

**Computer Science Department Technical Report  
University of California  
Los Angeles, CA 90024-1596**

**PERFORMANCE ANALYSIS OF FINITE-BUFFERED  
MULTISTAGE INTERCONNECTION NETWORKS WITH  
VARIOUS SWITCHING ARCHITECTURES**

**Tzung-I Lin**

**December 1990  
CSD-900049**



UNIVERSITY OF CALIFORNIA

Los Angeles

Performance Analysis of Finite-Buffered Multistage Interconnection Networks  
with Various Switching Architectures

A dissertation submitted in partial satisfaction of the  
requirements for the degree Doctor of Philosophy  
in Computer Science

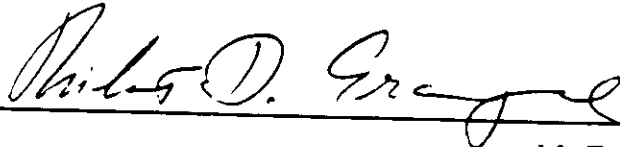
by

Tzung-I Lin

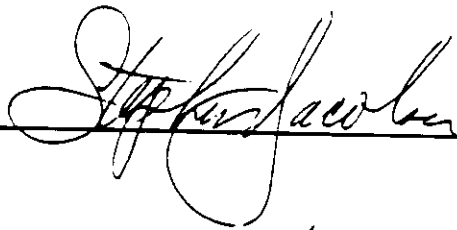
1990

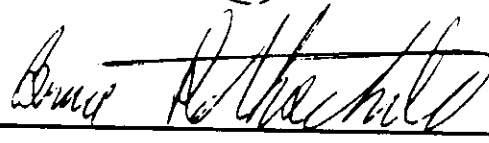


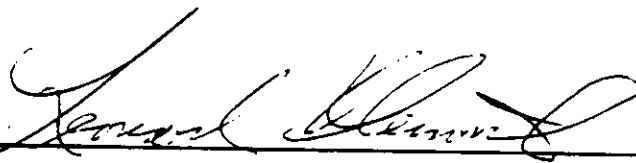
The dissertation of Tzung-I Lin is approved.

  
M. Ercegovac

  
M. Gerla

  
S. Jacobsen

  
B. Rothschild

  
Leonard Kleinrock, Committee Chair

University of California, Los Angeles

1990

To Mom and Dad, and all my family.

## TABLE OF CONTENTS

<b>1</b>	Introduction . . . . .	1
<b>2</b>	<b>Modelling of Processing Elements with Identical Traffic Patterns</b> . . . . .	<b>8</b>
2.1	Basic Models with a Hot Spot Traffic Pattern . . . . .	8
2.1.1	Architecture Description and Assumptions . . . . .	8
2.1.2	The Routing Model . . . . .	12
2.1.3	Analysis of the Unbuffered MIN . . . . .	12
2.1.4	Analysis of the Buffered MIN . . . . .	20
2.1.5	Model Verification . . . . .	40
2.1.6	Conclusion . . . . .	42
2.2	General Traffic Pattern Model . . . . .	43
2.2.1	Introduction . . . . .	43
2.2.2	Transformation Method . . . . .	44
<b>3</b>	<b>Modeling of Processing Elements with Different Traffic Patterns</b> 50	<b>50</b>
3.1	Basic Model . . . . .	50
3.1.1	Problem Characteristics . . . . .	50
3.1.2	Superposition Method . . . . .	53
3.2	Not Uniform Traffic Spots model . . . . .	57
3.2.1	Introduction . . . . .	57
3.2.2	Modelling Approach . . . . .	58
3.2.3	Model Results and Verifications . . . . .	63
3.3	Different Input Rate Model . . . . .	66
<b>4</b>	<b>Persistent Blocking Model</b> . . . . .	<b>69</b>
4.1	Modelling Approach . . . . .	69
4.2	Model Results . . . . .	73
4.3	Model Limitation . . . . .	79
<b>5</b>	<b>Re-submission Model</b> . . . . .	<b>80</b>
5.1	Modelling Approach . . . . .	81
5.2	Model Results . . . . .	82
5.3	Extended Model . . . . .	85
5.3.1	Rate Adjusted Model . . . . .	85
5.3.2	Maximal Input Rate for a Given Loss Probability . . . . .	89

<b>6</b>	<b>Delay Model Analysis</b>	<b>91</b>
6.1	The Turn Back Switch Model with a Uniform Traffic Pattern	92
6.1.1	Turn Back Model	93
6.1.2	Modeling Approach	95
6.1.3	Performance Measure	103
6.1.4	Solution Algorithm	103
6.1.5	Results	106
6.2	The Turn Back Switch Model with a Non-uniform Traffic Pattern	110
6.2.1	Modelling Approach	112
6.2.2	Solution Algorithm	118
6.2.3	Results	120
6.3	The Blocking Switch	122
6.3.1	Modelling Approach	122
6.3.2	Results	124
6.4	The Rotating Switch Model	126
<b>7</b>	<b>Summary and Future Research</b>	<b>130</b>
7.1	Summary	130
7.2	Future Research	133
	<b>References</b>	<b>135</b>



## LIST OF FIGURES

2.1	3 stage Banyan network with buffers at output ports of each switch	9
2.2	Probability of Acceptance for $q=0.1$ case . . . . .	16
2.3	The PA degradation percentage for $q=0.1$ case . . . . .	16
2.4	Probability of Acceptance for $q=1.0$ case . . . . .	17
2.5	The PA degradation percentage for $q=1.0$ case . . . . .	17
2.6	Markov chain of a queue $Q_{i,j}$ extracted from the network where the state variable represents the number of packets in that queue: $K=4$ . . . . .	23
2.7	Improving the Probability of Acceptance by adding buffers for interconnection networks with $q=0.1$ and a uniform traffic pattern	27
2.8	The PA improvement percentage by adding buffers for intercon- nection networks with $q=0.1$ and a uniform traffic pattern . . . . .	27
2.9	Improving the Probability of Acceptance by adding buffers for interconnection networks with $q=1.0$ and a uniform traffic pattern	28
2.10	The PA improvement percentage by adding buffers for intercon- nection networks with $q=1.0$ and a uniform traffic pattern . . . . .	28
2.11	Improving the Probability of Acceptance by adding buffers for interconnection networks with $q=0.1$ and a non-uniform traffic pattern ( $r=0.9$ ) . . . . .	29
2.12	The PA improvement percentage by adding buffers for intercon- nection networks with $q=0.1$ and a non-uniform traffic pattern ( $r=0.9$ ) . . . . .	29
2.13	Improving the Probability of Acceptance by adding buffers for interconnection networks with $q=1.0$ and a non-uniform traffic pattern ( $r=0.9$ ) . . . . .	30
2.14	The PA improvement percentage by adding buffers for intercon- nection networks with $q=1.0$ and a non-uniform traffic pattern ( $r=0.9$ ) . . . . .	30
2.15	The mean busy buffer size for a light traffic case ( $q=0.1$ ) with $K=8$	32
2.16	The mean busy buffer size for a moderate traffic case ( $q=0.5$ ) with $K=8$ . . . . .	32
2.17	The mean busy buffer size for a moderate-to-heavy traffic case ( $q=0.7$ ) with $K=8$ . . . . .	34
2.18	The mean busy buffer size for a heavy traffic case ( $q=1.0$ ) with $K=8$	34
2.19	The Tree Build-up Time diagrams . . . . .	36
2.20	The upper bound of the tree build-up time for a 9-stage, 8 buffered interconnection network . . . . .	39
2.21	The mean delay for a 9-stage, 8 buffered interconnection network	39

2.22	The analytical model vs. the simulation for a 9-stage, 8 buffered interconnection network. The dotted lines are simulations. . . . .	41
2.23	The analytical model vs. the simulation for a 2-stage, 8 buffered interconnection network . . . . .	41
2.24	A General memory referencing pattern shown in terms of accessing probabilities $A$ , . . . . .	46
3.1	(a) A 4x4 network with 4 switching points (•) in each stage. (b)-(f) routing probability matrix . . . . .	52
3.2	Markov chain of the queueing process where the state variable represents the number of packets in the queue . . . . .	59
3.3	Throughput-Delay performance from analytical model vs. simulator data . . . . .	65
3.4	Throughput comparison for a 64x64 Omega MIN under uniform traffic . . . . .	65
4.1	The states of a server during its busy period. . . . .	70
4.2	Comparison of results for a 6-stage, 4-buffered Banyan network with and without the "memory" behavior improvement . . . . .	74
4.3	Throughput comparison for a 4-buffered, 6 stage Omega MIN with a uniform traffic pattern . . . . .	75
4.4	Mean delay comparison for a 4-buffered, 6 stage Omega MIN with a uniform traffic pattern . . . . .	75
4.5	Throughput comparison for a 4-buffered, 6 stage Omega MIN with EFOS traffic pattern . . . . .	76
4.6	Mean delay comparison for a 4-buffered, 6 stage Omega MIN with EFOS traffic pattern . . . . .	76
4.7	Throughput comparison for an 8-buffered, 6 stage Omega MIN with a uniform traffic pattern . . . . .	77
4.8	Mean delay comparison for an 8-buffered, 6 stage Omega MIN with a uniform traffic pattern . . . . .	77
4.9	Throughput comparison for a 4-buffered, 10 stage Omega MIN with uniform traffic pattern . . . . .	78
4.10	Mean delay comparison for a 4-buffered, 10 stage Omega MIN with uniform traffic pattern . . . . .	78
5.1	Analytical model vs. simulation for a 5 stage Omega under EFOS traffic pattern . . . . .	83
5.2	Analytical model vs. simulation for a 5 stage Omega under EFOS traffic pattern . . . . .	83
5.3	Analytical model vs. simulation for a 6 stage Omega under uniform traffic pattern . . . . .	84
5.4	Analytical model vs. simulation for a 6 stage Omega under uniform traffic pattern . . . . .	84

5.3	Mean c-link length comparison for rate adjusted model . . . . .	86
5.6	Mean delay comparison for rate adjusted model . . . . .	86
5.7	Throughput-delay curve for rate adjusted model, 5 stage, EFOS pattern . . . . .	89
6.1	4x4 interconnection network with turn back switch as building block	94
6.2	Markov chain for (a) simultaneous operation (b) no simultaneous operation. . . . .	96
6.3	The recurrent relationship among variables in a particular path . . . . .	102
6.4	Analytical model vs. simulation for a 6-stage Omega . . . . .	107
6.5	Analytical model vs. simulation for a 10-stage Omega . . . . .	107
6.6	Analytical model vs. simulation for a 6 stage, 4 buffer Omega with infinite buffer at PE under uniform traffic pattern . . . . .	108
6.7	Analytical model vs. simulation for a 6 stage, 4 buffer Omega with infinite buffer at PE under uniform traffic pattern . . . . .	108
6.8	Feedback Model vs. Combined Rate Model . . . . .	109
6.9	Simultaneous model vs. non-simultaneous model for a 6 stage Omega . . . . .	111
6.10	Simultaneous model vs. non-simultaneous model for a 10 stage Omega . . . . .	111
6.11	The recurrent relationship among variables in a particular path . . . . .	113
6.12	The effect of feed back from different queues and different stages to PE 1 . . . . .	116
6.13	Analytical model vs. simulation for a 6 stage Omega under hot spot traffic pattern . . . . .	121
6.14	Analytical model vs. simulation for a 6 stage Omega under EFOS pattern . . . . .	121
6.15	Markov chain for a discrete time queue at PE . . . . .	123
6.16	Blocking switch vs. turn back switch for a 6 stage Omega under uniform traffic pattern . . . . .	125
6.17	A 2 stage 4x4 interconnection network with rotating switches . . . . .	127
6.18	Analytical result for a 6 stage rotating switches under uniform traffic pattern . . . . .	128

## ACKNOWLEDGEMENTS

I wish to thank Chun-Hsien Lu, Shiou-Pyn Shen and Asser Tantawi for their discussions and comments which helps to clarify some of the problems. I am also grateful to Li-Min Huang and Michelle Horng for their assistance in proofreading the dissertation. I would also like to thank Kong Li, Pinghann Wang, Lih-Hsing Ke, Chi-Cho Lin and many other friends for their four years companionship of studying aboard. Most importantly, I would like to thank my committee chair, Professor Leonard Kleinrock, and my family for their continuing support and encouragement.

