X_k of k elements inti i nonempty disjoint subsets. The new expression for bandwidth is given by

$$BW = BH(B) + \sum_{i=1}^{B-1} i h(i), \text{ where } H(B) = \sum_{i=B}^{M} h(i).$$

In order to distinguish between the two expressions of bandwidth, they are called BW^f and BW^h respectively. It is however shown that in the bus-sufficient case (i.e.

where $B \geq M$) we have $BW^h = BW^f$.

Until now it is assumed that blocked requests are rejected, which in practise is not true. Actually they are stored in memory queue and resubmitted to the same memory module in the next cycle. The probability of acceptance of a memory request is given as $P_a = \frac{BW}{Np}$. Due to the resubmission of memory requests to the same memory module, teh request rate is going to increase. Let α be the *adjusted* request rate. Ratio of the number of successful memory requests to teh total number of requests is the probability of acceptance, P_a, given by $\frac{1-1}{\frac{\alpha}{p}-1}$. The above two equations for P_a can be used in an iterative scheme to get an improved estimate of bandwidth due to the new request rate α as follows

$$\alpha_{k+1}^{-1} = 1 + \frac{BW(\alpha_k)}{Np^2}(1-p).$$

where $BW(\alpha_k)$ is obtained by replacing p by α in the expressions for $f(i)$ and $h(i)$, used in the expressions for BW. Finally, some asymptotic approximations to the bandwidth of multiple bus systems is considered. From our expression of bandwidth, BW^f is bounded by $\sum_{i=0}^{M} Bf(i)$, i.e. $BW^f \leq B$. Also replacing the first term on the right hand side by $\sum_{i=B}^{M} if(i)$,we get $BW^f \leq \sum_{i=0}^{M} if(i)$, i.e. $BW^f \leq Mq$. Hence, $BW = min(B, Mq)$. Mq is called bus-sufficient bandwidth and making $B > Mq$ does not improve the bandwidth anymore.

Markov chain models for the analysis of multiple bus multiprocessors were developed by Marson and Gerla [22]. The performance index was the average number of active processors called *processing power, P*. The model specifies that a processor can be in any of the following three states:

1) processor is busy in internal computation using its private cache memory.

2) processor exchanges data with other cooperating processors, by accessing the global memory modules. 3) processor is waiting in the queue of any of the global memory modules.

A processor is said to be *active* when it is in state 1) and the processing power is

defined as:

$$P = E[\text{number of active processors}].$$

Processors join memory queues if the requesting memory module is busy, and before proceeding to service (i.e. accessing memory) must be given a permit to access a bus. The permit is returned upon completion of service. With these assumptions a closed queueing network model is developed with the following parameters : 1) the duration of the access to the common memory is an independent exponentially distributed random variable with mean $\frac{1}{\mu_j}$ for the j^{th} memory module. In order to simplify the model the parameter is assumed to be the same for all memory modules and simply called $\frac{1}{\mu}$. 2) The interval between subsequent access requests is an independent, exponentially distributed random variable with mean $\frac{1}{\lambda_i}$ for the i^{th} processor. This was also assumed to be same for all the processors and simply called $fracl\lambda$. 3) A uniform reference model was assumed, that a request from a processor is directed to any of the common memory modules with probability $fracl m$. Marsan and Gerla carried out the exact performance analysis for a $p \times m \times b$ system with the above assumptions. Applying Little's law to the entire memory system including queues and servers, we have the average customer delay $D = \frac{p-P}{P\lambda}$. The average queueing time is given as $W = \frac{p-P(1+\rho}{P\lambda}$, where $\rho = \frac{\lambda}{\mu}$. The average number of queued processors is $N_q = WP\lambda = p - P(1+\rho)$; and average number of processors accessing common memory is $N_s = \frac{D-W}{P\lambda} = P\rho$. The average cycle time (average time taken by the customers to cycle through the closed queueing network) is easily obtained as $C = W + \frac{1}{\lambda} + \frac{1}{\mu} = \frac{p}{P\lambda}$.

Marsan and Gerla also constructed a Markov chain model using the theory of lumpable Markov chains for defining the states. The state definition is $(n_m, q_1, q_2, \ldots, q_m)$ for the genral $p \times m \times b$ system. n_m is the number of processors currently accessing a common memory, $q_1 \ldots q_b$ are numbers of processors queueing for memory modules currently accessed and $q_{b+1} \ldots q_m$ are numbers of processors queueing for free memory not accessible because no free bus is available. The general $p \times m \times b$ case was

not easy to handle because of teh huge explosion of states as the number of processors and memory modules are increased. The state transition rates for a $3 \times 3 \times 2$ model could be computed. Next, various approximate models were developed with reduced number of states, by reducing the amount of information about the status of the queues in each state. In the first model, the state of the system is represented by (n_m, n_q) where n_m =number of processors currently accessing a common memory module, and n_q =number of processors queued. The transition rates were computed for the most general case. Since the number of active processors is $p - n_m - n_q$, the processing power can be evaluated once the steady state distribution of the Markov chain is known. In the second model the state definition is the same as in the first model. The transition rates are evaluated using an "averaging" technique. Various identical states are merged into a "macro-state", the rates from the merged state to all neighbouring state are added up and the new transition rate for the merged state is defined as the ratio between the sum and the number of merged states. In the third model, a state is represented by (n_m, n_q, f), where the definitions of n_m and n_q remains as before and f is a flag which is set to zero when no processor is queued for a bus. Transition rates for the $p \times m \times 2$ were computed for this model. In the fourth model, the system state is simply the number of active processors. The transition rates are evaluated using the averaging technique.

A semi Markov model for the performance of multiple bus systems was proposed by Mudge and Al-Saldoun [21]. The basic assumptions are as follows: At any cycle, a processing element (PE) can be in any of the three states, *thinking* i.e. busy in unternal compute, *accessing* i.e. connected to a memory module, and *waiting* i.e. waiting in the queue of a memory module. A memory module can also be in any of the following two states: *busy*, when a PE is connected to it, and *idle*, when no PE is connected to it. Each PE will submit a request after its thinking time T, a discrete random variable, which is the inter-request time. Also there is a connection time between a processor and a memory module, given by the discrete independent

random variable C. Considering a $N \times M \times B$ system, a semi-Markov process was used to describe the state of each PE. This considerably simplifies the state space reducing it to only four states as opposed to using a discrete Markov chain model for each PE, where the number of states can grow very large. Briefly, a semi-Markov process (SMP) is a stochastic process which can be in any one of the K states $1, 2, \ldots, K$. Each time it enters a state i it remains there for a definite amount of time called the *sojourn time*, having mean η_i and then makes a transition to state j with probability p_{ij}. The four states are as follows: The first state is the thinking state 0. The second state is the accessing state 1. The third state is the full waiting state 2. The PE enters state 2 when it requests an idle MM simultaneously with at least one other PE, and it fails to access the MM (by passing the two levels of arbitration as proposed by Lang). The fourth state is the residual waiting state 3. A PE enters state 3 when it requests a busy MM or when, due to bus contention, access is blocked to an MM even though it is idle. The underlying approximation of this model is in describing any PE behaviour independent from other PE's while compensating for the coupling between the PE's in the transition probabilities between the states of the SMP.

Detailed analysis was carried out and the following performance indices were computed for the $N \times M \times B$ multiprocessor system. The transition probabilities between the states were computed and the steady state probabilities were determined. Let α_i be the probability of the process entering state i and P_i be the steady state probability of the same state. The following parameters were computed:

Bandwidth, $BW = NP_1$

Processor utilization, $PU = P_0 + P_1$

Memory utilization, $MU = \frac{N}{M} P_1$

Bus utilization, $BU = \frac{N}{B} P_1$

Average queue length, $L = \frac{N}{M}(P_2 + P_3)$

Average waiting time, $W = \frac{\eta_2 \alpha_2 + \eta_3 \alpha_3}{\alpha_1}$.

The performance of multiple bus multiprocessor systems under a non- uniform reference model called hierarchical requesting model is studied by Chen and Sheu [25]. Various types of multiple bus networks investigated involve i)complete(full) bus case ii)partial bus case iii)single bus case and the newly introduced iv)partial bus with K classes. In a single bus network, each processor is connected to all the buses, but each memory module is connected to a single bus. A new architecture of $N \times M \times B$ multiple bus networks called partial bus networks with K classes is introduced, where there are K classes of memory modules ($K \leq B$). The memory modules of class C_K are connected to all the buses 1 to B, memory modules of class C_{K-1} are connected to $B-1$ buses from 1 to bus $B-1$ etc. In general, memory modules in class C_j are connected to $j+B-K$ buses. Fig 4 shows an example of a partial bus network with 3 classes. Memory modules which needs more fault tolerence of buses and which are more frequently referenced are connected to more number of buses in this scheme. To avoid memory and bus contention the two stage arbitration scheme proposed by Lang [19] is used. The cost and fault tolerent capability of each type of network are

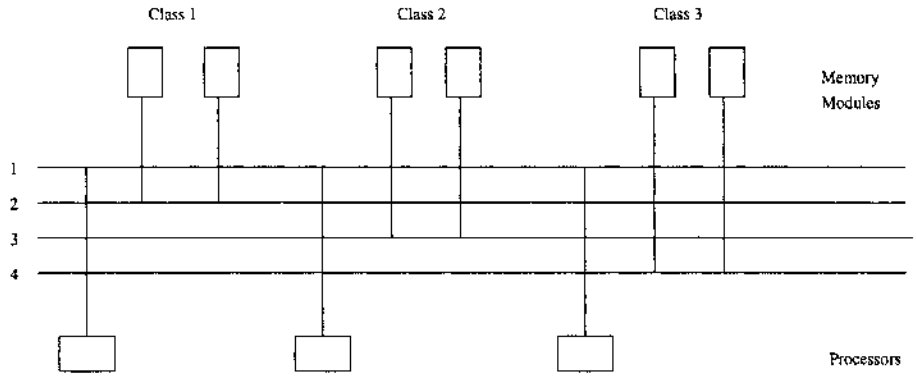


Figure 2.3: Example of Partial Bus Network with three classes

evaluated and compared. The cost of a multiple bus network is proportional to the number of connections in the network. For the $N \times M \times B$ multiple bus with full bus-memory connection, cost is $O(B(N+M))$, and the degree of fault tolerence is $B-1$. Similarly cost for single bus systems is $O(BN+M)$, and it is less fault tolerent than the complete bus case. The cost in case of the partial bus case is $O(N+\frac{M}{g})$, where g is the number of groups; the fault tolerence is $\frac{B}{g}-1$. In case of partial bus network with K classes, cost is proportional to $BN+\sum_{j=1}^{K} M_j(j+B-K)$, where M_j is the number of memory modules in class C_j. The degree of fault tolerence of this network is $B-K$. It is seen that the cost and fault-tolerence of networks with partial bus-memory connection scheme are intermediate between the networks with full and single bus-memory connection.

To analyse the performance of multiple bus networks a hierarchical requesting model is proposed. For an $N \times N \times B$ system, assume that $N = k_1 k_2 \ldots k_n$. Each processor P_i has a memory module MM_i as its favourite memory. These processors and memories are organised into an $n-$level hierarchy. First, the N pairs of processors and memory modules are partitioned into k_1 clusters in the first level, each cluster containing $k_2 k_3 \ldots k_n$ pairs of processor- memory. In the second level, each of k_1 clusters is partitioned into k_2 subclusters of equal size, and so on. For an $n-$level hierarchy, there are $n+1$ different request rates for a processor, $m_0, m_1 \ldots m_n$ depending on the hierarchy of the memory module addressed to. We assume that $m_0 \geq m_1 \geq \ldots m_n$. If N_i be the number of processors or memory modules belonging to the same subcluster in the $(n-i)th$ level excluding those in the $(n-i+1)th$ level, then N_i is given as $N_i = (k_{n-i+1}-1)k_{n-i+2} \ldots k_{n-1} k_n$ and $\sum_{i=0}^{n} m_i N_i = 1$ where $N_0 = 1$. Let X be the probability that there is at least one request for a particular memory module MM_j and let p_0 be the probability of connection to a favourite memory. If r be the request generation rate then the probability that at least one request is generated by those processors which request connection to MM_j with fraction m_i is $p_i = 1-(1-rm_i)^{N_i}$. Hence, probability that at least one processor requesting

connection to MM_j is $X = 1 - (1 - rm_0)(1 - rm_1)^{N_1} \ldots (1 - rm_n)^{N_n}$. Then the probability that exactly i of the N memory-request arbiters output a memory request is given by $P(i) = \binom{N}{i} X^i (1 - X)^{N-i}$. The memory bandwidth in case of full bus-memory connection is given by

$$MBW = NX - \sum_{i=B+1}^{N} (i - B) P(i).$$

For the single bus case, if M_i be the number of memory modules connected to bus i, then the probability that there is at least one memory servicein bus i is $Y_i = 1 - (1 - X)M_i$. Then the memory bandwidth is given by