## VITA

October 4, 1961	Born, Taipei, Taiwan
1984	B.S. National Taiwan University Department of Electrical Engineering
1988	M.S. University of California Los Angeles Computer Science Department

## PUBLICATIONS

- T. Lin, "An Analysis of State Restoration in Distributed Systems" Masters Thesis, UCLA Computer Science Department, 1988.
- T. Lin and A. Tantawi, "Performance Evaluation of Packet-Switched Multistage Interconnection Networks under a Model of Hot-Spot Traffic", *ORSA/TIMS Joint National Meeting*, Philadelphia, PA, October 29-31, 1990.
- T. Lin and L. Kleinrock, "Performance Analysis of Finite-Buffered Multistage Interconnection Networks with a General Traffic Pattern", submitted to *ACM SIGMETRICS 1991*

## ABSTRACT OF THE DISSERTATION

Performance Analysis of Finite-Buffered Multistage Interconnection Networks  
with Various Switching Architectures

by

Tzung-I Lin

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 1990

Professor Leonard Kleinrock, Chair

We present analytical models for evaluating the performance of finite-buffered packet switching multistage interconnection networks using blocking switches and turn back switches under any general traffic pattern. Most of the previous research work has assumed the case of no buffers, single buffer or infinite buffers, and all of them assumed that processing elements have the same traffic pattern, either a uniform traffic pattern or a specific hot spot pattern. However, those models cannot be applied very generally. There is a need for an analytical model to evaluate the system performance under general conditions. Our approach is to create a model which approximates these networks, derive the system equations for this model, and then solve these equations iteratively.

We first propose a decomposition and iteration model for a specific hot spot pattern. This model is then generalized to handle general traffic patterns using a

transformation method. A superposition method and a weighting factor are then proposed to be used with the iteration model and the transformation method to analyze the interconnection networks with a general traffic condition where each processing element has its own traffic pattern and input rate.

In order to account for the "memory" characteristic of a blocking switch which causes persistent blocking of packets contending for the same output ports, we propose an approximate method. Moreover, An analytical model is proposed to analyze a re-submission interconnection network where each processing element has a finite buffer space to accept the rejected packets from the network. A rate adjusted model is then proposed to reduce the time delay while maintaining the throughput at the same level.

Finally, we develop an analytical model for interconnection networks using the turn back switches. The delay performance of packets passing through a particular path is analyzed using recurrence equations. The performance of the turn back switches and blocking switches are compared. This leads us to propose the "rotating switch" which combines the advantages of the turn back switch and the blocking switch; we then evaluate its performance relative to the blocking switch and the turn back switch.





## CHAPTER 1

### Introduction

With the increasing demands for large computing power, parallel processing has attracted a significant amount of research interest in recent years. One of the problems we encounter with large parallel processing system is how to interconnect thousands of Processing Elements (PE) and Memory Modules (MM). Many interconnection networks are reviewed in [Sieg 79], [Feng 81] and [Ahma 89]. The Multistage Interconnection Network (MIN) was proposed as a cost effective alternative to the more powerful, but more expensive crossbar (a fully connected interconnection network). However, in the case of a non-uniform traffic pattern, the interconnection network performance degrades significantly due to contention for a favorite memory module (also known as the "hot spot"). Degradation due to a hot spot occurs even for the fully connected networks (e.g., crossbar). In the case of the multistage interconnection networks, the presence of a "hot spot" causes a saturated tree to be formed along all possible paths that lead to the hot spot. This phenomenon is called **tree saturation** [Pfis 85]. Tree saturation not only blocks packets destined for the favorite memory module, but also blocks packets destined for the other memory modules. Modeling and analysis is thus needed in order to understand the relationship between the traffic pattern and the system performance. In addition, these (synchronous) multistage interconnection networks have been proposed for applications to fast packet switching networks [Turn 86] and for applications to ATM (Asynchronous Transfer Mode) switches. The performance analysis of multistage interconnection networks thus

becomes an important issue.

A considerable amount of performance analysis has been done on clocked, packet-switched multistage interconnection networks. Most of the previous research was limited to unbuffered or infinite buffered cases with a uniform traffic pattern or a particular hot spot traffic pattern. An unbuffered MIN is an interconnection network where the switching elements do not have any buffer space for storing packets. A buffered MIN is an interconnection network with either finite buffers or infinite buffers in each switching element. According to the buffer size and the traffic pattern, they can be categorized as follows :

**Unbuffered MIN with a uniform traffic pattern :** Patel proposed an analytical model based on a recurrence equation [Pate 81]. For an interconnection network with  $k \times k$  switches (a  $k \times k$  switch is a  $k \times k$  crossbar switch), the probability that stage  $i$  is not empty,  $P_i$ , is given in terms of  $P_{i-1}$  and the switching element size  $k$  as follows :

$$P_i = 1 - \left(1 - P_{i-1} \cdot \frac{1}{k}\right)^k$$

$P_i$  was used to calculate the probability of acceptance (PA) and the throughput. Kruskal and Snir [Krus 83] solved for the asymptotic behavior of  $P_i$  (for  $i \gg 1$ ) in terms of the offered load  $q$  and the switching element size  $k$  :

$$P_i = \frac{2k}{(k-1) \cdot i + \frac{2k}{q}}$$

Although they also proposed an approximate formula for estimating the mean delay for an interconnection network with an infinite queue in each switching element, the simulation indicated a large discrepancy for a moderate traffic load.

**Buffered MIN with a uniform traffic pattern :** Dias and Jump [Dias 81] analyzed both the unbuffered and finite-buffered network. They employed a probabilistic approach to model the unbuffered case. A complicated iterative approach

was proposed for the finite-buffered case. They showed that the performance of a buffered MIN is comparable to the performance of a crossbar. In addition, they showed that the performance of a buffered MIN degrades slowly as the network size grows. Furthermore, little performance improvement is achieved when they added more than two buffers; i.e. two buffers are "enough". Jenq proposed an iterative model for a single buffered interconnection network [Jenq 83]. The single buffer was placed at the input port of the switching element. Yoon, Lee and Liu [Yoon 90] extended Jenq's model to analyze an interconnection network with multiple buffers. Szymanski and Shaikh [Szym 89] proposed an approximate Markov chain model for an interconnection network with finite buffered switching elements. These models all assume that different queues in the same switching element are independent. This independence assumption causes the analytical models to predict optimistic behavior. Kruskal, Snir and Weiss, [Krus 86] and [Krus 88], analyzed an interconnection network with an infinite buffer size in each switching element. They solved for the distribution of delay for the queue in the first stage as an M/G/1 queue. Then they estimated the distribution of delays for later stages. The queues in later stages were assumed to be identical.

**Unbuffered MIN with a non-uniform traffic pattern :** Bhuyan studied the performance of an interconnection network [Bhuy 85] with several non-uniform traffic patterns using a probabilistic approach.

**Buffered MIN with a non-uniform traffic pattern :** Kim and Garcia [Kim 90] proposed an analytical model for a single buffered MIN. Several non-uniform traffic patterns were analyzed. They claimed that the single buffered MIN could easily be extended to multiple buffers; however, they did not carry out this extension.

**Buffered MIN with a general traffic pattern :** Kurisaki and Lang

[Kuri 88] proposed a Not Uniform Traffic Spots (NUTS) traffic pattern. The processing elements were allowed to have different traffic patterns. The resulting overall traffic pattern may seem uniform, but causes congested spots inside the network. The performance was shown to be severely degraded. However, their approach was a simulator which is not suitable for a large sized network. Willick and Eager [Will 90] proposed an analytical model for an interconnection network with infinite buffer size and general traffic conditions. Their model is good for the uniform traffic pattern; it appears that it is not good for non-uniform traffic patterns. Their model is based on a Mean Value Analysis; it has the potential to analyze the case where each processing element has its own traffic pattern.

In general, we see that exact analytical models have been found for unbuffered interconnection networks. However, for finite buffered networks with a non-uniform traffic pattern, either a simulator or an iterative, approximate model was used to analyze the system performance. The interdependence of finite buffered queues with blocking makes an exact analysis very difficult. Our models also employ an iterative approach to analyze the finite-buffered interconnection networks. However, this iterative approach provides a general framework which can be easily extended to analyze various switching architectures with modifications incorporated in the approach.

We focus on either a Banyan interconnection network [Goke 73] or an Omega interconnection network [Lawr 75]. These networks and some other popular networks have been shown to be equivalent [Feng 81]. The basic building block of an interconnection network can be either a blocking switch or a turn back switch. The difference between a blocking switch and a turn back switch is the way they

handle the conflicting packets which contend for limited buffer space.

- The blocking switch at a given stage of the MIN will block an incoming packet to that stage when there is no buffer space for that packet. This packet then remains at the preceding stage and waits to try again in the next cycle. Feedback information is needed to notify the preceding stage that such a situation occurs. On receiving the blocking signal, the server in the preceding stage stops sending a packet until the next cycle.
- The turn back switch rejects the packet if this situation occurs, and re-submits it to the source (PE). Feedback information is not needed, and every switch outputs one packet per cycle if it is not empty.

We concentrate on the analysis of the blocking switch in Chapters 2 to 5. The modelling of the turn back switch is discussed in Chapter 6. The performance of these two switches is also compared in Chapter 6.

We begin by proposing an analytical model which approximates the traffic behavior in an interconnection network using blocking switches with a specific hot spot traffic pattern. All processing elements are assumed to have the same traffic pattern and input rate. An iterative approach is proposed in section 2.1 to solve for the throughput and delay performance of a finite-buffered MIN. A general traffic pattern model is proposed in section 2.2 in which a transformation method is incorporated in the basic analytical model. The method transforms a given traffic pattern into a set of routing probabilities which reflects the steady state behavior of packets flowing through the network.

In the real world, however, each processing element can have its own traffic pattern and input rate. This creates an even more general traffic pattern than the traffic pattern that we discussed in section 2.2. A superposition method is

proposed in section 3.1 to be incorporated in the basic analytical model to analyze an interconnection network whose processing elements have different traffic patterns. (The Not Uniform Traffic Spot (NUTS) traffic patterns [Lang 88] are analyzed in section 3.2 as examples.) A weighting factor is incorporated in the superposition method to handle the case where each processing element has its own input rate. However, the model result is shown to be optimistic compared to the simulation. The inherited "memory" behavior in the blocking switch causes persistent blocking which is not accounted for in the analytical model.

To account for the "memory" behavior of a blocking switch, we modify the renewal routing choice assumption in the modelling approach. An approximation method is proposed in Chapter 4 to account for the persistent blocking. The simulation comparison indicates that the approximation method is very good in capturing the persistent blocking effect for various traffic patterns. However, the approximation method still does not capture the whole effect of persistent blocking when the hot spot congestion is very severe.

In Chapter 5, we study a re-submission model which accounts for rejected traffic in the blocking switch. A finite number of buffers are placed at the processing elements to accommodate rejected packets (due to full buffers in the network). It is shown in section 5.3 that a rate adjusted model can reduce the mean delay without sacrificing the throughput. The maximal input rate which satisfies a given loss probability is calculated in section 5.3.2.

In Chapter 6, we study a new model, namely the turn back switch model. We first propose this turn back switch model for an interconnection network with a uniform traffic pattern in section 6.1. Different assumptions and different modelling approaches are discussed and compared. When the traffic pattern is not uniform, the traffic pattern of the re-submitted traffic may be different from

the given traffic pattern. Hence the simple modelling approach in section 6.1 is not sufficient. A different model is proposed for an interconnection network with a non-uniform traffic pattern in section 6.2. The model employs an algorithm which traces the re-submitted traffic in detail. We then change the traffic pattern to properly reflect the presence of the re-submitted traffic. In order to compare the blocking switch with the turn back switch, an infinite queue is added to the blocking switch model in section 6.3. The throughput-delay curves of both switches indicates that both switches have their advantages in a certain range of system load. This fact leads us to suggest a new configuration, namely the rotating switch, which combines the advantages of both switches. The throughput-delay curve of the rotating switch is obtained and is compared to those of the blocking switch and the turn back switch. The result shows that the rotating switch outperforms the other switches at all system loads.

A summary is given in Chapter 7 with some concluding remarks, as well as directions for future research.

## CHAPTER 2

### Modelling of Processing Elements with Identical Traffic Patterns

#### 2.1 Basic Models with a Hot Spot Traffic Pattern

In this section, we propose three models to evaluate the performance of the unbuffered MIN and the finite-buffered MIN with a hot spot traffic pattern. By changing a parameter of the system, the models are reduced to the uniform traffic case.

The architecture descriptions and the model assumptions are described in section 2.1.1. In section 2.1.2, the routing model is described. The unbuffered MIN is analyzed in section 2.1.3 with different hot spot traffic patterns. The asymptotic behavior for the unbuffered MIN with a hot spot traffic is also studied. The buffered MIN is analyzed using an approximate model in section 2.1.4. The probability of acceptance, the average number of busy buffers and the average time delay are evaluated. Our model of the buffered MIN is verified using simulation in section 2.1.5. Finally, concluding remarks are given in section 2.1.6.