$$MBW = \sum_{i=1}^{B} Y_i.$$

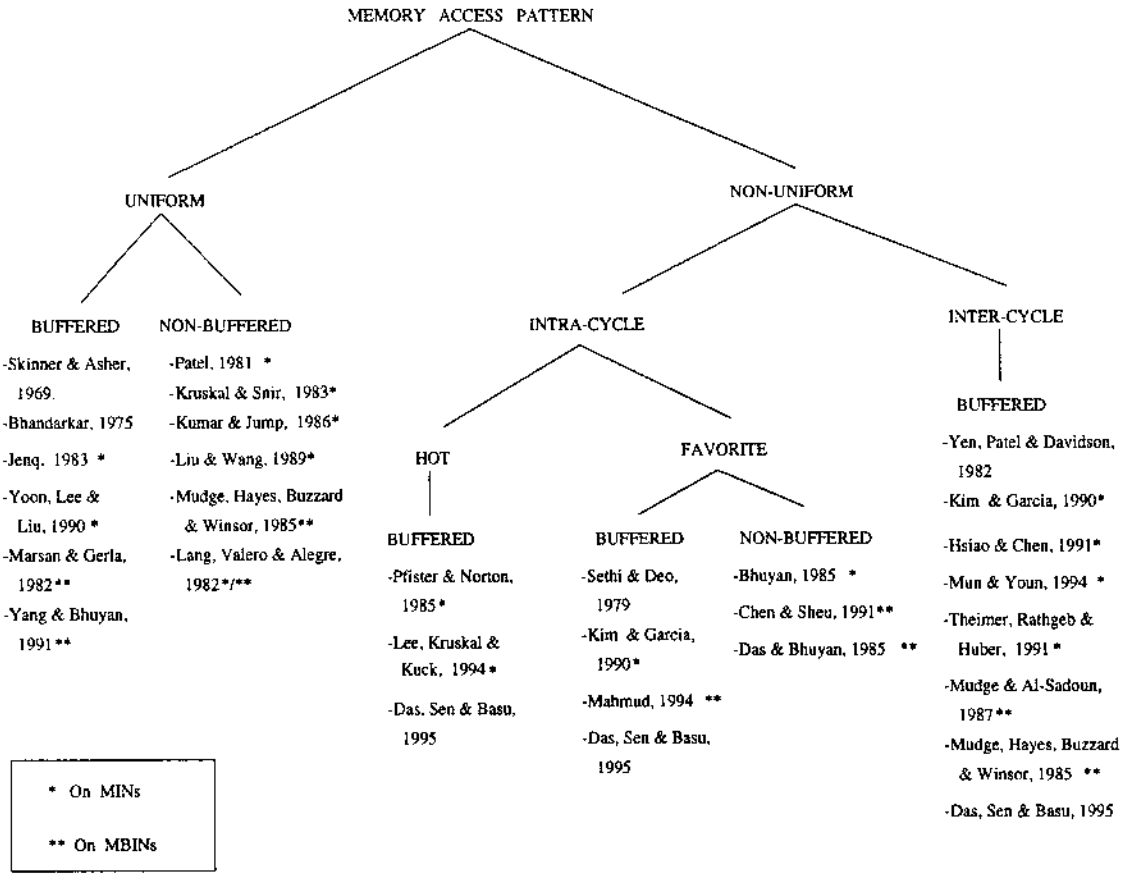


Figure 2.4: A taxonomy of related work on memory references

2.3. Conclusion

In this chapter, we discussed some of the important previous work in the area of processor memory interconnection. A taxonomy of all the work which has been described so far is shown on figure 2.4. All types of interconnection schemes can be broadly classified into two groups - multiple stage interconnection networks (MIN) and multiple bus interconnection networks (MBIN). Under the first scheme, we have distinguished between two types of memory access patterns, nemely, the uniform reference and the non-uniform reference models. Since no work has been done for uniform reference model under MBINs, we do not have any such classification in the second case.

CHAPTER 3

MARKOV MODEL FOR MULTIPROCESSORS WITH HOT SPOTS AND FAVORITE MEMORIES

In this chapter, we derive a discrete Markov model for shared memory multiprocessor systems in which each memory module has a buffer for queueing up unsatisfied memory requests. The following assumptions are made in order to develop our model, allowing two types of non-uniformity, namely, *favorite* and *hot* memories.

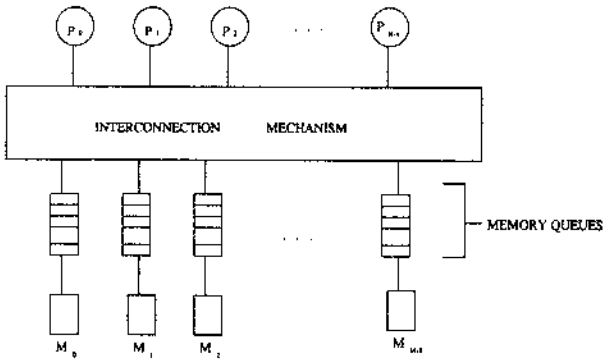


Figure 3.1: A Buffered memory system

1. There are N processors and M memory modules in the system; all processors are identical, so are all the memory modules.

2. When a processor requests access to a particular memory module, it is granted if the requested memory module is not busy.

3. There is no contention in the interconnection mechanism.

4. If two or more processors request access to the same memory module then only one of them is granted permission, and the rest are queued up in the queue

associated with the memory module. The process of selection is completely random.

5. The system is synchronous, i.e. all the requests made by the processors are at the beginning of the cycle. All the memory cycles are of equal length.

6. All the processors have an identical memory request rate λ.

7. The memory access of a processor may last over several memory cycles. The completion rate of a memory access by a processor is given by μ. In almost all related works, it has been tacitly assumed that $\mu = 1$, to simplify the models.

8. If there is a 'hot' memory module in the system, then the probability of accessing the hot memory is given by m. If there are K hot memories, they are all identical and memory reference is uniform among the hot memory modules.

9. The memory reference pattern among the "non-hot, non-favorite" memory modules is also uniform.

10. Successive memory requests follow the following scheme, which models the temporal locality concept of memory references. If in the current cycle, a processor accesses memory module M_j, then, given it completes its access in the current memory cycle, the probability that it will again request access from the module M_j is given by α. Also, references to other memory modules are uniformly distributed, with probability given by $\frac{1-\alpha}{M-1}$.

We follow a step by step approach in developing our model. We start with the case where there is only one hot memory in the system and the rest are all non-hot memories. The favorite memory case is not included in this analysis. Then we extend the model to K hot memories, again without any favorite memory. Next the favorite memory case is introduced along with a single hot memory. As the state space becomes enormously large for $M, N > 2$, we restrict our analysis to systems with two

processors and M memory modules, and N processors and two memory modules. A scheme for extension to the general case of $N \times M$ system using simulation, is discussed in Section 4. and the consequences of the experiments discussed. In Section 5, an approximate probabilistic analysis of a $N \times M$ system is given and a heuristic is proposed to find the value of K beyond which the memory bandwidth is expected to saturate.

3.1. $2 \times M$ system, one hot spot without any Favorite memory

There are two processors and M memory modules in the system. Of these M memory modules, one is a hot memory. Let m be the probability of request of a hot memory by any processor. Then, $\frac{1-m}{M-1}$ is the probability of request of any other $M-1$ non-hot memories. If λ is the request rate of a processor, then the probability that a processor requests the hot memory is given by, $p_h = \lambda m$. Probability that a processor requests a particular non-hot memory is, $p = \lambda\frac{(1-m)}{(M-1)}$. The next step is to enumerate all the possible states for this system.

The states are represented as two or three tuples, where each element gives the number of processor requests in the memory queue. Since there are only two processors in the system, at most two of the M memory modules have their memory queues filled up with processor request. We assume that the memory queue also holds the currently active processor. The first tuple always represents the number of processor requests in the queue of the hot memory. The remaining one (two) tuple(s) represents the number of requests in the queue of any one (two) (as there can be at most two memory modules holding a processor request) of the memory modules. A memory queue with no processor waiting or active is not always shown explicitly. They are usually shown as a zero tuple, when

(i) we want to distinguish between the non-existence of a processor in a hot memory module and the non-existence of a processor in a non-hot memory module. For example, the state $< 0, 1 >$ represents the situation where there is no processor in the

hot memory and only one processor in any one of the non-hot memory modules. (*ii*) the state 0 represented by $< 0, 0 >$ represents the situation when all processors are busy in their internal computation, and not accessing any memory module.

The enumeration of seven possible states of the $2 \times M$ system is given below.

state	*state vector*
S_0	$< 0, 0 >$
S_1	$< 0, 1 >$
S_2	$< 0, 1, 1 >$
S_3	$< 0, 2 >$
S_4	$< 1, 0 >$
S_5	$< 1, 1 >$
S_6	$< 2, 0 >$

3.1.1. Bandwidth Analysis

The state transition probabilities are next computed, as shown in Table 3.1. In this case,

$\bar{\lambda} = 1 - \lambda$;

$b = \binom{2}{1}\lambda(1 - \lambda)$;

$c = p(M - 1)$;

$d = p(M - 2)$;

$\bar{\mu} = 1 - \mu$;

$\bar{m} = 1 - m$;

This gives a 7×7 linear system of the form $A^T\Pi = \Pi$ where A is the state transition matrix and $\Pi = (\Pi_0, \Pi_1, \ldots, \Pi_6)$ is a vector containing the limiting probabilities such that Π_i denotes the limiting probability of state S_i. This system of linear equations can be solved by replacing any one of them by $\sum_{i=0}^{6} \Pi_i = 1$. After some simple matrix manipulations, the following reduced form of the linear system is obtained, which is

Table 3.1: **State Transition Matrix A of $2 \times M$ system**

states	0	1	2	3	4	5	6
0	$\bar{\lambda}^2$	$b\bar{m}$	cd	cp	bm	$2cp_h$	p_h^2
1	$\mu\bar{\lambda}^2$	$\mu b\bar{m} + \bar{\mu}\bar{\lambda}$	$\mu cd + \bar{\mu}d$	$\mu cp + \bar{\mu}p$	μbm	$2\mu cp_h + \bar{\mu}p_h$	μp_h^2
2	$\mu^2\bar{\lambda}^2$	$2\mu\bar{\mu}\bar{\lambda} + \mu^2 b\bar{m}$	$\bar{\mu}^2 + 2\mu\bar{\mu}d + \mu^2 cd$	$2\mu\bar{\mu}p + \mu^2 cp$	$\mu^2 bm$	$2\mu\bar{\mu}p_h + 2\mu^2 cp_h$	$\mu^2 p_h^2$
3	0	$\mu\bar{\lambda}$	μd	$\bar{\mu} + \mu p$	0	μp_h	0
4	$\mu\bar{\lambda}^2$	$\mu b\bar{m}$	μcd	μcp	$\bar{\mu}\bar{\lambda} + \mu bm$	$\bar{\mu}c + 2\mu cp_h$	$\bar{\mu}p_h + \mu p_h^2$
5	$\mu^2\bar{\lambda}^2$	$\mu\bar{\mu}\bar{\lambda} + \mu^2 b\bar{m}$	$\mu\bar{\mu}d + \mu^2 cd$	$\mu\bar{\mu}p + \mu^2 cp$	$\mu\bar{\mu}\bar{\lambda} + \mu^2 bm$	$\bar{\mu}^2 + \mu\bar{\mu}c + \mu\bar{\mu}p_h + 2\mu^2 cp_h$	$\bar{\mu}\mu p_h + \mu^2 p_f^2$
6	0	0	0	0	$\mu\bar{\lambda}$	μc	$\bar{\mu} + \mu p_h$

solved for the limiting probabilities. The seven equations are enumerated below.

$$(\bar{\lambda}^2 - 1)\Pi_0 + \mu\bar{\lambda}^2\Pi_1 + \mu^2\bar{\lambda}^2\Pi_2 + \mu\bar{\lambda}^2\Pi_4 + \mu^2\bar{\lambda}\Pi_5 = 0$$

$$\tfrac{b\bar{m}}{\bar{\lambda}^3}\Pi_0 + (\bar{\mu} - \tfrac{1}{\bar{\lambda}})\Pi_1 + 2\mu\bar{\mu}\Pi_2 + \mu\Pi_3 + \mu\bar{\mu}\Pi_5 = 0$$

$$(c\bar{\lambda} - b\bar{m})\Pi_0 + \bar{\lambda}^2\Pi_1 + \tfrac{\bar{\lambda}^3(\bar{\mu}^2 - 1)}{d}\Pi_2 = 0$$

$$(c\bar{\lambda} - b\bar{m})\Pi_0 + \bar{\lambda}^2\Pi_1 - \tfrac{\bar{\lambda}^3\mu}{p}\Pi_3 = 0$$

$$\tfrac{bm}{\bar{\lambda}^3}\Pi_0 + (\bar{\mu} - \tfrac{1}{\bar{\lambda}})\Pi_4 + \mu\bar{\mu}\Pi_5 + \mu\Pi_6 = 0$$

$$\tfrac{bm}{\bar{\lambda}^3}C_{6,0}\Pi_0 - \tfrac{1}{\bar{\lambda}}(d + p - 1)\Pi_1 + \Pi_2 + \Pi_3 + \tfrac{1}{\bar{\lambda}}\Pi_4 + \bar{\mu}^2\Pi_5 + \bar{\mu}\Pi_6 = 1$$

$$(\tfrac{p_h\bar{\lambda}}{bm} - 1)\Pi_0 + \tfrac{\bar{\lambda}^2}{bm}\Pi_4 - \tfrac{\bar{\lambda}^3\mu}{p_h bm}\Pi_6 = 0$$

where $C_{6,0} = -\left[1 + \left\{(1 - \tfrac{\bar{\lambda}}{b\bar{m}})\tfrac{1}{p} - (1 - \tfrac{c\bar{\lambda}}{b\bar{m}})(1 + \tfrac{d}{p})\right\}\tfrac{p\bar{m}}{m}\right]$.