##### 2.1.1 Architecture Description and Assumptions

In this paper, the interconnection network we consider is a clocked, packet-switched finite-buffered Banyan network where each  $2 \times 2$  switch has buffers of finite size  $K$  at its output ports (see Figure 2.1). There are  $N$  processing elements and  $N$  memory modules interconnected by the  $n$ -stage (i.e.  $N = 2^n$ ) interconnection network. All the operations of input and output take place at the



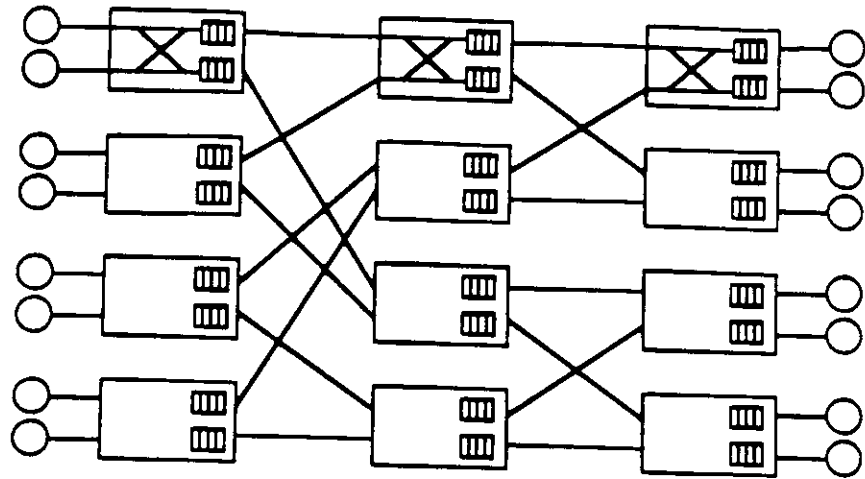


Figure 2.1: 3 stage Banyan network with buffers at output ports of each switch end of each cycle. The interconnection network accepts requests from the input nodes (processing elements), then routes them to the output nodes (memory modules). These requests will be returned from the output nodes through the interconnection network in the reverse direction to the original requesting nodes. The "forward" network and "backward" network are distinct, but are identical in topology. It is sufficient to discuss the delay and throughput performance of the forward network only.

Each packet generated at the processing elements carries an address tag with a number of bits equal to the number of stages of the interconnection network. The address tag is a binary representation of the destination address. The address tag is generated according to a destination distribution (traffic pattern). (Each processing element may have a different destination traffic pattern. If all the processing elements have an identical traffic pattern, then the overall traffic pattern for the memory modules is the same as this one. If each processing element has its own traffic pattern, then the overall destination traffic pattern for the memory modules is the superposition of the traffic patterns of the individual

processing elements.) The packet is then fed into the first stage of the network. The first stage switch examines the first bit of the address tag; if it is a 0, the packet is routed to the queue at the upper output port. If the first bit is a 1, the packet is routed to the queue at the lower output port (see Figure 2.1). The packet then waits in the queue until its turn to be transmitted to the next stage. The routing process is repeated in each stage, thus sending the packet to its destination.

A *blocking* switch is assumed in which if a head-of-queue packet cannot go to the next stage due either to a full buffer or to its inability to capture the single available buffer in this next stage, then it stays at the current queue and waits for the next cycle to try again; in either case, we refer to this as a contention failure. The blocking phenomenon has an implied memory characteristic in that a blocked packet will attempt to reach the same output port again. This memory characteristic makes analytical modelling difficult. We discuss an approximation method to be incorporated into the analytical model to model this memory characteristic in later sections. When two packets from different queues in the same stage contend for the same output queue in the next stage, a potential problem occurs. If there are more than two spaces available at this output queue, the switch is assumed to be fast enough to accept both packets in one cycle. If there is only one space available, a packet is randomly chosen to fill up this space; the other packet is then "blocked" (a contention failure) and stays at the original queue. However, if no space is available in the next stage, then both packets are blocked (again, a contention failure).

Packets are assumed to be of the same length (i.e. fixed size packets). A packet is generated by each processing element independently with probability  $q$  in each cycle. All processing elements are assumed to have this identical bernoulli

input process. This assumption is later relaxed by using weighting factors to allow each processing element have its own input rate  $q_j$ ,  $1 \leq j \leq N$ . We assume that there is no buffer space at the processing elements. After being generated, a packet is discarded if it cannot be delivered to the first stage of the interconnection network either due to a full buffer or a contention failure. Discarded packets at the entry to the network are not re-submitted. A packet, once accepted by the network, is never discarded inside the network. The input process is independent of the discarding process. (An extension of the model to allow blocked packets to be stored in a finite-sized queue or an infinite queue is discussed in later chapters.) From this assumption, the time delay of our performance measure is the total time a packet spends in the network. Time delay is meaningful only for those packets accepted into the network. The probability of acceptance (PA), another performance measure, is the probability that a packet is accepted into the network after it is generated. The normalized throughput is simply the probability of acceptance multiplied by the input rate. Current work also includes an extension to the case of multiple packet generation.

Each processing element has a memory module referencing pattern. A referencing pattern is the set of probabilities that a packet accesses the various memory modules. All previous work assumes that processing elements have the same referencing pattern. We shall allow P.E.s to have their own traffic pattern in Chapter 3. The memory module is assumed to be fast enough to accept 1 packet per cycle from switches at the last stage. This fast memory module assumption implies that there is no blocking at the last stage since a dedicated link connects one memory module to the output queue (see Figure 2.1). A slower memory module (e.g. 2 cycles to accept a packet) will have a severe effect on the performance of the network. Extension to slower memory models is underway.

### 2.1.2 The Routing Model

In the real world, packets are routed according to their destination address. However, in order to analyze the network analytically, we establish an abstract flow model that can be used in an approximate analytical model that faithfully reflects the steady state flow situation in the network. We propose a routing matrix  $r_{i,j}$ ,  $1 \leq i \leq n, 1 \leq j \leq N$  where  $r_{i,j}$  is the routing probability of the  $j$ th input port in stage  $i$ . A packet entering a switch will be routed either to the upper output queue with probability  $r_{i,j}$  or to the lower output queue with probability  $1 - r_{i,j}$ . To simulate a uniform traffic pattern, we simply let all  $r_{i,j}$  be 0.5. With equal probability of choosing output queues, no memory module is preferred. A special hot spot pattern can be created by letting all  $r_{i,j}$  be an identical value different than 0.5. For instance, by letting all  $r_{i,j}$  be 0.8 in a 10 stage network, 10.7% ( $= (0.8)^{10}$ ) of the total traffic will go to memory module 0 in a 1024-node network with 2.7% ( $= 0.2(0.8)^9$ ) of the traffic going to each of the second highest referenced memory modules (all memory modules with a single 1-digit in their address tag) and other fractions of traffic to the other memory modules. The advantage of this routing model is that by changing the value of  $r_{i,j}$  with proper mappings from real traffic patterns, we can evaluate any general traffic pattern. We leave the case of general  $r_{i,j}$  to be discussed in section 2.2 and in Chapter 3. Throughout this section, all  $r_{i,j}$  are assumed to have an identical value of  $r$ .

### 2.1.3 Analysis of the Unbuffered MIN

In this section, we study the unbuffered MIN and we have two objectives. The first is to solve for the probability of acceptance of a packet generated by

a processing element; the second is to show the effect any hot spot has on the throughput of the unbuffered MIN. The modelling approach employs a recurrence equation similar to the one Patel proposed [Pate 81]. It is an exact solution to the (approximate) model. The model assumptions and notation are presented in section 2.1.3.1. The results are shown in section 2.1.3.2. In section 2.1.3.3, the asymptotic behavior of the probability of acceptance is discussed.

### 2.1.3.1 Assumptions and Analysis

We assume that the network under study is a clocked, packet-switching MIN. There are  $N$  PE's and  $N$  MM's interconnected by an  $n$ -stage (i.e.  $N = 2^n$ ) interconnection network. All the operations of input and output take place at the end of each cycle which is the instance when we observe the system. With probability  $q$ , each PE generates a packet. The packet is submitted to the switch in the first stage at the end of the cycle. With probability  $1 - q$ , the PE remains idle. A packet is routed to an output port according to an address tag. Each switch sends a packet to the next stage at the end of the cycle if it is not empty. If two packets are contending for the same output port, one packet is randomly accepted while the other is discarded (since there are no extra buffers). The contention resolution is unbiased. The input generation process of the PE's is independent of packet discarding. The discarded packets are not re-submitted. As long as the output port is not empty, the speed of the MM is assumed to be fast enough to remove one packet from the output port of the last stage. We also assume that the operation of the MIN is simultaneous, which means that if all the stages are non-empty, then in the next cycle, all packets move to the next stage simultaneously.

In order to calculate the probability of acceptance, we must calculate  $P_{i,j}$ .

the probability that  $j$ th output port of stage  $i$  is not empty. Then  $P_{n,j}$  is the probability that  $j$ th queue in the last stage is not empty. Hence the summation of  $P_{n,j}$  for all  $j$ 's is the total output rate of the network. Dividing the total output rate by the total input rate,  $N \cdot q$ , we get the average probability of acceptance.

Let us designate the  $j$ th queue (in a column) in stage  $i$  as  $Q_{i,j}$  and let  $P_{i,j}$  be the probability that  $Q_{i,j}$  is not empty. We first determine the equations for  $P_{1,1}$  and  $P_{1,2}$ , then we generalize this equation to a general  $P_{i,j}$ . In the first switching element of the first stage, there are two queues (namely  $Q_{1,1}$  and  $Q_{1,2}$ ) that take incoming packets from two processing elements. The probability that a PE is not empty is  $q$ . Therefore the probabilities that  $Q_{1,1}$  and  $Q_{1,2}$  are not empty are as follows :

$$P_{1,1} = 1 - [1 - q \cdot r]^2$$

$$P_{1,2} = 1 - [1 - q \cdot (1 - r)]^2$$

A processing element sends a packet to  $Q_{1,1}$  with probability  $q \cdot r$ . Hence,  $(1 - q \cdot r)^2$  is the probability that  $Q_{1,1}$  does not receive a packet from either incoming source (processing element). Thus  $P_{1,1} (= 1 - (1 - q \cdot r)^2)$  is the probability that there is a packet at  $Q_{1,1}$ . Similarly, a processing element sends a packet to  $Q_{1,2}$  with probability  $q \cdot (1 - r)$ . Hence  $P_{1,2} (= 1 - (1 - q \cdot (1 - r))^2)$  is the probability that there is a packet at  $Q_{1,2}$ . Since all processing elements are identical with packet generation rate  $q$ , the queues of other switching elements in the first stage have the same probability  $P_{1,1}$  and  $P_{1,2}$  that they are not empty. Hence for switching elements in the first stage, the probability that the upper output port is not empty is  $P_{1,1}$  and the probability that the lower output port is not empty is  $P_{1,2}$ . With Banyan interconnection, all upper output ports are interconnected to the

first half of the switching elements in the next stage, and all lower output ports are interconnected to the second half of the switching elements. Therefore the equivalent source that feeds each of the output queues of switching elements in the first half of the second stage has a rate equal to  $P_{1,1}$ . Therefore the probability that any of the upper output queues of the first half of the switching elements in the second stage is not empty is  $P_{2,1} = 1 - [1 - P_{1,1} \cdot r]^2$ . Similarly, the probability that each of the lower output queues of the first half of the switching elements in the second stage is not empty is  $P_{2,2} = 1 - [1 - P_{1,1} \cdot (1 - r)]^2$ . In general,  $P_{i,j}$  acts as a source in stage  $i$  that feeds a set of switching elements in stage  $i+1$ . Hence the probability that the upper ports and the lower ports of these switching elements are not empty can be found as follows :

$$\begin{aligned}
 P_{i+1,2j-1} &= 1 - [1 - P_{i,j} \cdot r]^2 \\
 P_{i+1,2j} &= 1 - [1 - P_{i,j} \cdot (1 - r)]^2
 \end{aligned} \tag{2.1}$$

Solving  $P_{i,j}$  stage by stage, we get  $P_{n,j}$ , for  $j=1$  to  $N$ . Then the average probability of acceptance (PA) is found as :

$$PA = \frac{\sum_{j=1}^N P_{n,j}}{N \cdot q} \tag{2.2}$$

where  $\sum_{j=1}^N P_{n,j}$  is the total output rate from the MIN, and  $N \cdot q$  is the total input rate to the interconnection network. This is our recurrent solution for the probability that a generated packet is eventually accepted.

### 2.1.3.2 Results

By substituting different values of  $r$  into the model, we get PA values of the unbuffered MIN under different hot spot traffic patterns. The light traffic load ( $q=0.1$ ) case is shown in Figure 2.2.