This system of linear equations can be solved by Kramer's rule to obtain expressions for the Π_i's.

$$\Pi_0 = \frac{1}{D}\begin{vmatrix} 0 & \mu\bar{\lambda}^2 & \mu^2\bar{\lambda}^2 & 0 & \mu\bar{\lambda}^2 & \mu^2\bar{\lambda}^2 & 0 \\ 0 & \frac{\bar{\lambda}^2(\bar{\mu}\bar{\lambda}-1)}{b\bar{m}} & \frac{2\mu\bar{\mu}\bar{\lambda}^3}{b\bar{m}} & \frac{\mu\bar{\lambda}^3}{b\bar{m}} & 0 & \frac{\mu\bar{\mu}\bar{\lambda}^3}{b\bar{m}} & 0 \\ 0 & \frac{\bar{\lambda}^2}{b\bar{m}} & \frac{\bar{\lambda}^3(\bar{\mu}^2-1)}{bd\bar{m}} & 0 & 0 & 0 & 0 \\ 0 & \frac{\bar{\lambda}^2}{b\bar{m}} & 0 & -\frac{\mu\bar{\lambda}^3}{b\bar{m}p} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\bar{\lambda}^2(\bar{\mu}\bar{\lambda}-1)}{bm} & \frac{\mu\bar{\mu}\bar{\lambda}^3}{bm} & \frac{\mu\bar{\lambda}^3}{bm} \\ \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\lambda}^2}{b\bar{m}}(1-d-p) & \frac{\bar{\lambda}^3}{b\bar{m}} & \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\lambda}^2}{bm} & \frac{\bar{\mu}^2\bar{\lambda}^3}{bm} & \frac{\bar{\mu}\bar{\lambda}^3}{bm} \\ 0 & 0 & 0 & 0 & \frac{\bar{\lambda}^2}{bm} & 0 & -\frac{\mu\bar{\lambda}^3}{p_h bm} \end{vmatrix}$$

$$\Pi_1 = \frac{1}{D}\begin{vmatrix} \bar{\lambda}^2-1 & 0 & \mu^2\bar{\lambda}^2 & 0 & \mu\bar{\lambda}^2 & \mu^2\bar{\lambda}^2 & 0 \\ 1 & 0 & \frac{2\mu\bar{\mu}\bar{\lambda}^3}{b\bar{m}} & \frac{\mu\bar{\lambda}^3}{b\bar{m}} & 0 & \frac{\mu\bar{\mu}\bar{\lambda}^3}{b\bar{m}} & 0 \\ \frac{c\bar{\lambda}}{b\bar{m}}-1 & 0 & \frac{\bar{\lambda}^3(\bar{\mu}^2-1)}{bd\bar{m}} & 0 & 0 & 0 & 0 \\ \frac{c\bar{\lambda}}{b\bar{m}}-1 & 0 & 0 & -\frac{\mu\bar{\lambda}^3}{b\bar{m}p} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & \frac{\bar{\lambda}^2(\bar{\mu}\bar{\lambda}-1)}{bm} & \frac{\mu\bar{\mu}\bar{\lambda}^3}{bm} & \frac{\mu\bar{\lambda}^3}{bm} \\ C_{6,0} & \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\lambda}^2}{bm} & \frac{\bar{\mu}^2\bar{\lambda}^3}{bm} & \frac{\bar{\mu}\bar{\lambda}^3}{bm} \\ \frac{p_h\bar{\lambda}}{bm}-1 & 0 & 0 & 0 & \frac{\bar{\lambda}^2}{bm} & 0 & -\frac{\mu\bar{\lambda}^3}{p_h bm} \end{vmatrix}$$

$$\Pi_2 = \frac{1}{D}\begin{vmatrix} \bar{\lambda}^2-1 & \mu\bar{\lambda}^2 & 0 & 0 & \mu\bar{\lambda}^2 & \mu^2\bar{\lambda}^2 & 0 \\ 1 & \frac{\bar{\lambda}^2(\bar{\mu}\bar{\lambda}-1)}{b\bar{m}} & 0 & \frac{\mu\bar{\lambda}^3}{b\bar{m}} & 0 & \frac{\mu\bar{\mu}\bar{\lambda}^3}{b\bar{m}} & 0 \\ \frac{c\bar{\lambda}}{b\bar{m}}-1 & \frac{\bar{\lambda}^2}{b\bar{m}} & 0 & 0 & 0 & 0 & 0 \\ \frac{c\bar{\lambda}}{b\bar{m}}-1 & \frac{\bar{\lambda}^2}{b\bar{m}} & 0 & -\frac{\mu\bar{\lambda}^3}{b\bar{m}p} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & \frac{\bar{\lambda}^2(\bar{\mu}\bar{\lambda}-1)}{bm} & \frac{\mu\bar{\mu}\bar{\lambda}^3}{bm} & \frac{\mu\bar{\lambda}^3}{bm} \\ C_{6,0} & \frac{\bar{\lambda}^2}{b\bar{m}}(1-d-p) & \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\lambda}^2}{bm} & \frac{\bar{\mu}^2\bar{\lambda}^3}{bm} & \frac{\bar{\mu}\bar{\lambda}^3}{bm} \\ \frac{p_h\bar{\lambda}}{bm}-1 & 0 & 0 & 0 & \frac{\bar{\lambda}^2}{bm} & 0 & -\frac{\mu\bar{\lambda}^3}{p_h bm} \end{vmatrix}$$

$$\Pi_3 = \frac{1}{D}\begin{vmatrix} \bar{\lambda}^2-1 & \mu\bar{\lambda}^2 & \mu^2\bar{\lambda}^2 & 0 & \mu\bar{\lambda}^2 & \mu^2\bar{\lambda}^2 & 0 \\ 1 & \frac{\bar{\lambda}^2(\bar{\mu}\bar{\lambda}-1)}{b\bar{m}} & \frac{2\mu\bar{\mu}\bar{\lambda}^3}{b\bar{m}} & 0 & 0 & \frac{\mu\bar{\mu}\bar{\lambda}^3}{b\bar{m}} & 0 \\ \frac{c\bar{\lambda}}{b\bar{m}}-1 & \frac{\bar{\lambda}^2}{b\bar{m}} & \frac{\bar{\lambda}^3(\bar{\mu}^2-1)}{bd\bar{m}} & 0 & 0 & 0 & 0 \\ \frac{c\bar{\lambda}}{b\bar{m}}-1 & \frac{\bar{\lambda}^2}{b\bar{m}} & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & \frac{\bar{\lambda}^2(\bar{\mu}\bar{\lambda}-1)}{bm} & \frac{\mu\bar{\mu}\bar{\lambda}^3}{bm} & \frac{\mu\bar{\lambda}^3}{bm} \\ C_{6,0} & \frac{\bar{\lambda}^2}{b\bar{m}}(1-d-p) & \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\lambda}^2}{bm} & \frac{\bar{\mu}^2\bar{\lambda}^3}{bm} & \frac{\bar{\mu}\bar{\lambda}^3}{bm} \\ \frac{p_h\bar{\lambda}}{bm}-1 & 0 & 0 & 0 & \frac{\bar{\lambda}^2}{bm} & 0 & -\frac{\mu\bar{\lambda}^3}{p_h bm} \end{vmatrix}$$

$$\Pi_4 = \frac{1}{D}\begin{vmatrix} \bar{\lambda}^2-1 & \mu\bar{\lambda}^2 & \mu^2\bar{\lambda}^2 & 0 & 0 & \mu^2\bar{\lambda}^2 & 0 \\ 1 & \frac{\bar{\lambda}^2(\bar{\mu}\bar{\lambda}-1)}{b\bar{m}} & \frac{2\mu\bar{\mu}\bar{\lambda}^3}{b\bar{m}} & \frac{\mu\bar{\lambda}^3}{b\bar{m}} & 0 & \frac{\mu\bar{\mu}\bar{\lambda}^3}{b\bar{m}} & 0 \\ \frac{c\bar{\lambda}}{b\bar{m}}-1 & \frac{\bar{\lambda}^2}{b\bar{m}} & \frac{\bar{\lambda}^3(\bar{\mu}^2-1)}{bd\bar{m}} & 0 & 0 & 0 & 0 \\ \frac{c\bar{\lambda}}{b\bar{m}}-1 & \frac{\bar{\lambda}^2}{b\bar{m}} & 0 & -\frac{\mu\bar{\lambda}^3}{b\bar{m}p} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & \frac{\mu\bar{\mu}\bar{\lambda}^3}{bm} & \frac{\mu\bar{\lambda}^3}{bm} \\ C_{6,0} & \frac{\bar{\lambda}^2}{bm}(1-d-p) & \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\mu}^2\bar{\lambda}^3}{bm} & \frac{\bar{\mu}\bar{\lambda}^3}{bm} \\ \frac{p_h\bar{\lambda}}{bm}-1 & 0 & 0 & 0 & 0 & 0 & -\frac{\mu\bar{\lambda}^3}{p_h bm} \end{vmatrix}$$

$$\Pi_5 = \frac{1}{D}\begin{vmatrix} \bar{\lambda}^2-1 & \mu\bar{\lambda}^2 & \mu^2\bar{\lambda}^2 & 0 & \mu\bar{\lambda}^2 & 0 & 0 \\ 1 & \frac{\bar{\lambda}^2(\mu\bar{\lambda}-1)}{b\bar{m}} & \frac{2\mu\bar{\mu}\bar{\lambda}^3}{b\bar{m}} & \frac{\mu\bar{\lambda}^3}{b\bar{m}} & 0 & 0 & 0 \\ \frac{c\bar{\lambda}}{b\bar{m}}-1 & \frac{\bar{\lambda}^2}{b\bar{m}} & \frac{\bar{\lambda}^3(\bar{\mu}^2-1)}{bd\bar{m}} & 0 & 0 & 0 & 0 \\ \frac{c\bar{\lambda}}{b\bar{m}}-1 & \frac{\bar{\lambda}^2}{b\bar{m}} & 0 & -\frac{\mu\bar{\lambda}^3}{b\bar{m}p} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & \frac{\bar{\lambda}^2(\mu\bar{\lambda}-1)}{bm} & 0 & \frac{\mu\bar{\lambda}^3}{bm} \\ C_{6,0} & \frac{\bar{\lambda}^2}{bm}(1-d-p) & \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\lambda}^2}{bm} & \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\mu}\bar{\lambda}^3}{bm} \\ \frac{p_h\bar{\lambda}}{bm}-1 & 0 & 0 & 0 & \frac{\bar{\lambda}^2}{bm} & 0 & -\frac{\mu\bar{\lambda}^3}{p_h bm} \end{vmatrix}$$

$$\Pi_6 = \frac{1}{D}\begin{vmatrix} \bar{\lambda}^2-1 & \mu\bar{\lambda}^2 & \mu^2\bar{\lambda}^2 & 0 & \mu\bar{\lambda}^2 & \mu^2\bar{\lambda}^2 & 0 \\ 1 & \frac{\bar{\lambda}^2(\mu\bar{\lambda}-1)}{b\bar{m}} & \frac{2\mu\bar{\mu}\bar{\lambda}^3}{b\bar{m}} & \frac{\mu\bar{\lambda}^3}{b\bar{m}} & 0 & \frac{\mu\bar{\mu}\bar{\lambda}^3}{b\bar{m}} & 0 \\ \frac{c\bar{\lambda}}{b\bar{m}}-1 & \frac{\bar{\lambda}^2}{b\bar{m}} & \frac{\bar{\lambda}^3(\bar{\mu}^2-1)}{bd\bar{m}} & 0 & 0 & 0 & 0 \\ \frac{c\bar{\lambda}}{b\bar{m}}-1 & \frac{\bar{\lambda}^2}{b\bar{m}} & 0 & -\frac{\mu\bar{\lambda}^3}{b\bar{m}p} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & \frac{\bar{\lambda}^2(\mu\bar{\lambda}-1)}{bm} & \frac{\mu\bar{\mu}\bar{\lambda}^3}{bm} & 0 \\ C_{6,0} & \frac{\bar{\lambda}^2}{bm}(1-d-p) & \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\lambda}^2}{bm} & \frac{\mu^2\bar{\lambda}^3}{bm} & \frac{\bar{\lambda}^3}{bm} \\ \frac{p_h\bar{\lambda}}{bm}-1 & 0 & 0 & 0 & \frac{\bar{\lambda}^2}{bm} & 0 & 0 \end{vmatrix}$$

where,

$$D = \begin{vmatrix} \bar{\lambda}^2-1 & \mu\bar{\lambda}^2 & \mu^2\bar{\lambda}^2 & 0 & \mu\bar{\lambda}^2 & \mu^2\bar{\lambda}^2 & 0 \\ 1 & \frac{\bar{\lambda}^2(\mu\bar{\lambda}-1)}{b\bar{m}} & \frac{2\mu\bar{\mu}\bar{\lambda}^3}{b\bar{m}} & \frac{\mu\bar{\lambda}^3}{b\bar{m}} & 0 & \frac{\mu\bar{\mu}\bar{\lambda}^3}{b\bar{m}} & 0 \\ \frac{c\bar{\lambda}}{b\bar{m}}-1 & \frac{\bar{\lambda}^2}{b\bar{m}} & \frac{\bar{\lambda}^3(\bar{\mu}^2-1)}{bd\bar{m}} & 0 & 0 & 0 & 0 \\ \frac{c\bar{\lambda}}{b\bar{m}}-1 & \frac{\bar{\lambda}^2}{b\bar{m}} & 0 & -\frac{\mu\bar{\lambda}^3}{b\bar{m}p} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & \frac{\bar{\lambda}^2(\mu\bar{\lambda}-1)}{bm} & \frac{\mu\bar{\mu}\bar{\lambda}^3}{bm} & \frac{\mu\bar{\lambda}^3}{bm} \\ C_{6,0} & \frac{\bar{\lambda}^2}{bm}(1-d-p) & \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\lambda}^3}{bm} & \frac{\bar{\lambda}^2}{bm} & \frac{\bar{\mu}^2\bar{\lambda}^3}{bm} & \frac{\bar{\mu}\bar{\lambda}^3}{bm} \\ \frac{p_h\bar{\lambda}}{bm}-1 & 0 & 0 & 0 & \frac{\bar{\lambda}^2}{bm} & 0 & -\frac{\mu\bar{\lambda}^3}{p_h bm} \end{vmatrix}$$

From the state table of the $2 \times M$ system, the bandwidth of the system is given by,

$$BW = \Pi_1 + 2\Pi_2 + \Pi_3 + \Pi_4 + 2\Pi_5 + \Pi_6$$

The system, in all other states except S_2 and S_5, have only one active memory module. In state S_2 and in state S_5, it has two. Some of the values of bandwidth obtained for various values of the input parameters are shown in Table 3.2.