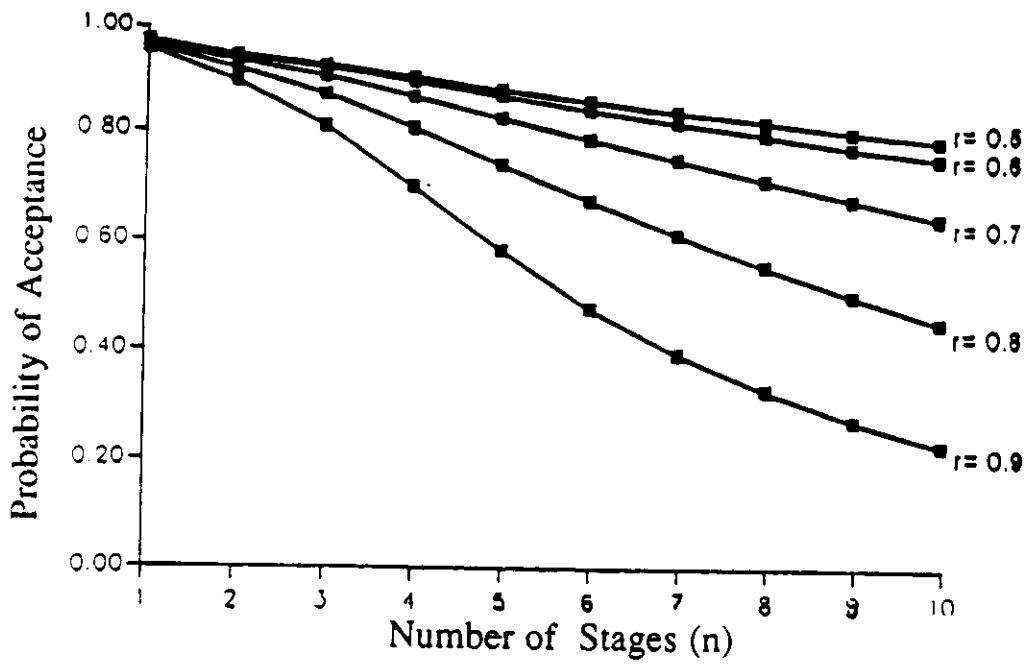


Figure 2.2: Probability of Acceptance for q=0.1 case

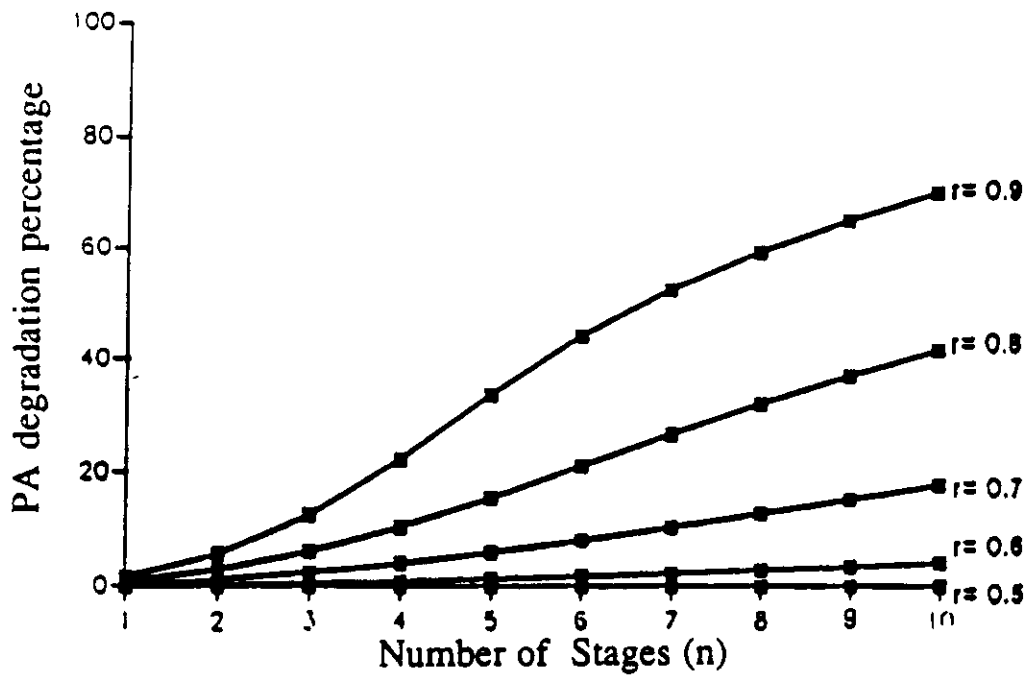


Figure 2.3: The PA degradation percentage for q=0.1 case



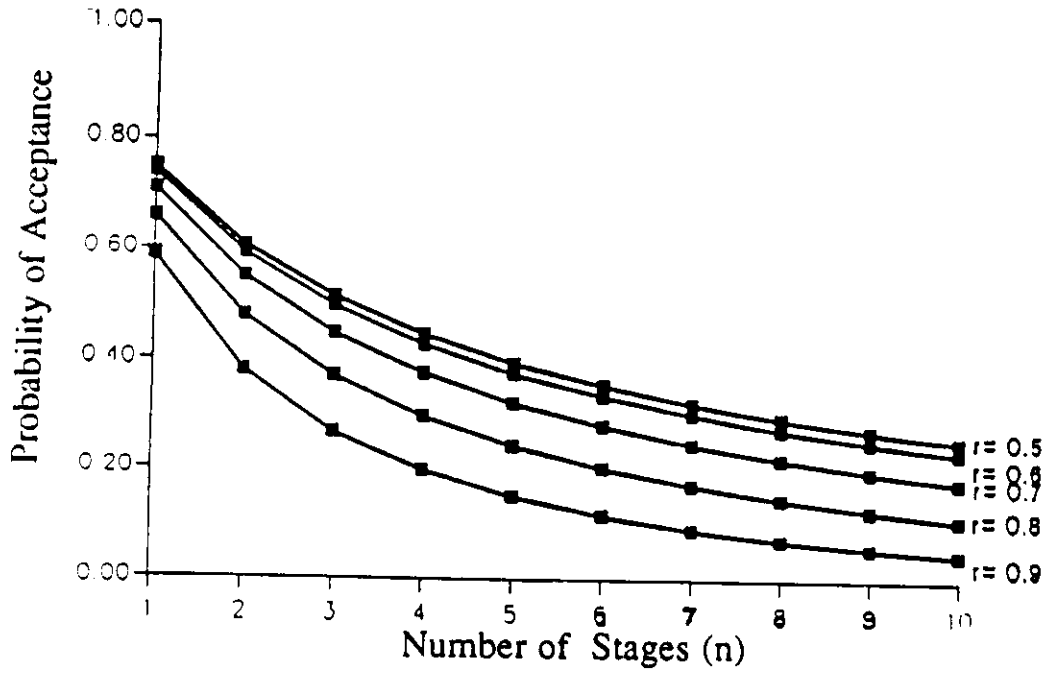


Figure 2.4: Probability of Acceptance for  $q=1.0$  case

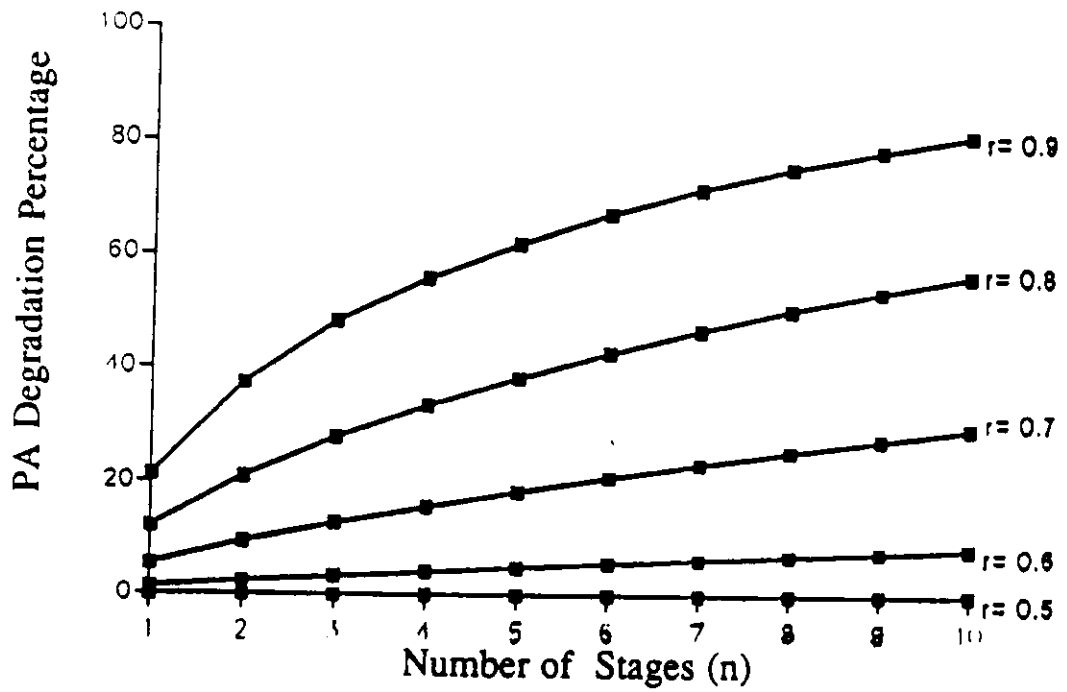


Figure 2.5: The PA degradation percentage for  $q=1.0$  case

From the figure, we see that the larger the network size, the worse the PA degradation is when we increase the non-uniformity of the traffic. In Figure 2.3, we show the PA degradation percentage when we increase hot spot traffic from  $r=0.5$  gradually to  $r=0.9$  for each stage. The reduction in PA for the  $r=0.9$ , 10-stage case, compared to the  $r=0.5$  case, is 71%. Since the traffic load is small, when we have a small number of stages the contention probability is small, as well. Thus, there is no significant difference in the values of PA's for various values of  $r$ . As we increase the network size, the difference between the PA's begins to show because the contention probability is high in a large network.

Increasing the offered traffic to  $q=1.0$ , the probability of collision for small networks becomes high. Hence, even with a uniform traffic pattern, the PA of a 10 stage network drops to 0.27 in the heavy load case ( $q=1.0$ ), compared to 0.8 in the light load case ( $q=0.1$ ). Furthermore, PA is severely degraded by hot spots under heavy traffic, as shown in Figure 2.4. If we compare the PA degradation percentage of the light traffic case to the heavy traffic case, the degradation in the heavy traffic case is more serious than in the light traffic case. An 80% degradation in PA for the  $r=0.9$ ,  $n=10$  case indicates that a hot spot has a dramatic influence on PA for an unbuffered MIN.

### 2.1.3.3 Asymptotic Behavior

In this section, we discuss the asymptotic behavior of the probability of acceptance of the output port queues that are in the paths which lead to the hot spot. We notice that along these paths, all the ports in the same stage behave the same. Therefore, we can represent the queues in the same stage by a probability,  $P_j$ . Then the recurrence equation which relates probabilities in neighboring stages is :

$$P_{j+1} = 1 - (1 - P_j \cdot r)^2$$

$$P_j = 1 - (1 - P_{j-1} \cdot r)^2$$

Subtracting the second equation from the first one, we get the following :

$$P_{j+1} - P_j = [2 - r \cdot (P_j + P_{j-1})] \cdot r \cdot (P_j - P_{j-1})$$

The first term on the right hand side in square brackets is always positive. Hence,  $P_j$  is either an increasing function or a decreasing function. If  $P_j$  is an increasing function, then

$$1 - (1 - P_j \cdot r)^2 > P_j$$

After some simple algebra, we find the asymptotic behavior of  $P_j$  ( $j \gg 1$ ) as follows :

$$P_j < \frac{2r - 1}{r^2} \quad (2.3)$$

A similar expression can be derived for the decreasing case :

$$P_j > \frac{2r - 1}{r^2} \quad (2.4)$$

From equations (2.3),(2.4), we know that when  $r > 0.5$ ,  $P_j$  converges to  $\frac{2r-1}{r^2}$  when  $j$  approaches infinity which means that the output port that connects to the favorite memory module converges to  $\frac{2r-1}{r^2}$  for  $j \gg 1$ .

For the  $r < 0.5$  case, we can show that  $P_j$  approaches zero when  $j$  approaches infinity as follows :

$$P_{j+1} = 2r \cdot P_j - (r \cdot P_j)^2$$

$$P_{j+1} < 2r \cdot P_j$$

It turns out that  $P_j$  converges to zero faster than  $q \cdot (2r)^j$  for  $r < 0.5$  case. This is the probability for other memory modules when  $j \gg 1$ .

We derive the asymptotic expression of  $P_j$  for a MIN composed of  $k \times k$  switches by using the method in [Krus 83]. The result is as follows :

$$P_j = \frac{1}{\frac{1}{q} \cdot \left(\frac{1}{kr}\right)^j + \left(\frac{k-1}{2k} + (k+1)\frac{kr-1}{6k}\right) \cdot \sum_{i=0}^{j-1} \left(\frac{1}{kr}\right)^i} \quad (2.5)$$

When  $k$  equals 2, we get

$$\lim_{k \rightarrow 2, j \rightarrow \infty} P_j = \frac{2r-1}{r^2}$$

which is the convergence given in the previous discussion. If we let  $r = \frac{1}{k}$  (the uniform traffic case), we get the following :

$$P_j = \frac{2k}{(k-1) \times j + \frac{2k}{q}}$$

which is Kruskal's result [Krus 83].

#### 2.1.4 Analysis of the Buffered MIN

The finite-buffered MIN is now analyzed using an approximate iterative method. The approximation comes from the decomposition of a network of queues into independent queues. The iteration is repeated until PA converges within  $10^{-6}$ . In section 2.1.4.1, the model assumption and model approach are both described. The probability of acceptance (PA), the mean queue size of the tree and the average time delay of a packet are determined in section 2.1.4.2.

##### 2.1.4.1 Modeling Analysis

The proposed approximate analytical approach employs a *decomposition* and *iteration* method. The real interconnection network is in fact a network of finite-buffered queues with blocking. The dependency among queues, caused by the

blocking from stage to stage, makes the exact analysis intractable. We shall use a similar approximation technique as that applied in tandem queues with blocking [Bran 88], [Case 79] and [Perr 86] where approximate analyses are used. The approximation method decomposes a queue in the tandem configuration with equivalent input rates and blocking conditions. The Markov chain of the decomposed queue is then solved; the steady state probabilities are then used as equivalent input and blocking conditions for other queues. The iterative method decomposes and analyzes the tandem queues one by one, then the whole process is repeated until it converges, if it has a steady state. The concept of using this decomposition and iteration approximation method in analyzing the finite-buffered Banyan network is very similar to that of tandem queues. The only difference is that instead of a single input source and a single output queue for each queue in the tandem configuration, the interconnection network has 2 input sources and 2 output queues for each queue in the network (except for the last stage queue where only 1 output sink is presented, namely, the memory module). Therefore, when we solve for the equivalent input rates and blocking conditions for a decomposed queue, we consider the combined input from 2 input sources and the combined probability of blocking from the 2 output queues. The approach is described below.

Let  $P_{i,j}(k)$  be the steady state probability that there are  $k$  packets in the queue  $Q_{i,j}$ . Let  $Q_{i-1,j1}$  and  $Q_{i-1,j2}$  be two input sources from stage  $i-1$  that feed  $Q_{i,j}$ . Let  $X[i]$  be the probability that there are  $i$  packets destined for  $Q_{i,j}$  from its two input sources (since each source at most sends one packet every cycle, hence  $0 \leq i \leq 2$ ). We solve for the equivalent input rates for a queue  $Q_{i,j}$  which is located at output port 0 as follows:

$$\begin{aligned}
X[1] &= r[P_{i-1,j_1}(0)(1 - P_{i-1,j_2}(0)) + P_{i-1,j_2}(0)(1 - P_{i-1,j_1}(0))] \\
&\quad + 2r(1 - r)(1 - P_{i-1,j_1}(0))(1 - P_{i-1,j_2}(0)) \\
X[2] &= [r(1 - P_{i-1,j_1}(0))] \cdot [r(1 - P_{i-1,j_2}(0))] \\
X[0] &= 1 - X[1] - X[2] \tag{2.6}
\end{aligned}$$

The first term in  $X[1]$  equation is the case that one queue in the previous stage is empty, and the other is not. The non-empty one chooses the output port 0 with probability  $r$ . The second term is the case that both queues in the  $(i-1)$ th stage are not empty, and one chooses output port 0 with probability  $r$  and the other chooses another output port with probability  $1 - r$ . The summation of probabilities in both cases represents the probability that only one input packet feeds the queue. The  $X[2]$  equation represents the case when both queues in the previous stage are not empty and they both choose output port 0 with probability  $r$  (This calculation assumes that the output port queue is the upper of the two; if it is the lower output port queue, then the routing probability  $r$  is replaced by  $1 - r$ ).

Regarding the equivalent blocking condition, let  $B_{i,j}$  be the probability that a packet in the  $j$ th queue in stage  $i$  is blocked at the end of a cycle. Let  $C_{i,j}$  be the probability that the  $j$ th queue in stage  $i$  is blocking a packet in stage  $i - 1$ . Let  $Q_{i+1,j_1}$  and  $Q_{i+1,j_2}$  be the two output queues of  $Q_{i,j}$  and let  $Q_{i,l}$  be the other queue that also feeds both  $Q_{i+1,j_1}$  and  $Q_{i+1,j_2}$  (i.e. a possible contending queue for  $Q_{i,j}$ ). Then the equivalent blocking condition for queue  $Q_{i,j}$  is as follows :

$$\begin{aligned}
B_{i,j} &= r \cdot C_{i+1,j_1} + (1 - r) \cdot C_{i+1,j_2} \\
\text{where } C_{i+1,j_1} &= P_{i+1,j_1}(K) + \frac{r}{2} \cdot (1 - P_{i,l}(0)) \cdot P_{i+1,j_1}(K - 1) \tag{2.7}
\end{aligned}$$

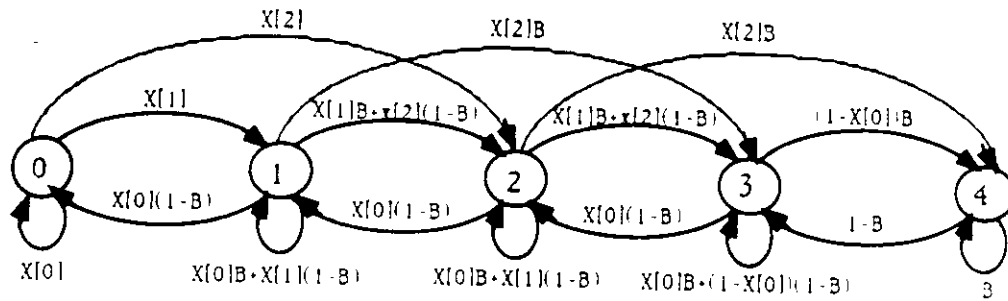


Figure 2.6: Markov chain of a queue  $Q_{i,j}$  extracted from the network where the state variable represents the number of packets in that queue:  $K=4$ .

The first term in the  $B_{i,j}$  equation represents the case when the packet at the head of queue  $Q_{i,j}$  chooses  $Q_{i+1,j1}$  with probability  $r$  and is blocked by  $Q_{i+1,j1}$ . The second term represents the other case when the packet chooses  $Q_{i+1,j2}$  and is blocked. There are two situations in which a queue blocks a packet in the preceding stage : firstly, when the queue is full, and secondly, when the queue has only one more space and a contention from  $Q_{i,l}$  wins the arbitration.

Given a set of initial values for the variables of the network, we "extract" queue  $Q_{1,1}$  from the network (with the equivalent input rates and blocking conditions as exist in the network) as an independent queue. The Markov chain for this queue is then solved to get new values for the state probabilities. A sample Markov chain for the queue  $Q_{i,j}$  is shown in Figure 2.6 where  $B$  represents the blocking probability  $B_{i,j}$ . We repeat this process for other queues in the first stage, in the order  $Q_{1,2}, Q_{1,3}, \dots, Q_{1,N}$ . Using these new state probabilities as the new input rates, we repeat the same process for all queues in the second stage in the order  $Q_{2,1}, Q_{2,2}, \dots, Q_{2,N}$ . This process is then repeated for all stages. Now we have a new set of values for network variables which can be used to compute the new blocking probabilities. This new set of values is used in the next iteration to compute another set of new values. etc.. The iteration process is repeated until

the difference of the probability of acceptance between two consecutive iterations is below  $10^{-6}$ .

The performance measures that are of interest are the probability of acceptance, the normalized throughput and the average time delay. There are two ways to calculate probability of acceptance. If we sum the output rate over all output ports and divide it by the total input rate, we get the probability of acceptance :

$$P.A_{out} = \frac{\sum_{i=1}^N [1 - P_{n,i}(0)]}{q \cdot N} \quad (2.8)$$

The total output rate over the input rate is the probability of acceptance at the output port. From the input port, we solve for the probability that a packet generated at the PE's is discarded due to a full buffer or a contention failure at the first stage. This discarding probability is  $B_{0,j}$  , which can be solved for using equation (2.7). Hence,

$$P.A_{in} = 1 - B_{0,j} \quad (2.9)$$

Both values, although solved in different ways, should be equal when the MIN reaches steady state. (This can be used to test for the correctness of the model.) The normalized throughput is found by multiplying the probability of acceptance by the input rate. We apply Little's result (the average number of customers equals the arrival rate times the mean delay) to calculate the average time delay of a packet. When the network reaches steady state, we take the sum of the mean queue sizes summed over the whole network using the steady state probabilities of the queue size for each queue. Given the throughput and the average numbers of customers in the system, the average time delay can be solved for by applying Little's result.



#### 2.1.4.2 Results

In this section, we focus on four performance measures. The first measure is the probability of acceptance for both a hot spot traffic pattern and a uniform traffic pattern for both the unbuffered case and the buffered case. The effect of buffering on PA is given. The second measure is the mean queue size of those queues in the saturated tree. The motivation for this study is to see how the tree is formed under the influence of hot spot traffic and offered load. Next, we study (an upper bound on) the tree build-up time. The last performance measure that we are interested in is the average time delay of a packet in the MIN.

##### PA Improvement

In this section, we compare PA values of the finite-buffered MIN to the PA values of the unbuffered MIN which are taken from the first model solved in section 2.1.3. We increase the number of buffers gradually, for different network sizes (1 to 9 stages) under different traffic conditions.

The uniform light traffic case is shown in Figures 2.7 and 2.8. The performance improvement is insignificant because the contention probability is small due to a uniform, light traffic pattern. Adding one buffer provides the greatest improvement because this buffer saves most of the collided packets which would be discarded in the unbuffered case. For the unbuffered case, as the number of stages grows, we see that the PA decreases (in fact, it decreases to zero as stage number approaches infinity,  $P_j < \frac{2r-1}{r^2}$  as discussed in section 2.1.3.3). By adding one buffer, we improve the probability of acceptance to 0.98. We see an insignificant improvement after adding two buffers when PA approaches 1. This agrees with what [Dias 81] had claimed. We notice that for the single stage MIN, the PA for the unbuffered and the single buffer case are equal. Although

the unbuffered MIN has no buffer to store the blocked packet, it has a room for temporarily keeping a packet. Thus it is essentially a one buffer system. The difference between this and the single buffer case is attributed to the fact that a collided packet is dropped in the unbuffered case, but is stored in the buffered case. For the single stage MIN, the collision for the unbuffered MIN happens only at the input of PE's, and this is the same situation as in buffered MIN. Therefore, the probabilities are equal. But for a larger MIN, i.e. with more than one stage, packet discarding occurs through all stages for the unbuffered MIN. The buffered MIN still maintains the property that packet discarding occurs only at the PE's. Hence, storing blocked packets in the buffered case improves the system performance. Figure 2.8 shows the improvement percentage of PA by adding buffers compared to the unbuffered case.

In an uniform, heavy traffic case (Figures 2.9 and 2.10), if a MIN does not have buffers to hold those collided packets, PA decreases significantly as the number of stages grows due to the high contention probability under heavy traffic condition. After adding buffers, PA improves significantly. Thus buffering becomes important to improve the performance when the contention probability is high. PA jumps up sharply by adding two buffers, but slows down when buffer size grows beyond that. After adding roughly 4 buffers, the difference of PA's between different network sizes is very small because 4 buffers are enough to save most of the collided packets.