We find, from the results in Table 3.2, that if the request service rate, μ is increased keeping the request arrival rate, λ, constant, all other parameters remaining the same, the memory bandwidth decreases. This can be intuitively explained as follows. Increasing the service rate of the processing element (in this case a memory module) leads to higher rate of completion of jobs, and without additional jobs coming in

Table 3.2: **Memory bandwidth with varying λ and μ for $M = 10$ and $m = 0.75$.**

λ / μ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	.91									
0.2	.67	.95								
0.3	.52	.82	.99							
0.4	.42	.71	.91	1.04						
0.5	.36	.63	.83	.98	1.08					
0.6	.31	.56	.77	.93	1.04	1.13				
0.7	.27	.51	.71	.88	1.01	1.11	1.18			
0.8	.24	.46	.66	.83	.97	1.08	1.17	1.23		
0.9	.21	.43	.62	.79	.94	1.06	1.15	1.22	1.27	
1.0	.20	.39	.58	.75	.90	1.03	1.14	1.22	1.27	1.31

to fill the empty slots, the processing element tends to remain inactive for more amount of time. In our case, this directly leads to a decrease in the number of active memory modules, thereby decreasing the effective bandwidth. This phenomenon is also reflected in a decrease in the mean memory queue length with increasing μ and constant λ (see graphs).

On the other hand, memory bandwidth seems to increase with increasing λ and constant μ. Service rate remaining the same, an increase in the arrival rate of requests leads to more number of active memory modules. Of course, we have assumed that $\lambda \leq \mu$, so that requests don't pile up in memory queues leading to a decrease in bandwidth.

3.1.2. Mean Queue Length and Waiting Time

Let MQL_h and MQL_{nh} be the mean queue lengths for a hot memory module and non-hot memory module, respectively. Since we are dealing with a system of multiple

queues, we actually consider the mean of the maximum queue lengths in the system. As is evident from the state table, these are given by the following expressions.

$$MQL_h = \Pi_4 + \Pi_5 + 2\Pi_6$$
$$MQL_{nh} = \Pi_1 + \Pi_2 + 2\Pi_3 + \Pi_5.$$

The mean queue length for the whole system is given by

$$MQL_s = \Pi_1 + \Pi_2 + 2\Pi_3 + \Pi_4 + \Pi_5 + 2\Pi_6.$$

The arrival rate of requests for a hot memory module is given by $m\lambda$, hence by Little's law, the mean waiting time for a processor request in a hot module is given by $MWT_h = \frac{MQL_h}{m\lambda}$. Similarly, for a non-hot memory module, the request arrival rate is given as $\frac{(1-m)\lambda}{M-1}$ and, by Little's law, the mean waiting time for a request in a non-hot memory module is $MWT_{nh} = \frac{MQL_{nh}}{\frac{(1-m)\lambda}{M-1}}$. Since the probability of accessing a hot memory module is m, and that for a non-hot module is $(1-m)$, the mean waiting time for the whole system is given as follows.

$$\begin{aligned} MWT_s &= m \times MWT_h + (1-m) \times MWT_{nh} \\ &= \frac{1}{\lambda}(MQL_h + (M-1) \times MQL_{nh}). \end{aligned}$$

The variations of the mean queue length and mean waiting time are depicted graphically (Figure 3.2).

3.2. $2 \times M$ system, K hot spots and no favorite memory

There are K hot memories and $(M-K)$ non-hot memories. We assume that within the K hot memories the access pattern is uniform. The probability of accessing hot memories is given as before by, $m\lambda$. Hence, the probability of accessing a particular hot memory is given as, $p_h = \frac{m\lambda}{K}$. Similarly, probability that a processor requests a particular non-hot memory module is given by $p = \lambda\frac{(1-m)}{(M-K)}$.

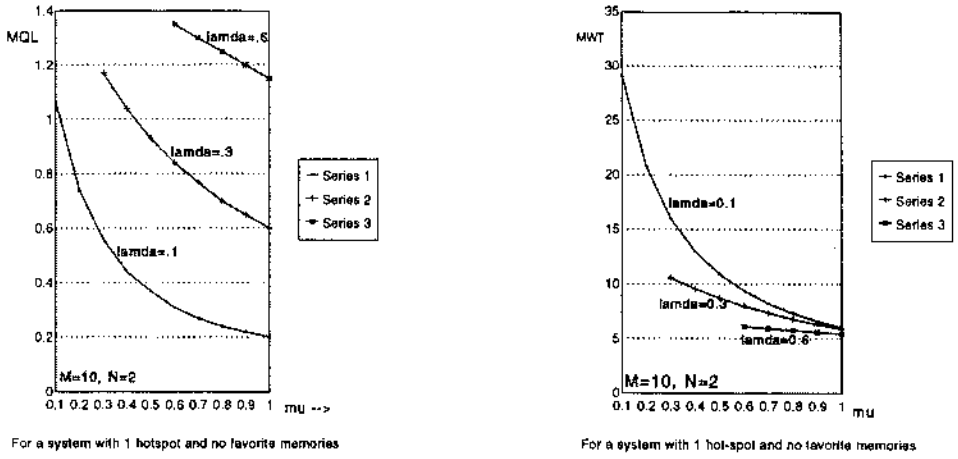


Figure 3.2: Variation of mean queue length and mean waiting time with request service rate

The possible states are enumerated as in Section 3.1 with one additional state. As there are multiple hot memories, the two processors can access two hot memories simultaneously, and this leads to the eighth state $< 1, 1, 0 >$, where the first two tuples stand for any two hot memory queues. In all other cases the first tuple represents a hot memory queue, while the rest of the tuples represent non-hot memory queues. The possible enumeration of states is given below.

state	*state vector*
S_0	$< 0,0 >$
S_1	$< 0,1 >$
S_2	$< 0,1,1 >$
S_3	$< 0,2 >$
S_4	$< 1,0 >$
S_5	$< 1,1 >$
S_6	$< 2,0 >$
S_7	$< 1,1,0 >$

The state transition probabilities are next computed, which are shown in Table 3.3.

Table 3.3: **State Transition Matrix of $2 \times M$ system with K hot spots**

states	0	1	2	3	4	5	6	7
0	$\bar{\lambda}^2$	$b\bar{m}$	$c'd'$	$c'p$	bm	$2c'p_h k$	$p_h^2 k$	$p_h k p_h (k-1)$
1	$\mu\bar{\lambda}^2$	$\mu b\bar{m} + \bar{\mu}\bar{\lambda}$	$\mu c'd' + \bar{\mu}d'$	$\mu c'p + \bar{\mu}p$	μbm	$2\mu c'p_h k + \bar{\mu}p_h k$	$\mu p_h^2 k$	$\mu p_h k p_h (k-1)$
2	$\mu^2\bar{\lambda}^2$	$2\mu\bar{\mu}\bar{\lambda} + \mu^2 b\bar{m}$	$\bar{\mu}^2 + 2\mu\bar{\mu}d' + \mu^2 c'd'$	$2\mu\bar{\mu}p + \mu^2 c'p$	$\mu^2 bm$	$2\mu\bar{\mu}p_h k + 2\mu^2 c'p_h k$	$\mu^2 p_h^2 k$	$\mu^2 p_h k p_h (k-1)$
3	0	$\mu\bar{\lambda}$	$\mu d'$	$\bar{\mu} + \mu p$	0	$\mu p_h k$	0	0
4	$\mu\bar{\lambda}^2$	μbm	$\mu c'd'$	$\mu c'p$	$\bar{\mu}\bar{\lambda} + \mu bm$	$\bar{\mu}c' + 2\mu c'p_h k$	$\bar{\mu}p_h + \mu p_h^2 k$	$\bar{\mu}p_h(k-1) + \mu p_h k p_h (k-1)$
5	$\mu^2\bar{\lambda}^2$	$\mu\bar{\mu}\bar{\lambda} + \mu^2 bm$	$\mu\bar{\mu}d' + \mu^2 c'd'$	$\mu\bar{\mu}p + \mu^2 c'p$	$\mu\bar{\mu}\bar{\lambda} + \mu^2 bm$	$\bar{\mu}^2 + \mu\bar{\mu}c' + \mu\bar{\mu}p_h k + 2\mu^2 c'p_h k$	$\bar{\mu}\mu p_h + \mu^2 p_h^2 k$	$\mu\bar{\mu}p_h(k-1) + \mu^2 p_h k p_h (k-1)$
6	0	0	0	0	$\mu\bar{\lambda}$	$\mu c'$	$\bar{\mu} + \mu p_h$	$\mu p_h (k-1)$
7	$\mu^2\bar{\lambda}^2$	$\mu^2 bm$	$\mu^2 c'd'$	$\mu^2 c'p$	$\mu\bar{\mu}\bar{\lambda} + \mu^2 bm$	$\mu\bar{\mu}c' + \mu^2 2c'p_h k$	$\mu\bar{\mu}p_h + \mu^2 p_h^2 k$	$\bar{\mu}^2 + 2\mu\bar{\mu}p_h(k-1) + \mu^2 p_h k p_h (k-1)$

In this case,

$\bar{\lambda} = 1 - \lambda$;

$b = \binom{2}{1}\lambda(1-\lambda)$;

$c' = p(M-K)$;

$d' = p(M-K-1)$;

$\bar{\mu} = 1-\mu$;

$\bar{m} = 1-m$;

The linear system of the state transition equations are solved by a standard Fortran linear system solver, for various values of the five input parameters λ, μ, m, K, and M, which are respectively the request rate, the service rate, probability of request to a hot memory module, the number of hot modules and the total number of memory modules. The output vector, $\{\Pi_0, \Pi_1, \Pi_2 \ldots \Pi_7\}$ of the linear system solver gives the limiting probabilities for the eight states, $\{S_0, S_1 \ldots S_7\}$ of the processor-memory system. The bandwidth of the system can be obtained by the following expression.

$$BW = \Pi_1 + 2\Pi_2 + \Pi_3 + \Pi_4 + 2\Pi_5 + \Pi_6 + 2\Pi_7$$

Next the mean queue length and the mean waiting time for the system is computed. As in Section 3.1, we consider the mean of the maximum queue lengths in the system. As is evident from the state table, the mean queue lengths for a hot memory module and non-hot module are given as follows.

$$MQL_h = \Pi_4 + \Pi_5 + 2\Pi_6 + \Pi_7$$
$$MQL_{nh} = \Pi_1 + \Pi_2 + 2\Pi_3 + \Pi_5 + \Pi_7.$$

The mean queue length for the whole system is given by,

$$MQL_s = \Pi_1 + \Pi_2 + 2\Pi_3 + \Pi_4 + \Pi_5 + 2\Pi_6 + \Pi_7$$

The arrival rate of requests for a hot memory module is given by $\frac{m\lambda}{K}$, hence by Little's law the mean waiting time for a processor request in a hot module, MWT_h, is given by $MWT_h = \frac{MQL_h}{\frac{m\lambda}{K}}$. Similarly, for a non-hot memory module, the request arrival rate is given as $\frac{(1-m)\lambda}{M-K}$ and, by Little's law the mean waiting time for a request in a

non-hot memory module is, $MWT_{nh} = \frac{MQL_{nh}}{\frac{(1-m)\lambda}{M-K}}$. Since the probability of accessing a hot memory module is m, and that for a non-hot module is $(1-m)$, the mean waiting time for the whole system is given as follows.

$$\begin{aligned} MWT_s &= m \times MWT_h + (1-m) \times MWT_{nh} \\ &= \frac{1}{\lambda}(K \times MQL_h + (M-K) \times MQL_{nh}) \end{aligned}$$

Some of the values of bandwidth obtained for various values of the input parameters are shown in the Tables 3.4 - 3.6. The variations of the mean queue length and mean waiting time are depicted graphically (Figure 3.3).