In Figures 2.11 and 2.12, we show how a hot spot pattern affects PA for the unbuffered and the buffered MIN. Under the same light traffic condition, PA decreases significantly for the unbuffered MIN under the influence of a hot spot, as compared to the uniform traffic case shown in Figure 2.7. For a 9-stage unbuffered MIN, PA is 0.27 with a hot spot traffic pattern, compared to 0.82

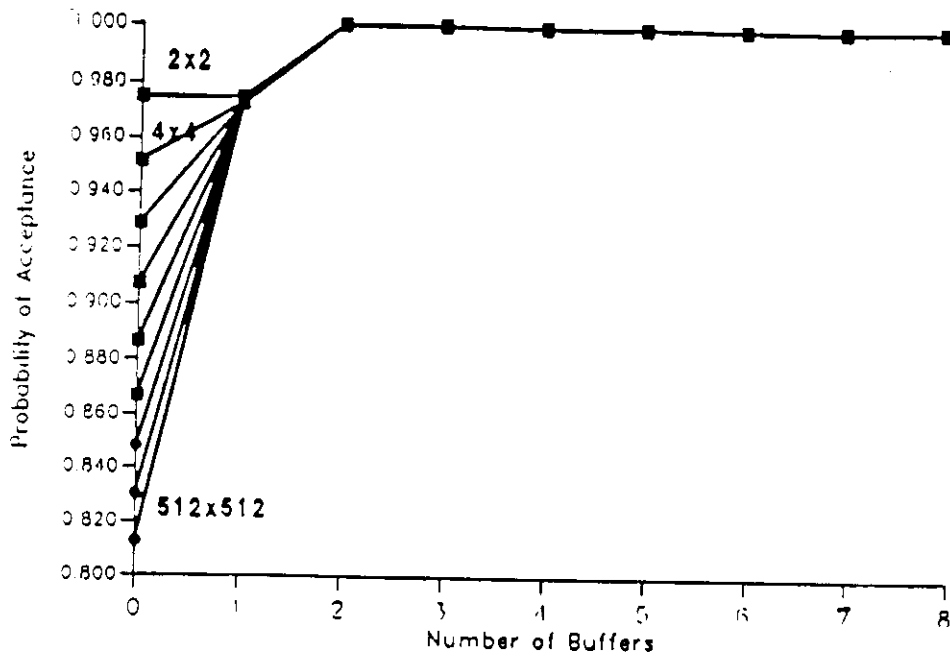


Figure 2.7: Improving the Probability of Acceptance by adding buffers for interconnection networks with  $q=0.1$  and a uniform traffic pattern

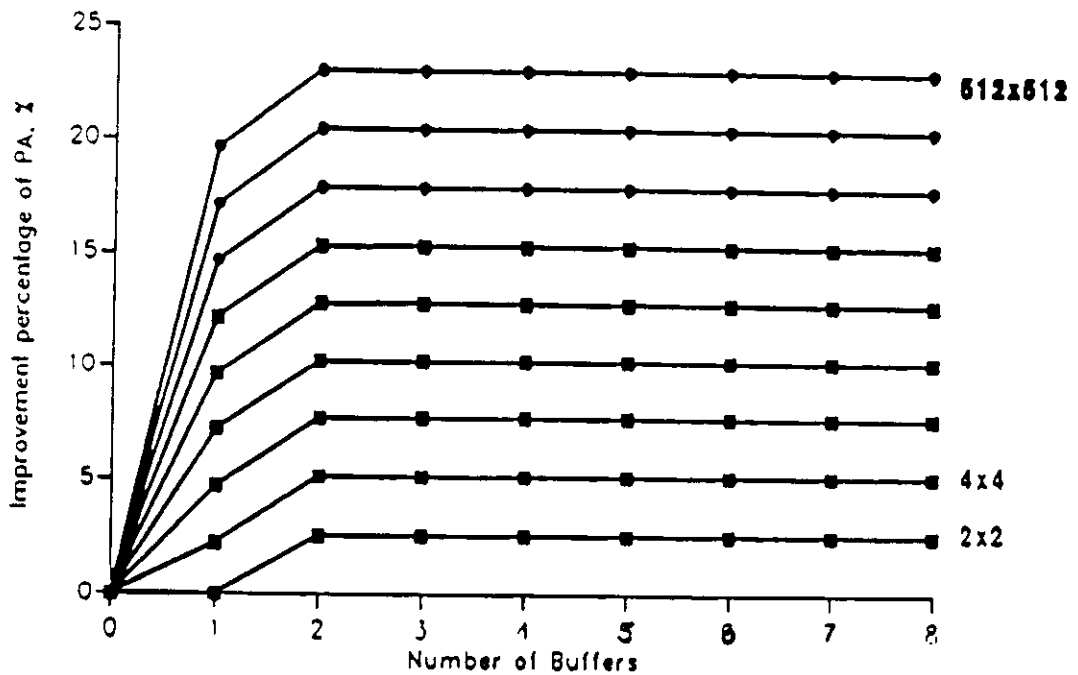


Figure 2.8: The PA improvement percentage by adding buffers for interconnection networks with  $q=0.1$  and a uniform traffic pattern

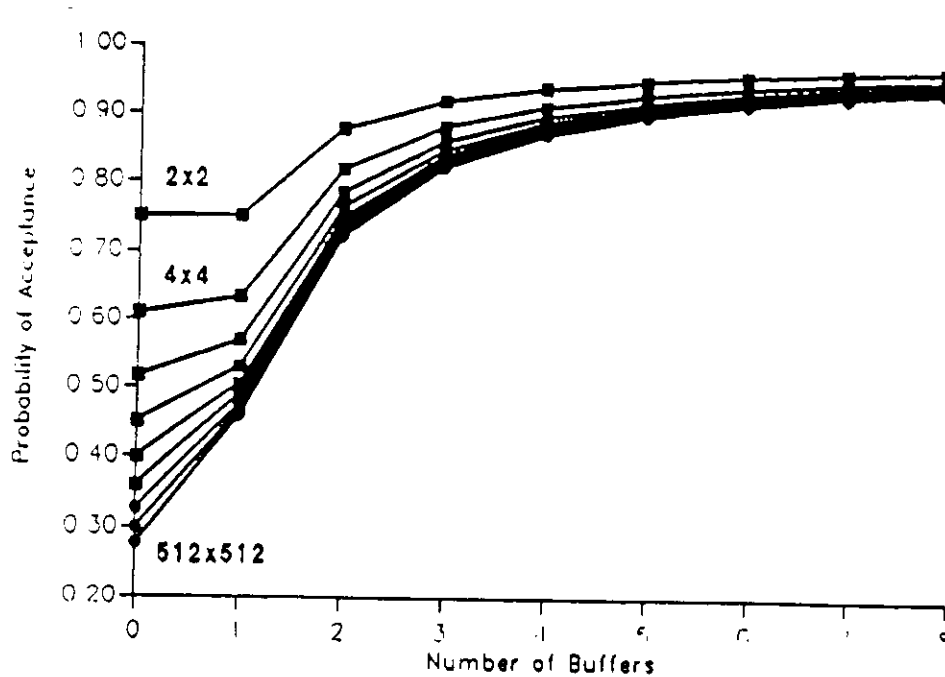


Figure 2.9: Improving the Probability of Acceptance by adding buffers for interconnection networks with  $q=1.0$  and a uniform traffic pattern

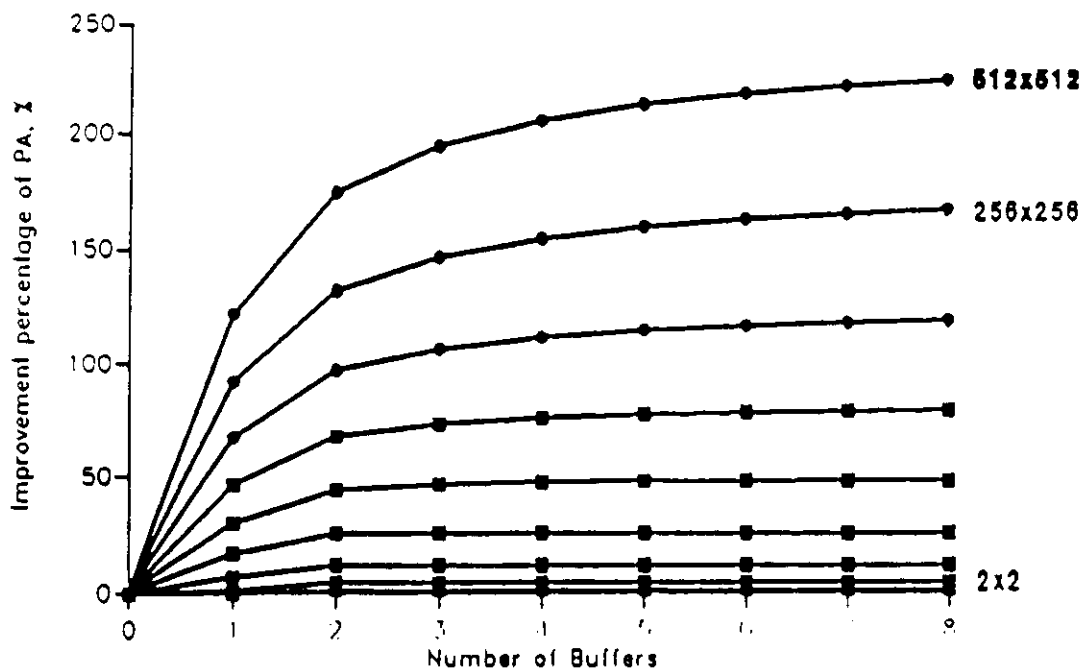


Figure 2.10: The PA improvement percentage by adding buffers for interconnection networks with  $q=1.0$  and a uniform traffic pattern

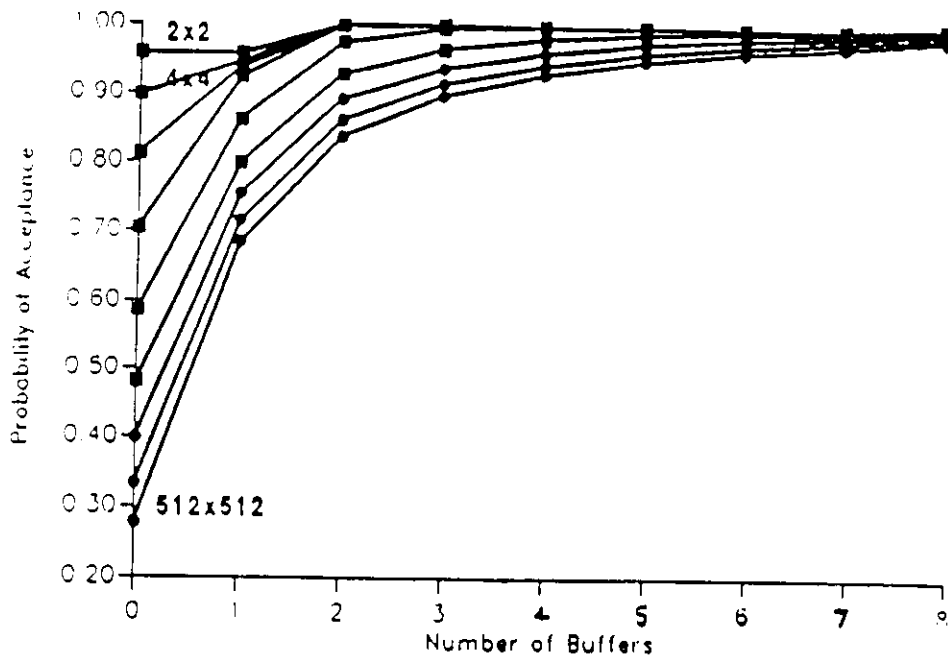


Figure 2.11: Improving the Probability of Acceptance by adding buffers for interconnection networks with  $q=0.1$  and a non-uniform traffic pattern ( $r=0.9$ )

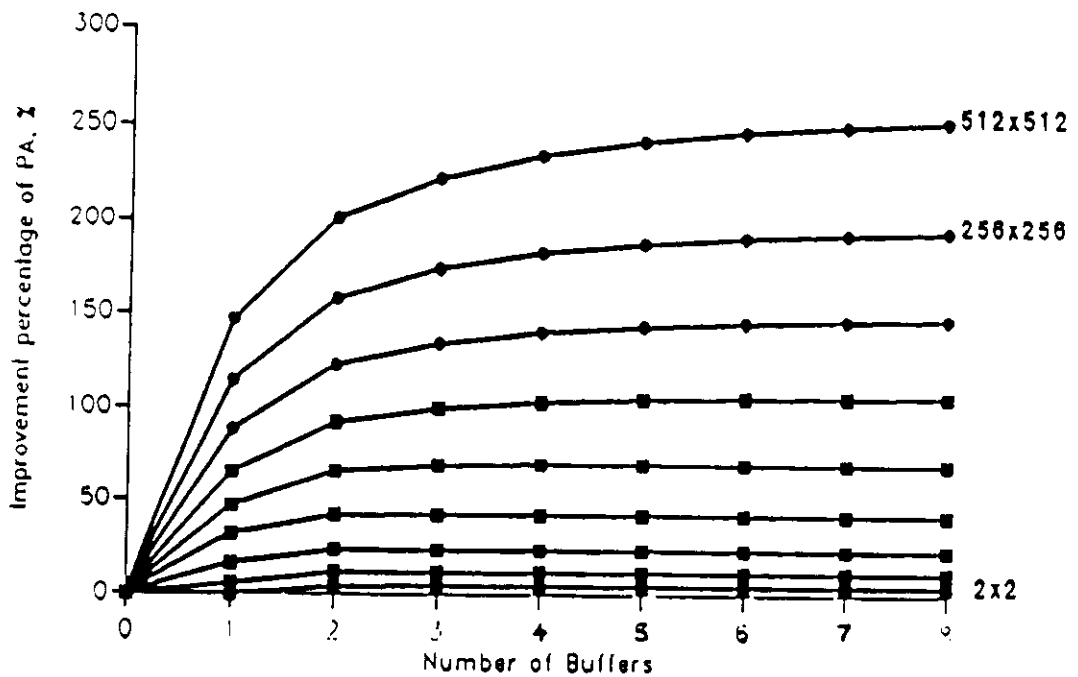


Figure 2.12: The PA improvement percentage by adding buffers for interconnection networks with  $q=0.1$  and a non-uniform traffic pattern ( $r=0.9$ )