Table 3.4: **Memory bandwidth with varying λ and μ for $K = 1, M = 10$ and $m = 0.75$.**

λ / μ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	.91									
0.2	.67	.95								
0.3	.52	.82	.99							
0.4	.42	.71	.91	1.04						
0.5	.36	.63	.83	.98	1.08					
0.6	.31	.56	.77	.93	1.04	1.13				
0.7	.27	.51	.71	.88	1.01	1.11	1.18			
0.8	.24	.46	.66	.83	.97	1.08	1.17	1.23		
0.9	.21	.43	.62	.79	.94	1.06	1.15	1.22	1.27	
1.0	.20	.39	.58	.75	.90	1.03	1.14	1.22	1.27	1.31

Table 3.5: **Memory bandwidth with varying λ and μ for $k = 5, M = 10$ and $m = 0.75$.**

λ / μ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	1.11									
0.2	.75	1.15								
0.3	.56	.94	1.20							
0.4	.45	.79	1.06	1.26						
0.5	.37	.68	.94	1.15	1.33					
0.6	.31	.60	.84	1.06	1.24	1.404				
0.7	.28	.53	.76	.97	1.16	1.34	1.49			
0.8	.24	.47	.69	.90	1.10	1.28	1.44	1.59		
0.9	.22	.43	.64	.84	1.04	1.22	1.40	1.56	1.72	
1.0	.20	.40	.60	.79	.98	1.17	1.35	1.53	1.70	1.87

Table 3.6: **Memory bandwidth with varying λ and μ for $k = 5, M = 10$ and $m = 0.95$.**

λ / μ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	1.14									
0.2	.77	1.18								
0.3	.57	.96	1.22							
0.4	.45	.81	1.07	1.27						
0.5	.37	.69	.95	1.16	1.33					
0.6	.32	.60	.85	1.06	1.24	1.40				
0.7	.27	.53	.77	.98	1.17	1.33	1.48			
0.8	.25	.48	.70	.91	1.09	1.27	1.43	1.57		
0.9	.22	.43	.64	.84	1.03	1.21	1.38	1.53	1.68	
1.0	.20	.40	.59	.78	.97	1.15	1.33	1.50	1.66	1.81

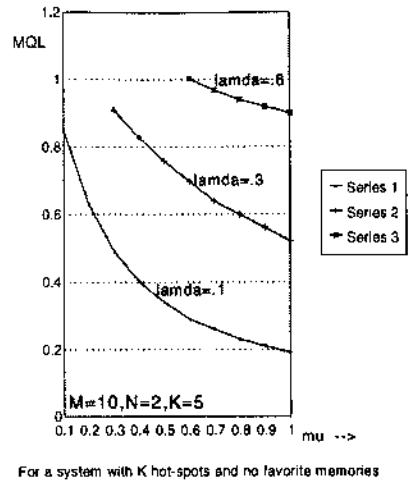


Figure 3.3: Variation of mean memory-queue length with request service rate

Table 3.4 gives the bandwidth values for $K = 1$, which corresponds to the case taken up in Section 3.1.1. The bandwidth values obtained by increasing the value of K to 5, are displayed in Table 3.5, while Table 3.6 gives the bandwidths for $K = 5$ but with increased m. It is observed from the results above that the memory bandwidth increases as the number, K, of hot spots is increased, because the greater memory traffic towards the hot memories is now distributed among the new modules. This leads to a decrease in the number of requesting processors queued up for memory access, thereby increasing the overall bandwidth. It is also observed that the bandwidths increase slightly as the probability of accessing hot modules, m, is increased. This will be true as long as there are sufficient number of hot memory modules to take care of the increased traffic in their direction. Here, for $N = 2$, five hot modules (refer Tables 3.5 and 3.6) are more than sufficient, as there cannot be more than two simultaneous requests to the hot modules. But, as we shall see in Chapter 4, in the general case of the $N \times M$ system, the bandwidth actually decreases when m is increased to the extent that the number of hot modules is not sufficient to meet the higher request rate.

3.3. $2 \times M$ system, one hot spot with favorite memory

We consider the system which has only one hot spot and also favorite memories among the non-hot modules. The term "favorite" is explained as follows. If a processor is accesssing the memory module M_i in the current cycle, then it will request the same memory module in the next cycle (provided it has completed its memory access by the current cycle) with probability α and the remaining $M-1$ memory modules with probability $1-\alpha$. Then the modified request rate of a processor to its favourite memory module is given as $\alpha\lambda$, and the modified request rate to non-favorite memories is $(1-\alpha)\lambda$. We assume that the hot memory is not a favorite memory to any of the processors, and the favorite memory is one of the non-hot memories only. The states remain the same as in the case dealing with one hot memory and no favorite memory (Section 3.1). Only changes are in the case of some of the transition probabilities due to the introduction of one additional parameter α. The enumeration of states is repeated here for convenience.

state	*state vector*
S_0	$< 0,0 >$
S_1	$< 0,1 >$
S_2	$< 0,1,1 >$
S_3	$< 0,2 >$
S_4	$< 1,0 >$
S_5	$< 1,1 >$
S_6	$< 2,0 >$

The corresponding state transition probabilities are shown in Table 3.7.

In this case, $f_1 = \alpha p + (1-\alpha)p(M-2)$, $f_2 = \alpha p + (1-\alpha)p(M-3)$, and all the other parameters are the same as in Section 3.1.

The linear system of the state transition equations are solved by a standard linear system solver, for various values of the five input parameters λ, μ, α, m, and M, which

Table 3.7: **State Transition Matrix of $2\times M$ system with hot spot and favorite memories**

states	0	1	2	3	4	5	6
0	$\bar{\lambda}^2$	$b\bar{m}$	cd	cp	bm	$2cp_h$	p_h^2
1	$\mu\bar{\lambda}^2$	$2\mu\bar{\lambda}f_1 + \bar{\mu}\bar{\lambda}$	$\mu f_1 d + \bar{\mu}d$	$\mu f_1 p + \bar{\mu}p$	μbm	$2\mu f_1 p_h + \bar{\mu}p_h$	μp_h^2
2	$\mu^2\bar{\lambda}^2$	$2\mu\bar{\mu}\bar{\lambda} + \mu^2 b\bar{m}$	$\bar{\mu}^2 + 2\mu\bar{\mu}f_2 + \mu^2 f_1 f_2$	$2\mu\bar{\mu}\alpha p + \mu^2 f_1 p$	$\mu^2 bm$	$2\mu\bar{\mu}p_h + 2\mu^2 f_1 p_h$	$\mu^2 p_h^2$
3	0	$\mu\bar{\lambda}$	μd	$\bar{\mu} + \mu\alpha p$	0	μp_h	0
4	$\mu\bar{\lambda}^2$	$\mu b\bar{m}$	μcd	μcp	$\bar{\mu}\bar{\lambda} + \mu bm$	$\bar{\mu}c + 2\mu cp_h$	$\bar{\mu}p_h + \mu p_h^2$
5	$\mu^2\bar{\lambda}^2$	$\mu\bar{\mu}\bar{\lambda} + \mu^2 2\bar{\lambda}f_1$	$\mu\bar{\mu}d + \mu^2 f_1 d$	$\mu\bar{\mu}p + \mu^2 f_1 p$	$\mu\bar{\mu}\bar{\lambda} + \mu^2 bm$	$\bar{\mu}^2 + \mu\bar{\mu}f_1 + \mu\bar{\mu}p_h + 2\mu^2 f_1 p_h$	$\bar{\mu}\mu p_h + \mu^2 p_h^2$
6	0	0	0	0	$\mu\bar{\lambda}$	μc	$\bar{\mu} + \mu p_h$

are respectively, the request rate, the service rate, probability of request to favorite memory, probability of request to the hot memory module, and the total number of memory modules. The output vector, $\{\Pi_0, \Pi_1, \Pi_2 \ldots \Pi_6\}$ of the linear system solver gives the limiting probabilities for the seven states, $\{S_0, S_1 \ldots S_6\}$ of the processor-memory system. The bandwidth of the system can be obtained by, $BW = \Pi_1 + 2\Pi_2 + \Pi_3 + \Pi_4 + 2\Pi_5 + \Pi_6$. The computation of mean queue length and mean waiting time for a processor request are exactly similar to those in Section 3.1.2. We can derive expressions for the mean waiting time for the favorite memory requests and that for the non-favorite requests to a non-hot memory module. Thus, we get $MWT_f = \frac{MQL_{nh}}{\frac{(1-m)\lambda\alpha}{M-1}}$ and $MWT_{nf} = \frac{MQL_{nh}}{\frac{(1-m)\lambda(1-\alpha)}{M-1}}$. Combining the above two expressions for the

waiting times in the favorite and non-favorite cases, the mean waiting time for a non-hot memory module is given by, $MWT_{nh} = \alpha MWT_f + (1-\alpha)MWT_{nf}$, which gives exactly the same formula for MWT_{nh} as derived in Section 3.1.

Some of the values of bandwidth obtained for various values of the input parameters are shown in Tables 3.8 and 3.9. The variation of mean waiting time of a memory request with various λ and μ is depicted graphically (Figure 3.4).

Table 3.8: **Memory bandwidth for various λ and μ, $M = 10, m = 0.75$ and $\alpha = 0.6$.**

λ / μ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	.91									
0.2	.67	.94								
0.3	.52	.82	.98							
0.4	.42	.71	.90	1 .02						
0.5	.36	.63	.83	.97	1.06					
0.6	.31	.57	.77	.92	1.03	1.10				
0.7	.27	.51	.71	.87	.99	1.08	1.14			
0.8	.24	.47	.66	.83	.96	1.06	1.12	1.17		
0.9	.22	.43	.62	.79	.93	1.04	1.11	1.16	1.19	
1.0	.20	.39	.58	.75	.90	1.01	1.10	1.16	1.19	1.2

Table 3.9: **Memory bandwidth for various α and μ, $M = 10, m = 0.75$ and $\alpha = 0.9$.**

λ / μ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	.91									
0.2	.67	.94								
0.3	.52	.82	.98							
0.4	.42	.71	.90	1 .02						
0.5	.36	.63	.83	.97	1.06					
0.6	.31	.57	.77	.92	1.03	1.10				
0.7	.27	.51	.71	.87	.99	1.08	1.14			
0.8	.24	.47	.66	.83	.96	1.06	1.13	1.17		
0.9	.22	.43	.62	.79	.93	1.04	1.11	1.17	1.2	
1.0	.20	.4	.59	.75	.90	1.02	1.10	1.16	1.2	1.21

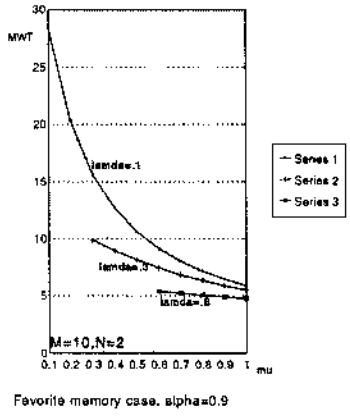


Figure 3.4: Variation of mean mean waiting time with request service rate

With the same values of M and m and with $\alpha > .5$ the average bandwidth in case of the system with favorite memory modules is less than that without such modules. This can be intuitively seen as follows. Without favorite memories, the processors' requests to non-hot memory modules are evenly distributed, while with the introduction of favorite memories, the requests tend to queue up to the same memory modules thereby decreasing the average number of active memories and hence the overall bandwidth.

3.4. $N \times 2$ system, one hot and one non-hot non-favorite memory

There are N processors and 2 memory modules in the system. Out of these two modules, one is hot and the other is non-hot. We do not consider here the favorite memory case. We also assume that the probability of request by a processor at any memory cycle is unity. That is, $\lambda = 1$. Consequently, $\mu = 1$ too. This assumption, though unrealistic, considerably simplifies the model. The number of states in this case is $N + 1$. The state vectors are as follows.

state	*state vector*
S_0	$< N, 0 >$
S_1	$< N-1, 1 >$
S_2	$< N-2, 2 >$
.	.
.	.
.	.
$S_{N/2}$	$< N/2, N/2 >$
.	.
.	.
.	.
S_{N-1}	$< 1, N-1 >$
S_N	$< 0, N >$

The Markov chain model for the system is shown in Figure 3.5.

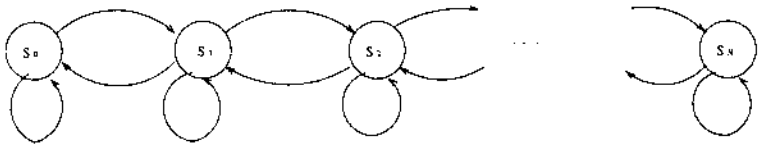


Figure 3.5: Markov chain model for $N \times 2$ system

To find the state transition probabilities, we consider three cases. The general case is for the ith state, S_i, of the Markov chain where $i \neq 0$ or N. Other two cases are for the boundary conditions, i.e. for the first and the last states of the chain (Fig. 3.5).

Transition probabilities for state S_i, where $i \neq 0$ or N.

The probability of transition from state S_i to the same state is given by $\alpha_{i,i} = m(1-m)$. Let $\alpha_{i,j}$ be the probability of transition from state S_i to state S_j. Then,

$$\alpha_{i,i-1} = m^2$$
$$\alpha_{i-1,i} = (1-m)^2$$

Transition probabilities for state S_0.

We need to compute only the outgoing transition probabilities for state S_0, as the incoming transition probabilities will be the same as in the general case. The required transition probabilities are obtained as follows:

$$\alpha_{0,0} = m$$
$$\alpha_{0,1} = 1-m$$
$$\alpha_{1,1} = \alpha_{i,i}, \text{ for } i = 1$$
$$\alpha_{1,0} = \alpha_{i,i-1}, \text{ for } i = 1$$

Transition probabilities for state S_N.