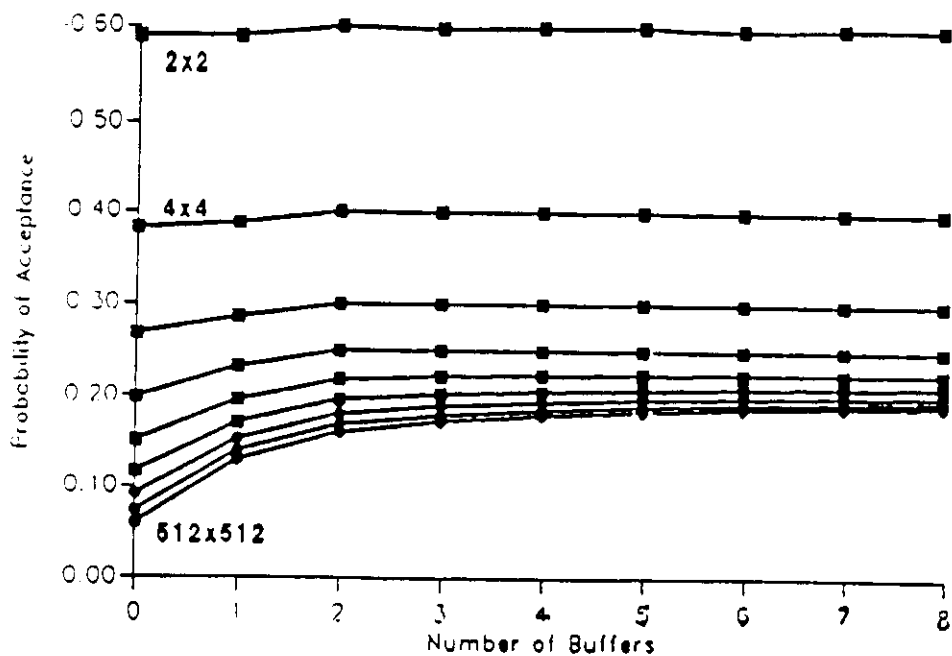


Figure 2.13: Improving the Probability of Acceptance by adding buffers for interconnection networks with  $q=1.0$  and a non-uniform traffic pattern ( $r=0.9$ )

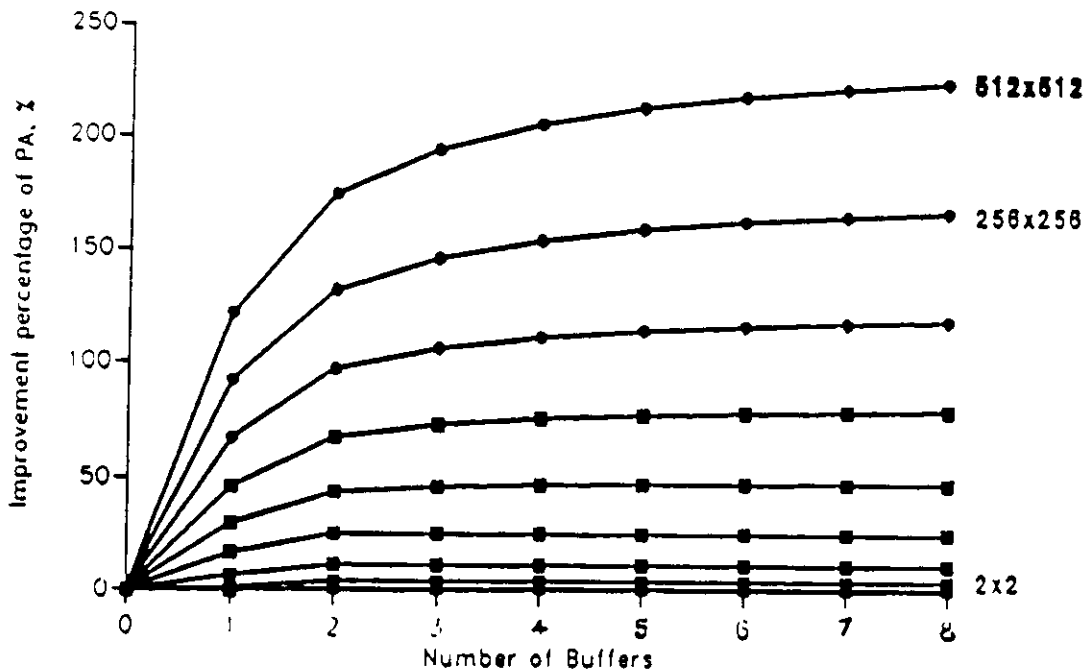


Figure 2.14: The PA improvement percentage by adding buffers for interconnection networks with  $q=1.0$  and a non-uniform traffic pattern ( $r=0.9$ )

with a uniform traffic pattern. As expected, adding the first buffer makes a big difference for a large MIN. The difference of PA's for different size networks gets smaller when the buffer size grows, as in the uniform traffic case. However, the PA improvement percentage is so high that for a 9-staged MIN, the percentage is over 250% for 3 buffers. Buffering improves the normalized throughput of a MIN under hot spot traffic significantly even when the offered traffic is very light.

For the heavy traffic case under a hot spot influence, the PA value is shown in Figures 2.13 and 2.14. In heavy traffic cases, even adding buffers cannot diminish the difference of PA for different network sizes. However, the improvement for a 9 stage MIN is still nearly 250% for the 3 buffers case. Buffering does not help much for the heavy load case with a hot spot traffic pattern because the tree is quickly saturated. The PA for a large network is not high even with 3 buffers added. For a 9-staged, 3-buffered MIN, the PA is less than 0.2. This low packet acceptance is due to tree saturation; hence most packets are discarded at the input to the MIN. Unlike the uniform traffic case where PA can be improved to nearly 1 by adding 3 buffers, tree saturation dominates the performance of the MIN in the non-uniform, heavy traffic case.

We conclude that a hot spot pattern degrades the performance significantly. Buffering helps to improve the probability of acceptance in many cases. However, buffering does not help much when the hot spot pattern is very severe.

### **The Average Number of Busy Buffers**

In this section, the average number of busy buffers for those queues in the saturated tree is found. Because tree saturation dominates the performance of the MIN, understanding the evolution of the number of busy buffers helps to study the behavior of tree saturation.

When the iterative model reaches steady state, we have the state probabilities

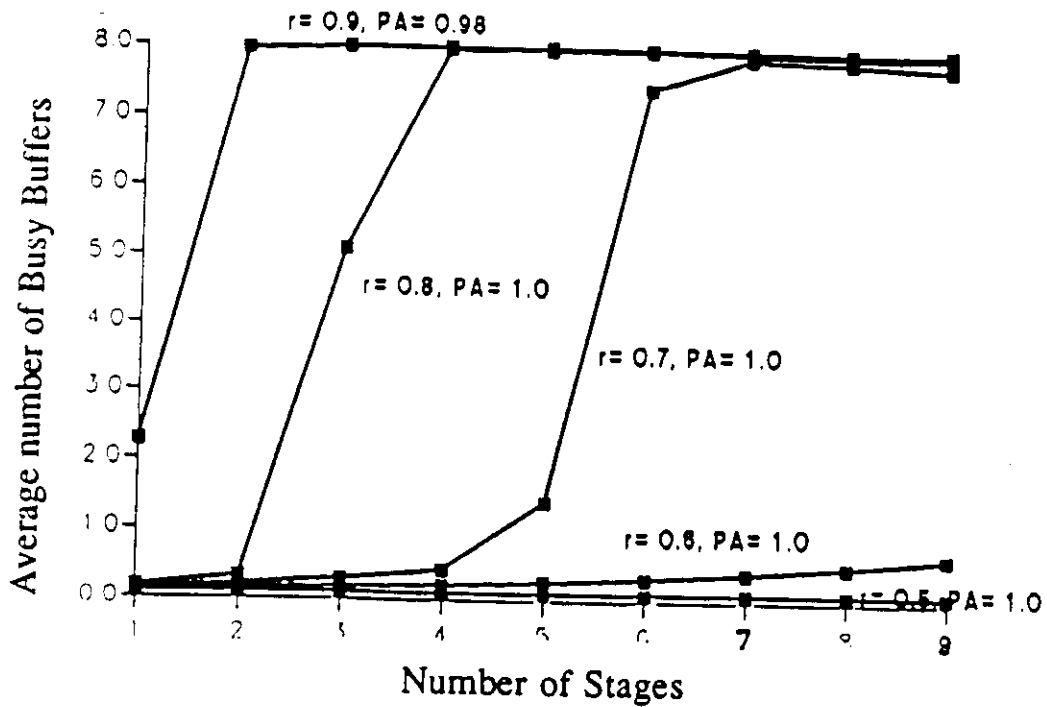


Figure 2.15: The mean busy buffer size for a light traffic case ( $q=0.1$ ) with  $K=8$

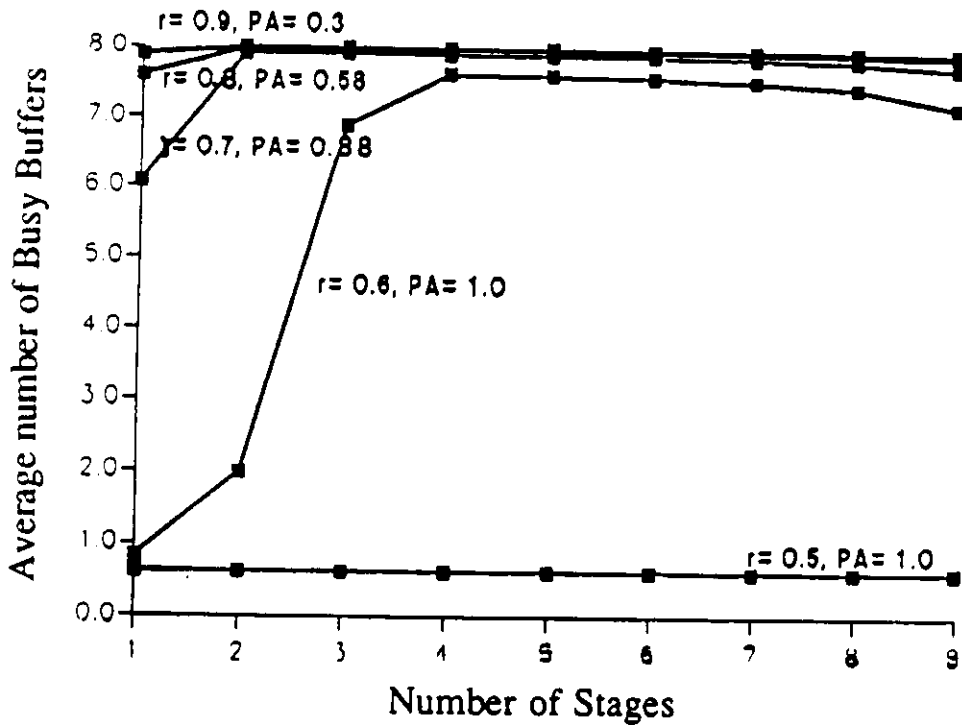


Figure 2.16: The mean busy buffer size for a moderate traffic case ( $q=0.5$ ) with  $K=8$



information for each queue in each stage. If we calculate the mean busy buffer size for those queues in the tree, then we know how the tree evolves. Note that queues in the same stage are statistically identical. In figure 2.15 - 2.18, we show the mean queue size for a 9-staged, 8-buffered MIN for different values of  $r$ , by changing the offered load in each chart.

When  $q=0.1$ , the offered traffic is so light that even adding a fraction of non-uniformity by increasing  $r$  from 0.5 to 0.6, the average number of busy buffers for an 8-buffered MIN is still below 1 for all stages. We notice that when  $r$  equals 0.7, a saturated tree begins to form starting from the hot memory module, and propagates back to stage 6. When  $r$  equals 0.9, all the queues in later stages are full, but the first stage contains only 2 packets on the average. Therefore, PA is still nearly 1 despite the fact that a small tree is formed.

When we increase the offered traffic to 0.5, the tree saturation begins to demonstrate its influence over the performance of the MIN. The average busy buffer size still remains below 1 for uniform traffic. A little increase in non-uniformity ( $r=0.6$ ) causes a tree to be formed for the last 7 stages. For  $r=0.9$ , even the queues in the first stage are full. Tree saturation reduces the value of PA when  $r \geq 0.8$ .