We need to compute only the outgoing transition probabilities for state S_N, as the incoming transition probabilities will be the same as in the general case. Thus

$$\alpha_{N,N} = 1-m$$

$$\alpha_{N,N-1} = m$$
$$\alpha_{N-1,N-1} = \alpha_{i,i}, \text{ for } i = N-1$$
$$\alpha_{N-1,N} = \alpha_{i,i-1}, \text{ for } i = N-1$$

3.4.1. Solutions of the state transition equations.

The state transition equations of the system are solved to obtain the limiting probabilities of the states in our Markov model. Let Π_i denote the limiting probability of state S_i. Solving the state equations,

$$\Pi_0 = \frac{1}{1+\frac{(1-m)(1-l^N)}{m^2(1-l)}} \quad \text{and} \quad \Pi_i = \frac{(1-m)^{2i-1}}{m^{2i}}\Pi_0, \quad \text{where,} \quad l = \frac{(1-m)^2}{m^2}.$$

The bandwidth is given by

$$\begin{aligned} BW &= \Pi_0 + 2(\Pi_1 + \Pi_2 + \ldots + \Pi_{N-1}) + \Pi_N \\ &= 2(\Pi_0 + \Pi_1 + \Pi_2 + \ldots + \Pi_{N-1} + \Pi_N) - \Pi_0 - \Pi_N \\ &= 2 - \Pi_0 - \Pi_N. \end{aligned}$$

Next the mean queue length and the mean waiting time of the system is computed in a similar manner as described in Sections 3.1 and 3.2. The mean queue length of the requests in the hot memory module is given from the state table as,

$$\begin{aligned} MQL_h &= N\Pi_0 + (N-1)\Pi_1 + (N-2)\Pi_2 + \ldots + \Pi_{N-1} \\ &= N\sum_{i=0}^{N-1}\Pi_i - \sum_{i=1}^{N-1} i\Pi_i \\ &= N(1-\Pi_N) - \frac{(1-m)}{m^2}\{\frac{1-l^{N-1}}{(1-l)^2} - \frac{(N-1)l^{N-1}}{1-l}\}\Pi_0, \end{aligned}$$

The mean queue length of the requests in the non-hot memory module is given from the state table as,

$$\begin{aligned} MQL_{nh} &= \Pi_1 + 2\Pi_2 + 3\Pi_3 + \ldots + N\Pi_N \\ &= \frac{(1-m)}{m^2}\{\frac{1-l^N}{(1-l)^2} - \frac{Nl^N}{1-l}\} \end{aligned}$$

From Little's law, $MWT_h = \frac{MQL_h}{m\lambda}$ and $MWT_{nh} = \frac{MQL_{nh}}{(1-m)\lambda}$. The mean waiting time for a memory request in the system is given below.

$$\begin{aligned} MWT_s &= m \times MWT_h + (1-m) \times MWT_{nh} \\ &= \frac{(MQL_h + MQL_{nh})}{\lambda}. \end{aligned}$$

The expression for the mean queue length of the system is, from the state table for the $N \times 2$ system and the limiting probabilities of the states, given by:

$$\begin{aligned} MQL_s &= N\Pi_0 + (N-1)\Pi_1 + (N-2)\Pi_2 + \ldots + \frac{N}{2}\Pi_{\frac{N}{2}} + (\frac{N}{2}+1)\Pi_{\frac{N}{2}+1} + \ldots + N\Pi_N \\ &= N\sum_{i=0}^{\frac{N}{2}}\Pi_i - \sum_{i=1}^{\frac{N}{2}} i\Pi_i + \frac{N}{2}\sum_{i=\frac{N}{2}+1}^{N}\Pi_i + \sum_{i=\frac{N}{2}+1}^{N} i\Pi_i, \end{aligned}$$

which after some manipulation gives the following closed form solution for MQL_s.

$$MQL_s = N - \frac{N}{2}\frac{(1-m)^{N+1}(1-l^{\frac{N}{2}})}{m^{N+2}(1-l)}\Pi_0 + \frac{(1-m)}{m^2}\{\frac{(1-m)^N}{m^N} - 1\}\{\frac{1-l^{\frac{N}{2}}}{(1-l)^2} - \frac{\frac{N}{2}l^{\frac{N}{2}}}{(1-l)}\}\Pi_0.$$

The bandwidths are evaluated for various values of m and N. As expected, the bandwidth value deteriorated as the value of m is increased, due to the fact that an increase in m results in progressive accumulation of the memory requests in the hot memory module, which remain unserviced due to the lack of memory modules, thereby decreasing the effective bandwidth. The results are shown in Table 3.10. For the same values of the input parameters, the mean queue lengths are depicted in Figure 3.6.

Table 3.10: **Memory bandwidths with various values of m and N for the $N \times 2$ system**

m	$N = 2$	$N = 5$	$N = 10$	$N = 15$	$N = 20$
.55	1.19	1.47	1.58	1.59	1.59
.65	1.32	1.44	1.45	1.45	1.45
.75	1.27	1.30	1.30	1.30	1.30
.85	1.16	1.17	1.17	1.17	1.17
.95	1.05	1.05	1.05	1.05	1.05

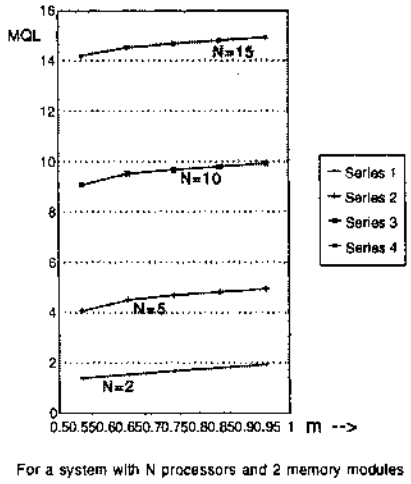


Figure 3.6: Mean queue length with probabilty of requesting hot modules

3.5. Conclusion

In this chapter we developed a discrete Markov model for analyzing memory interference in a $2 \times M$ and $N \times 2$ multiprocessor system. A step by step approach is followed in developing the model. First, a Markov model is developed for a system with one hot-spot and no favorite memories, which was later extended to a system with K hot-spots. Next, the model for a system consisting of one hot and multiple

favorite memories is developed. The probabilistic analysis of bandwidth, mean queue length and mean waiting time are done in each case and the results are shown either in tabular form or in the form of graphs.

CHAPTER 4

SIMULATION OF PROCESSOR MEMORY INTERCONNECTION SYSTEM

To analyze the general case of an $N \times M$ system with K hot-spots and favorite memories, the more common practise of simulation is resorted to. The number of states in the Markov model become prohibitively large for $M, N \geq 3$ and so it is almost impossible to derive an analytical model for such a system [1].

4.1. Simulation Environment

The sequential simulation of the processor-memory interconnection is carried out for a thousand simulation cycles. The simulation program has two main parts. The first part simulates the activity of the processors *e.g.* generating memory requests, which again can be directed to three distinct types of memory modules, namely, hot memories, favorite memories, and non-hot, non-favorite memories. The second part deals with the simulation of the activities of the memory modules, *e.g.* selecting the next processor request from the buffer if the current request is completed, otherwise continue serving the current request. The various input parameters to the simulation program are, memory service rate μ, memory request arrival rate λ, hot memory request rate m, number of memory modules M, number of hot modules K, and number of processors N. The simulation results are also used to validate the analytic results from the special case we considered, namely, $2 \times M$ system with K hot-spots, with and without favorite memories. Simulation was carried out for the cases with favorite memories and without, separately. The results of the simulation are tabulated below for various values of the input parameters, namely, λ, μ, K, M, N and m. The results are seen to tally well with the analytical results in the special cases.

4.2. Simulation of $N \times M$ system with K hot and $M - K$ non-favorite memories

The effective memory bandwidths from the simulation experiment for the various values of the input parameters are tabulated below. Tables 4.1 and 4.2 are the simulation counter-parts of the analytic results displayed in tables 3.4 and 3.5 respectively.

Table 4.1: **Memory bandwidth for $K = 1, M = 10, N = 2$ and $m = 0.75$.**

λ / μ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	.75									
0.2	.55	.84								
0.3	.54	.92	.97							
0.4	.40	.65	1.01	1.33						
0.5	.36	.79	.81	1.06	1.11					
0.6	.30	.60	.85	.89	1.19	1.18				
0.7	.24	.37	.71	.87	1.01	1.18	1.16			
0.8	.12	.39	.67	.93	.81	1.02	1.13	1.25		
0.9	.26	.36	.61	.78	1.01	1.02	1.16	1.26	1.23	
1.0	.14	.36	.62	.70	.80	1.02	1.16	1.23	1.23	1.26

Table 4.2: **Memory bandwidth for $K = 5, M = 10, N = 2$ and $m = 0.75$.**

λ / μ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	1.22									
0.2	.61	1.07								
0.3	.73	.81	1.36							
0.4	.45	.76	1.05	1.26						
0.5	.37	.69	.80	1.19	1.16					
0.6	.19	.70	.74	1.09	1.27	1.32				
0.7	.34	.37	.83	1.02	1.02	1.21	1.53			
0.8	.22	.43	.61	.82	1.07	1.23	1.47	1.46		
0.9	.35	.46	.52	.87	1.15	1.22	1.41	1.49	1.67	
1.0	.15	.55	.61	.84	.83	1.05	1.34	1.62	1.70	1.84

Table 4.3: **Memory bandwidth for $K = 20, M = 50, N = 30$ and $m = 0.75$.**

λ / μ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	9.94									
0.2	8.87	11.78								
0.3	7.31	11.03	12.85							
0.4	6.11	10.14	12.23	13.39						
0.5	5.67	8.96	10.78	12.88	13.30					
0.6	4.66	8.21	10.32	12.80	13.38	13.83				
0.7	4.15	7.39	9.99	11.83	13.28	14.83	14.36			
0.8	3.65	6.79	9.20	11.65	12.84	13.94	14.71	15.82		
0.9	3.45	6.25	9.39	10.88	12.86	14.76	14.77	16.18	15.9	
1.0	3.24	5.70	8.59	11.04	12.57	13.90	15.02	15.79	17.11	16.85

Table 4.4: **Memory bandwidth for $k = 75, M = 100, N = 60$ and $m = 0.75$.**

$\frac{\lambda}{\mu}$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	24.11									
0.2	19.56	26.59								
0.3	13.66	24.00	28.79							
0.4	12.71	20.34	26.15	30.41						
0.5	10.71	18.59	23.83	28.98	32.00					
0.6	9.31	16.54	21.86	26.87	30.42	32.39				
0.7	8.67	15.59	21.10	25.79	30.26	32.90	33.77			
0.8	6.97	14.18	19.20	24.49	29.42	32.09	35.57	36.18		
0.9	6.35	12.24	18.87	23.27	28.20	32.00	34.02	36.49	38.09	
1.0	5.73	12.00	17.62	22.92	26.51	30.22	33.84	36.80	39.34	40.03

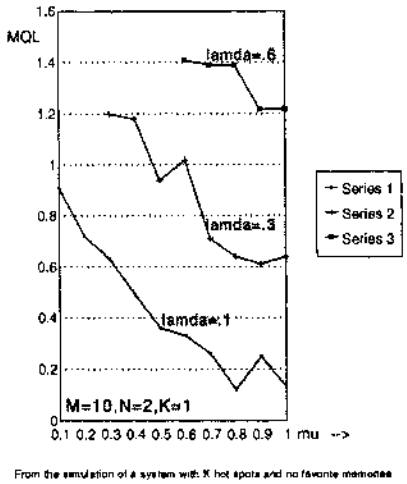


Figure 4.1: Variation of mean memory-queue length with request service rate

We note from the above results that the effective bandwidth of the system decreases with increasing service rate μ, the request arrival rate λ being kept constant. This has the same explanation as in the analytic case, that, increse in service rate with the arrival rate constant means a faster rate of processing without any increase in

input, which will correspondingly lead to lesser number of active processing elements which, in this case, are the memory modules. On the other hand, the bandwidth increases with increasing λ for constant μ because the increased request rate leads to an increase in the number of active memory modules. Figure 4.1 shows graphically the variation of mean queue length with μ for constant λ. The variation shows the same pattern as in the analytical case (Figure 3.3).

Keeping the values of λ and μ constant, the variation of bandwidth with K showed a definite pattern. With $M = 50$, the variation of bandwidth with K for $N = 10, 20, 30$ and 40 is observed. The bandwidth of the system seems to increase at first almost linearly with K and then at a progressively lower rate, until it almost saturates with the variation of K. When N is kept constant, and the variation of bandwidth with K is observed for $M = 10, 20, 30$ and 40, it showed similar patterns and it also decreases substantially after saturation. The value of K at which the change of bandwidth is almost insignificant is called the saturation value of K. The bandwidths shown in Table 4.5 are for four particular values of N and M in a system consisting of fifty memory modules and fifty processors respectively. The values of λ and μ are assumed to be unity.

Table 4.5: **Memory bandwidths for various values of K, with $\lambda = \mu = 1$, using simulation**

K	$M = 50$				$N = 50$			
	$N = 10$	$N = 20$	$N = 30$	$N = 40$	$M = 10$	$M = 20$	$M = 30$	$M = 40$
1	1.37	1.30	1.42	1.36	1.47	1.49	1.46	1.46
2	2.76	2.83	2.84	2.95	2.81	2.88	2.84	2.84
3	3.43	3.90	3.96	4.19	4.18	4.07	4.19	4.34
4	4.77	5.08	5.17	5.33	4.96	5.40	5.36	5.49
5	5.48	5.58	6.71	6.40	6.51	6.47	6.61	6.88
6	5.62	7.49	7.65	7.82	7.11	7.38	7.84	7.48
7	6.45	8.12	7.95	8.80	7.09	8.92	9.05	9.07
8	6.84	9.14	9.24	9.91	7.08	9.52	10.07	10.33
9	6.96	9.36	9.96	10.84	5.05	9.63	10.48	10.98
10	7.26	10.34	11.30	11.95		12.01	12.29	12.37
11	7.50	10.31	12.06	12.21		12.33	12.69	12.47
12	7.62	10.83	12.85	13.08		12.02	13.39	13.52
13	7.49	12.04	12.81	14.99		12.79	14.13	15.47
14	8.02	12.03	13.86	14.73		14.14	14.34	16.09
15	8.18	13.02	15.08	15.40		13.57	16.67	16.92
16	8.06	12.84	15.38	16.96		13.95	16.58	16.77
17	8.17	13.46	15.90	17.92		12.18	17.35	16.71
18	8.63	13.76	16.62	17.90		8.80	18.11	16.86
19	8.39	13.71	16.96	17.56		5.37	17.85	17.27
20	8.54	13.78	16.79	18.43			17.77	21.35
21	8.34	14.35	16.90	18.35			18.44	21.60
22	8.27	14.86	17.37	19.17			17.87	21.20
23	8.73	14.26	18.35	20.75			18.70	21.46
24	8.49	14.95	18.23	22.12			19.09	22.48
25	9.05	15.68	19.66	22.75			18.29	22.27
26	8.96	15.31	19.88	21.87			14.75	22.39
27	8.91	15.45	19.96	23.17			12.88	22.39
28	8.77	14.96	20.32	23.41			9.36	22.3
29	9.05	15.20	20.13	23.59			4.71	21.99
30	8.94	15.64	19.82	22.92				22.32
31	8.89	15.47	20.52	23.62				23.48
32	9.09	15.82	20.29	23.71				24.43
33	9.00	15.92	20.60	23.74				22.28
34	8.88	15.74	20.03	23.37				21.43
35	9.33	15.54	21.20	23.59				18.91
36	9.05	15.65	21.03	23.92				16.37
37	8.84	15.71	20.40	24.36				12.45
38	9.04	15.78	20.91	24.75				9.33
39	9.15	16.03	19.78	24.68				5.28
40	9.10	15.82	20.44	24.29				

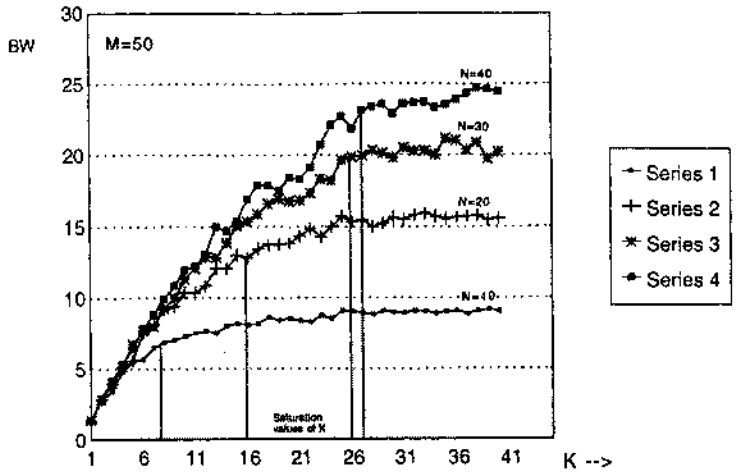


Figure 4.2: Variation of memory bandwidth with number of hot modules

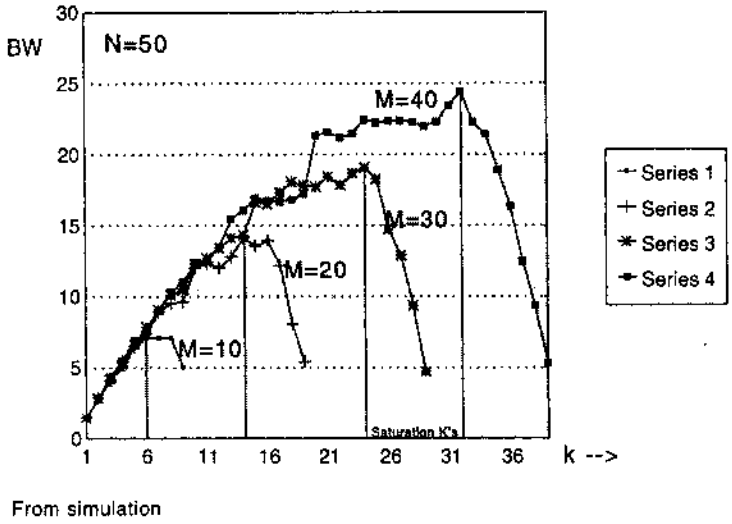


Figure 4.3: Variation of memory bandwidth with number of hot modules

From the simulation results, the values of K at which the rate of change of band-

width is very low are shown in the graphs (Figures 4.2 and 4.3). In the case where M is constant ($M = 50$), it appears that at $K = 8$ for $N = 10$, $K = 16$ for $N = 20$, $K = 26$ for $N = 30$ and $K = 27$ for $N = 40$, the bandwidths saturate. In the other case, for constant $N = 50$, it appears that at $K = 6$ for $M = 10$, $K = 14$ for $M = 20$, $K = 24$ for $M = 30$ and $K = 32$ for $M = 40$, the bandwidth saturates. These results will validate our probabilistic analysis in Section 5.

We also notice a sharp decrease in the bandwidth value immediately after it saturates. It can be intuitively explained as follows. As K is increased, the number of non-hot memory modules decreases. The decrease in bandwidth can be attributed to piling up of requests in the non-hot memory modules. Since the rate of request arrival in non-hot modules is less than that of hot modules, the effect is delayed. However, if $m \approx .5$, it is expected that the bandwidth curve will be more symmetrical, with the bandwidth reaching a maximum value at $K \approx \frac{M}{2}$.

4.3. Simulation of $N \times M$ system with K hot spots and favorite memories

The effective memory bandwidths from the simulation experiment for the various values of the input parameters are tabulated. Tables 4.6 and 4.7 are the simulation counter-parts of the analytic results displayed in tables 3.8 and 3.9 respectively.

Table 4.6: **Memory bandwidth for various λ and μ, $k = 1, M = 10, N = 2, m = 0.75$ and $\alpha = 0.6$.**

λ / μ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	.75									
0.2	.60	.87								
0.3	.43	.87	1.14							
0.4	.46	.66	.93	1.10						
0.5	.44	.63	.80	1.02	1.19					
0.6	.35	.50	.80	1.00	1.11	1.24				
0.7	.1	.52	.57	.85	1.06	1.14	1.17			
0.8	.23	.53	.56	.88	1.00	1.05	1.14	1.17		
0.9	.22	.44	.67	.84	.93	1.04	1.25	1.20	1.24	
1.0	.22	.35	.56	.72	.97	1.11	1.15	1.26	1.24	1.30

Table 4.7: **Memory bandwidth for various λ and μ, $K = 1, M = 10, N = 2, m = 0.75$ and $\alpha = 0.9$.**

λ / μ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	.75									
0.2	.53	.86								
0.3	.43	.73	.98							
0.4	.47	.71	1.00	1.18						
0.5	.38	.77	.84	1.03	1.09					
0.6	.34	.57	.91	.89	1.14	1.17				
0.7	.23	.45	.67	.86	.99	1.22	1.13			
0.8	.10	.43	.70	.84	.83	1.00	1.13	1.27		
0.9	.26	.36	.62	.85	1.05	1.11	1.13	1.26	1.3	
1.0	.21	.34	.63	.69	.93	1.02	1.19	1.20	1.26	1.29

Table 4.8: **Memory bandwidth for various λ and μ, $K = 20, M = 50, N = 30, m = 0.75$ and $\alpha = 0.9$.**

λ / μ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	5.92									
0.2	5.65	7.56								
0.3	5.85	9.3	8.14							
0.4	6.37	8.86	10.55	11.14						
0.5	5.01	8.40	11.26	12.86	13.07					
0.6	4.81	8.10	10.58	12.13	13.39	14.64				
0.7	3.86	7.73	10.11	12.03	13.54	13.88	14.43			
0.8	3.49	7.08	9.41	11.35	13.16	14.08	15.13	15.78		
0.9	2.97	6.53	9.06	11.90	12.50	14.45	14.35	15.48	16.40	
1.0	3.01	5.83	8.18	10.80	13.07	14.10	15.35	15.70	16.47	17.21

We see that, if the probability of accessing the favorite module is increased for a processor, all other conditions remaining the same, the bandwidth seems to decrease in most cases. This pattern is also seen in our analytic results, and can be explained by the fact that increasing α leads to an increase in the request rate to the same memory module, thereby leading to a piling up of requests in the memory queues. This leads to a decrease in the average number of active modules thereby decreasing the bandwidth. In fact, comparing tables 13 and 18, which give the bandwidth values for a general system configuration ($M = 50, N = 30, K = 20$) without favorite memories and with favorite memories respectively, we can draw the same conclusion.

4.4. Conclusion

In this chapter, we have described in moderate detail the implementation of the simulation environment for an $N \times M$ multiprocessor system and compared some of the results with the analytical results obtained for a $2 \times M$ and $N \times 2$ system. First, the simulation studies for an $N \times M$ system with K hot and $M - K$ non-favorite

memories are carried out, and then the experiments are extended to include favorite memories as well. In the first case, the variation of memory bandwidth with the number of hot modules, K, is also studied for various values of N and M. In all cases the bandwidth value seems to saturate after a certain value of K less than $min(N, M)$.

CHAPTER 5

PROBABILISTIC MODEL WITH ONLY HOT SPOTS

Consider for the moment the case where intra-cycle uniformity of memory requests is assumed. In that case we assume that once a processor's memory request is failed due to memory interference, it is altogether discarded in that cycle or the next. To compute the bandwidth in such a case, we proceed as follows.

Let BW^h and BW^{nh} denote the average number of active hot memory units and active non-hot, non-favorite memory modules respectively. In other words, BW^h denotes the contribution to the effective bandwidth from the hot memory modules, and BW^{nh} gives the contribution from non-hot memories. If K is the number of hot memory modules in the system, then according to [1], $BW^h = K\left[1 - \left(1 - \frac{\lambda m}{K}\right)^N\right]$, where $\frac{\lambda m}{K}$ gives the probability of requesting a particular hot memory module. Similarly, for the non-hot memory modules, $BW^{nh} = (M-K)\left[1 - \left(1 - \frac{\lambda(1-m)}{M-K}\right)^N\right]$. Hence the effective memory bandwidth for the entire system is

$$BW_a = K\left[1 - \left(1 - \frac{\lambda m}{K}\right)^N\right] + (M-K)\left[1 - \left(1 - \frac{\lambda(1-m)}{M-K}\right)^N\right].$$

But this is an approximate analysis of the bandwidth as in a more realistic situation the rejected memory requests are not discarded, but stored in order to make a fresh attempt in the next cycle. We will consider this case next following an iterative technique laid down by Mudge [20] to compute the effective bandwidth of the system. Let us assume that the rejected memory requests are resubmitted in the next cycle following a uniform distribution. This assumption over-estimates the bandwidth, because multiple rejected requests to the same memory module will be evenly distributed to all memory modules in the next cycle. Still, it gives a better estimate of the bandwidth than that given by equation .

Let β_h and β_{nh} be the modified request rates for the hot and non-hot memory modules. Following the scheme laid down in [20], we get the following iterative equation to evaluate the values of β's

$$\beta_{i+1}^{-1} = 1 + \frac{BW^*(\beta_i)}{Np^2}(1-p)$$

where BW^* denotes either $(i)BW^h$ or $(ii)BW^{nh}$ and p represents λm and $\lambda(1-m)$ for case (i) and (ii) respectively, and $BW^*(\beta_i)$ is obtained by replacing p by β_h or β_{nh} as the case may be. Solution for β_{i+1} when substituted in equation , gives a new value of bandwidth which is much more closer to that in the real situation.

Our simulation results (Table 4.5) reveal that for given values of M (the number of memory modules), N (the number of processors) and assuming λ and μ to be one, the memory bandwidth increases almost linearly with K (the number of hot spots) for low values of K. As the value of K continues to increase, the rate of increase of bandwidth decreases, and after some critical value of K, the bandwidth doesn't increase at all. We now propose a method to find out approximately this critical value of K for given M and N. The bandwidth evaluated by the preceding iterative scheme gives the closest possible approximation of the actual bandwidth, the only difference is due to the fact that, a rejected memory request instead of being repeated to the same memory module in the next cycle is uniformly distributed among all the memory modules. Still the analytical results seem to be close to the simulation results. Using this scheme, we propose the following heuristic to evaluate an approximate value of K beyond which we expect to get very little improvement on the bandwidth.

Heuristic :

Step 1: Initialize K to 1.

Step 2: Set M, N and m to given values; set λ and μ to 1,
and start with an initial estimate of β, in the range $0 < \beta < 1$.

Step 3: Apply a non-linear equation solver to evaluate β
using new estimates of BW (from equation) in successive stages.

Step 4: Use this β to get the final estimate of BW.

Step 5: Check the relative rate of change of bandwidth.
If the $rate > \epsilon$ (ϵ is a predefined small value), then increment K
and go to step 2. Else print the current value of K.

The proposed heuristic was implemented using a Fortran non-linear equation solver package and estimated the saturation value of K beyond which there is little or no change of memory bandwidth. The results obtained were quite satisfactory. While the iterative scheme to evaluate the bandwidth gave results which differed from those obtained from actual simulation to a small extent, the basic trend of the variations was remarkably similar. This is apparent from comparision of Tables 4.5 and 5.1. The variations are also depicted graphically (Figures 5.1, 5.2) for comparision. In all cases, the rate of change of the bandwidth progressively decreases for higher values of K. Our analytical model gives good estimates of memory bandwidths, although it tends to over-estimate bandwidths more for lower values of K than for higher values. Our analytical model also captures the decrease in bandwidth after saturation, for increasing K, although the decrease is not as substantial as found in the experiments.

Applying the heuristic, the saturation values of K are found to be $K = 8, 18, 26$ and 32, respectively for $N = 10, 20, 30$ and 40 and $M = 50$. When N is kept constant at 50, the saturation values of K are $6, 14, 22$ and 30, respectively for $M = 10, 20, 30$ and 40. Compare these results with the values of K obtained from the experiments under similar conditions (same values of the other parameters), which are $6, 16, 26$, 27 and $8, 16, 24$, and 32, respectively, in either case.

Table 5.1: **Memory bandwidths with various values of K, with $\lambda = \mu = 1$, using probabilistic model**

K	$M = 50$				$N = 50$			
	$N = 10$	$N = 20$	$N = 30$	$N = 40$	$M = 10$	$M = 20$	$M = 30$	$M = 40$
1	3.46	8.18	5.86	10.43	8.75	11.19	11.98	12.37
2	4.46	9.17	6.85	11.41	9.22	12.06	12.93	13.34
3	5.37	10.16	7.85	12.40	9.59	12.92	13.87	14.31
4	6.11	11.15	8.81	13.39	9.83	13.76	14.81	15.28
5	6.67	12.14	9.75	14.37	9.95	14.59	15.74	16.24
6	7.10	13.10	10.55	15.35	9.99	15.38	16.67	17.20
7	7.44	14.02	11.35	16.32	9.99	16.14	17.59	18.16
8	7.71	14.87	11.98	17.27	9.98	16.86	18.49	19.11
9	7.93	15.71	12.60	18.19	9.96	17.52	19.37	20.04
10	8.11	16.46	13.10	19.07		18.11	20.23	20.95
11	8.26	17.17	13.56	19.91		18.61	21.05	21.84
12	8.39	17.78	13.94	20.70		18.99	21.83	22.69
13	8.50	18.39	14.31	21.45		19.23	22.56	23.51
14	8.59	18.91	14.61	22.16		19.31	23.35	24.29
15	8.68	19.42	14.90	22.82		19.24	23.87	25.03
16	8.75	19.87	15.14	23.43		19.05	24.44	25.73
17	8.81	20.28	15.38	24.01		18.79	24.95	26.39
18	8.87	20.67	15.59	24.54		18.49	25.38	27.00
19	8.92	21.00	15.77	25.04		18.16	25.73	27.57
20	8.97	21.33	15.90	25.51			25.99	28.10
21	9.01	21.62	16.09	25.94			26.15	28.59
22	9.05	21.89	16.23	26.35			26.20	29.04
23	9.09	22.14	16.36	26.72			26.11	29.44
24	9.12	22.37	16.48	27.07			25.87	29.80
25	9.15	22.58	16.58	27.39			25.48	30.11
26	9.17	22.77	16.68	27.69			24.98	30.38
27	9.20	22.95	16.77	27.97			24.43	30.59
28	9.22	23.12	16.86	28.22			23.86	30.74
29	9.24	23.27	16.93	28.45			23.26	30.83
30	9.26	23.41	17.0	28.67				30.85
31	9.27	23.53	17.06	28.86				30.77
32	9.29	23.64	17.11	29.03				30.60
33	9.30	23.72	17.16	29.17				30.30
34	9.31	23.82	17.20	29.30				29.86
35	9.32	23.88	17.24	29.40				29.29
36	9.33	23.94	17.27	29.47				28.62
37	9.34	23.97	17.29	29.52				27.91
38	9.34	23.99	17.30	29.53				27.18
39	9.34	23.98	17.30	29.51				26.44
40	9.34	23.94	17.28	29.43				

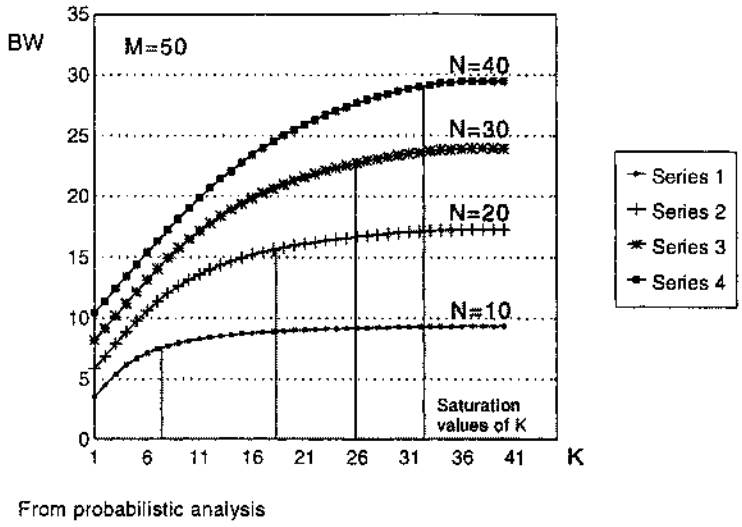


Figure 5.1: Variation of memory bandwidth with number of hot modules

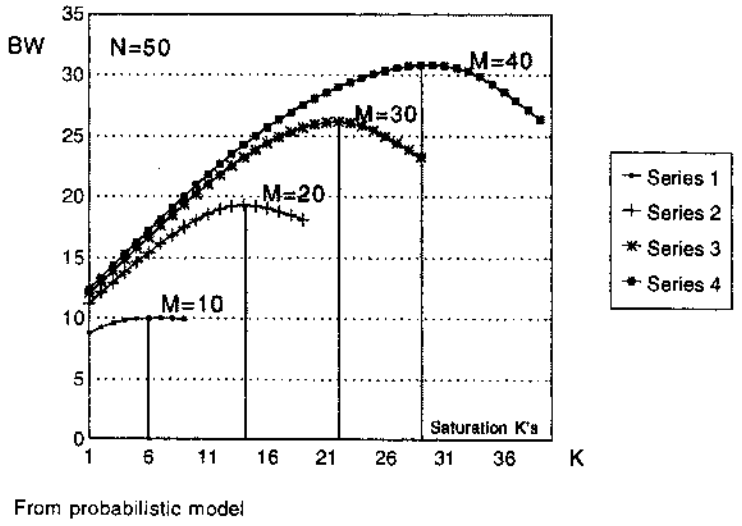


Figure 5.2: Variation of memory bandwidth with number of hot modules

5.1. Conclusion

In this chapter, we have developed an approximate probabilistic model for memory interference in an $N \times M$ multiprocessor system with K hot-spots. The iterative method formulated to give a good estimate of the bandwidth is used in the heuristic presented to estimate the saturation value of K beyond which the memory bandwidth is expected not to show any significant increase. The heuristic is implemented using a standard Fortran non-linear equation solver, and the results obtained are compared with that obtained from simulation experiments. Both results are shown to agree very well.

CHAPTER 6

CONCLUSIONS

In this thesis, we have presented an analysis of memory interference in multiprocessor systems consisting of various types of non-uniformities in memory access patterns. A discrete Markov chain is used to model the memory reference pattern in a set of memory modules constituting of hot spots, favorite memories or neither of these two, depending on the type of access non-uniformity. A step-by-step approach is taken to develop the model. First, a system consisting of one hot spot and no favorite memory modules is considered, which is then extended to K hot spots. Next, the system consisting of hot spot and favorite memories is developed. The last case considered is the system with N processors and two memory modules, one of them being a hot spot. Analytic expressions for bandwidth are derived for the very first and the last cases, while a linear sytem solver is used to solve the state transition equations in the others, of the four cases considered. For the general case of the $N \times M$ system, the more common practise of simulation is resorted to. Simulation is carried out for systems with hot spots only and systems with both hot spots and favorite memories. The simulation results are found to tally well with our analytic results in the special cases. A peculiar behavior of the system performance was observed when the bandwidth was computed from the simulation gradually increasing the number of hot memory modules. The bandwidth at first increased almost linearly with K, but saturated after some optimum value of K. A heuristic is proposed to evaluate the optimum value of K using an approximate probabilistic model of the memory interference. The model is approximate because it over-estimates bandwidth. However the value of K obtained from that heuristic seems to be a good approximation of the optimum value of K.

We have seen that the performance of the processor memory interconnection sys-

tem actually improves with the increase in the number of hot memory modules, and presented a method of finding the upper bound on this number beyond which the improvemnent virtually stops. It might be concluded that in a system with hot spots, distributing the shared variables in more than one memory modues is a good idea, as long as the number of newly added modules is below the saturation value of K. An interesting problem might be to look into a particular interconnection mechanism (*e.g* a bus based system), and analyse its performance in presence of all the various types of memory reference non-uniformity.

REFERENCES

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

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

[3] J. H. Patel, "Performance of processor-memory interconnections for multiprocessors," *IEEE Trans. Comput.*, vol c-30, pp. 771-780, Oct. 1981.

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

[5] C. E. Skinner and J. R. Asher, "Effects of storage contention on system performance," *IBM Syst. Journal*, vol 8, pp. 319-333, 1969.

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

[7] David, W. I. 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.

[8] L. Bhuyan, "Analysis of processor-memory interconnection networks," *IEEE Trans. Comput.*, vol c-34, pp. 279-283, March 1985.

[9] M. Kumar and J. R. Jump, "Performance of unbuffered shuffle-exchange networks," *IEEE Trans. Comput.*, vol c-35, pp. 573-578, June 1986.

[10] Y. C. Jenq, "Performance analysis of a packet switch based on single buffered banyan network," *IEEE J. Select. Areas Commun.*, vol SAC-3, pp. 1014-1021, Dec. 1983.

[11] D. M. Dias and J. R. Jump, "Analysis and simulation of buffered delta networks," *IEEE Trans. Comput.*, vol c-30, pp. 273-282, April 1981.

[12] H. S. Yoon, K. Y. Lee and M. T. Liu, "Performance analysis of multibuffered packet-switching networks in multiprocessor systems," *IEEE Trans. Comput.*, vol c-39, pp. 319-327, March 1990.

[13] S. H. Hsiao and C. Y. R. Chen, "Performance analysis of single buffered multistage interconnection networks," in *Proc. Third IEEE Symp. Parallel and Distributed Processing*, Dec. 1991, pp. 864-867.

[14] T. H. Theimer, E. P. Rathgeb, and M. N. Huber, "Performance analysis of buffered banyan networks," *IEEE Trans. Commun.*, vol c-39, pp. 269-277, Feb. 1991.

[15] Y. Mun and H. Y. Youn, "Performance analysis of finite buffered multistage interconnection networks," *IEEE Trans. Comput.*, vol 43, pp. 153-162, Feb. 1994.

[16] G. F. Pfister and V. A. Norton, "Hot Spot contention and combining in multistage interconnection networks," *IEEE Trans. Comput.*, vol c-34, pp. 943-948, Oct. 1985.

[17] G. Lee, C. P. Kruskal, and D. J. Kuck, "On the effectiveness of combining in resolving Hot Spot contention," *Journal of Parallel and Distributed Computing*, vol 20, pp. 136-144, 1994.

[18] H. S. Kim and A. L. Garcia, "Performance of buffered banyan networks under non-uniform traffic patterns," *IEEE Trans. Commun.*, vol 38, pp. 648-656, May 1990.

[19] T. Lang, M. Valero and I. Alegre, "Bandwidth of crossbar and multiple bus connections for multiprocessors," *IEEE Trans. Comput.*, vol c-31, pp. 1227-1223, Dec. 1982.

[20] T. N. Mudge, J. P. Hayes, G. D. Buzzard, and D. C. Winsor, "Analysis of multiple-bus interconnection networks," *Journal of Parallel and Distributed Computing*, vol 3, pp. 328-343, 1986.

[21] T. N. Mudge and H. B. Al-Sadoun, "A semi-Markov model for the performance of multiple-bus systems," *IEEE Trans. Comput.*, vol c-34, pp. 934-942, Oct. 1987.

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

[23] C. R. Das and L. Bhuyan, "Bandwidth availability of multiple-bus multiprocessors," *IEEE Trans. Comput.*, vol c-34, pp. 918-926, Oct. 1985.

[24] S. M. Mahmud, "Performance analysis of multilevel bus networks for hierarchical multiprocessors," *IEEE Trans. Comput.*, vol 43, pp. 789-805, July 1994.

[25] W-T Chen and J-P Sheu, "Performance analysis of multiple bus interconnection networks with hierarchical requesting model," *IEEE Trans. Comput.*, vol 40, pp. 834-842, July 1991.

[26] M. A. Holliday and M. K. Vernon, "Exact performance estimates for multiprocessor memory and bus interference," *IEEE Trans. Comput.*, vol c-36, pp. 76-85, Jan. 1987.

[27] C. P. Kruskal and M. Snir, "The performance of multistage interconnection networks for multiprocessors," *IEEE Trans. Comput.*, vol c-32, pp. 1091-1098, Dec. 1983.

[28] C. V. Ravi, "On the bandwidth and interference in interleaved memory systems," *IEEE Trans. Comput.*, vol c-21, pp. 899-901, Aug. 1972.

[29] F.A. Briggs and E. S. Davidson, "Organisation of semiconductor memories for parallel pipelined processors," *IEEE Trans. Comput.*, vol c-26, pp. 162-169, Feb. 1977.