When the offered traffic is increased to 0.7, the average busy buffer size for the uniform case is only 1. When  $r$  equals 0.6, the tree is saturated to the first stage with a mean queue size equal to 5 at the first stage. PA for this case is still high due to the small non-uniform traffic, where 40% of the traffic still flows into queues outside the tree which helps keep traffic from being congested. The same reasoning applies for the  $r=0.7$  case where almost all the queues in all stages are full, and the PA still remains high at 0.71.

Under heavy traffic conditions, a small increase in the hot spot traffic quickly

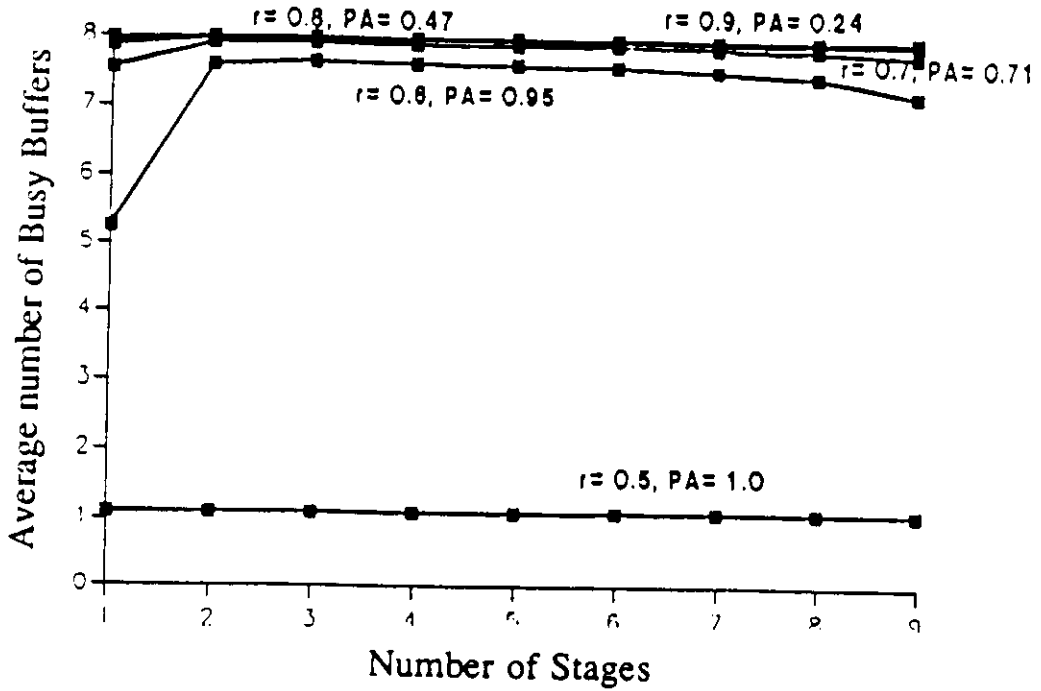


Figure 2.17: The mean busy buffer size for a moderate-to-heavy traffic case ( $q=0.7$ ) with  $K=8$

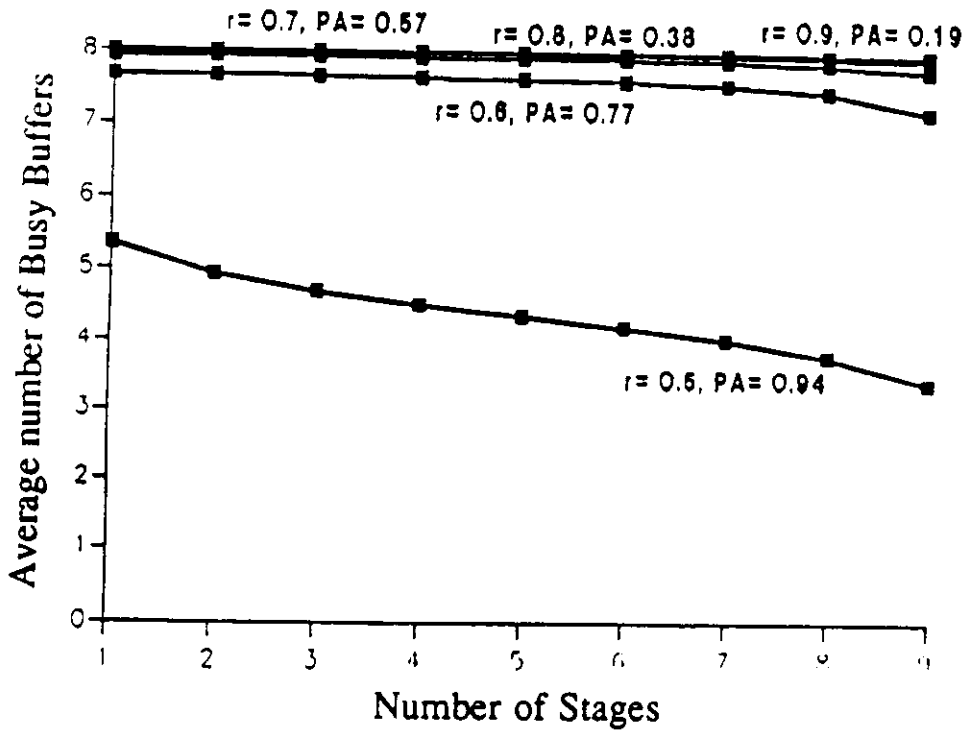


Figure 2.18: The mean busy buffer size for a heavy traffic case ( $q=1.0$ ) with  $K=8$

saturates the tree, reducing the probability of acceptance significantly except for the  $r=0.5$  case. When  $r=0.5$ , all packets have an equal probability of choosing any output port. The heavy traffic causes the MIN to fill up to a moderate degree. Note that the average busy buffer size decreases when the stage number increases. An analogy to this situation is a highway traffic jam. The effect of blocking is propagated back from the last stage to the first stage: therefore, the queues in the later stages contain fewer packets than do the ones in the earlier stages.

Depending on the values of  $q$  and  $r$ , tree saturation affects PA in various degrees. In some cases, even when the tree is formed, the MIN still maintains a decent value of PA. Adding more buffers might be able to compensate for the effect of tree saturation, but it becomes impractical when the network size is large.

### Tree Build-up Time

One interesting problem concerning the saturated tree is the tree build-up time for different loads and different degrees of non-uniformity. Let us define  $T_1$  to be the average time when the network first starts to drop packets at the input due to blocking, and  $T_2$  to be the average time for the system to stabilize.  $T_1$  is the time when the saturated tree reaches the processing elements. This is the performance measure in which we are interested. Applying Little's result, we have the following equation (see figure 2.19(a)) :

$$q \cdot N \cdot T_2 = \bar{N}_{MIN} + \bar{N}_{drop} + \bar{N}_{out}$$

where after  $T_2$  cycles, on the average,  $q \cdot N \cdot T_2$  is the total number of packets generated (remember that  $N$  is the number of processing elements),  $\bar{N}_{MIN}$  is the average number of packets still in the network,  $\bar{N}_{drop}$  is the average total

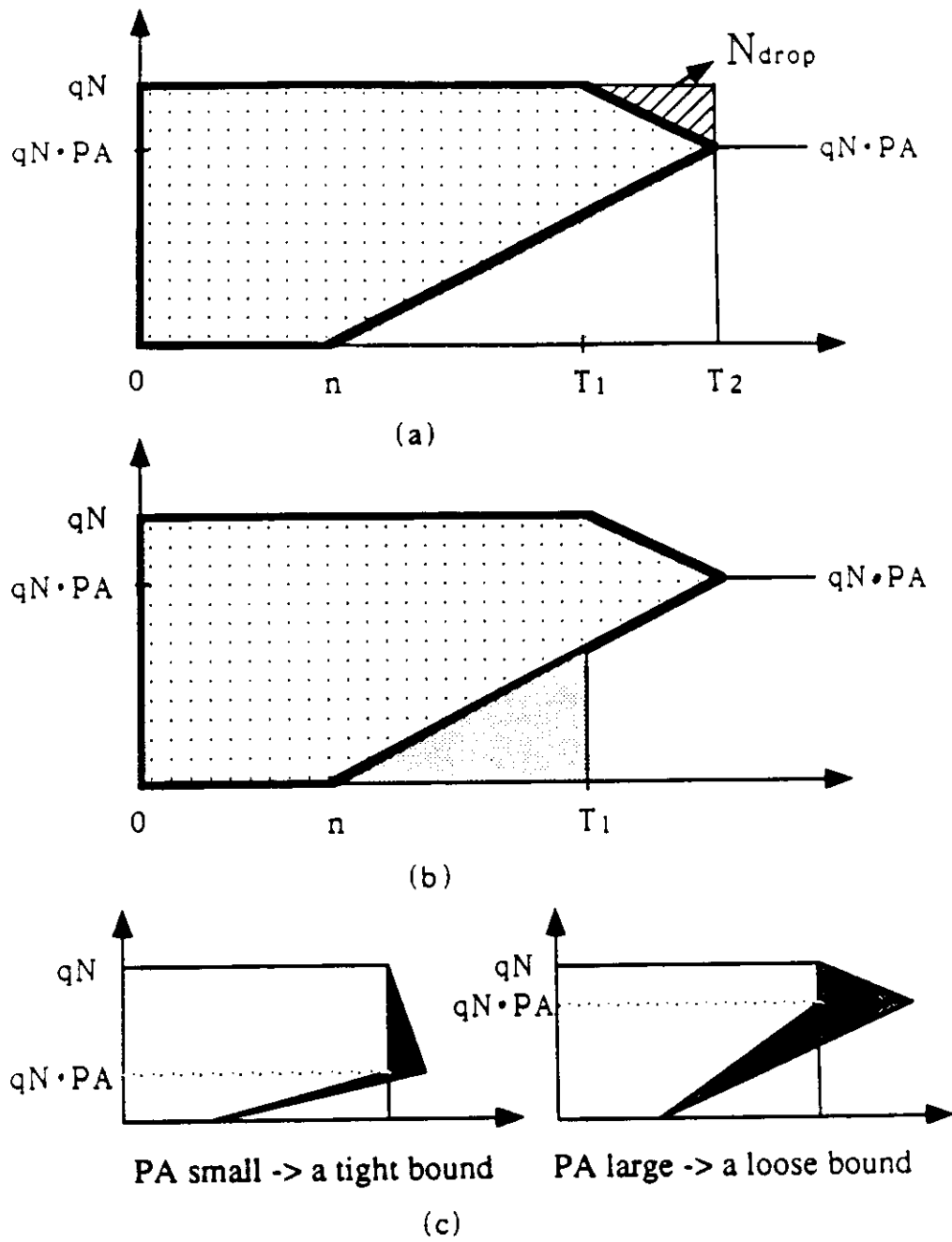


Figure 2.19: The Tree Build-up Time diagrams

number of discarded packets due to blocking at the first stage and  $\bar{N}_{out}$  is the average number of packets reaching their destinations. Starting from n-th cycle, packets begin to reach their destination. The output rate is increased until at cycle  $T_2$  when the output rate equals the input rate  $q \cdot N \cdot PA$ . This is the time when the system stabilizes.

Instead of calculating an exact value for  $T_1$ , we only come up with an upper bound on  $T_1$  as follows. In Figure 2.19(b),  $\bar{N}_o$  is the average number of packets that have reached the destination at time  $T_1$ . From the diagram, we come up with the following equation :

$$q \cdot N \cdot T_1 < \bar{N}_{MIN} + \bar{N}_o$$

$$q \cdot N \cdot T_1 < \bar{N}_{MIN} + \frac{q \cdot N \cdot PA}{2} \cdot (T_1 - n)$$

Hence, the upper bound for  $T_1$  is :

$$T_1 < \frac{2\bar{N}_{MIN} - q \cdot N \cdot PA \cdot n}{q \cdot N \cdot (2 - PA)} \quad (2.10)$$

The shaded area in Figure 2.19(c) represents the difference between  $q \cdot N \cdot T_1$  and  $\bar{N}_{MIN} + \frac{q \cdot N \cdot PA}{2} \cdot (T_1 - n)$ . Notice that when PA is small, the system is quickly saturated (hence a smaller shaded area) and thus we have a tight upper bound. When PA is large, we tend to have a loose bound. For a uniform light traffic load, the bound is not very accurate. However since our objective is to study the saturated tree with a moderate-to-heavy congestion (in this case, PA is small), equation (2.10) provides a good bound for  $T_1$ . Using this upper bound, we obtain the diagram of tree build-up time for different system parameters in Figure 2.20. For a uniform traffic case ( $r=0.5$ ), the only place where tree build-up occurs is when the offered load is 1 packet/cycle. Again, since PA is close to 1, we have a very loose upper bound. Taking (q,r) as a pair, there are 4 pairs.

(0.2,0.9),(0.3,0.8),(0.5,0.7) and (0.7,0.6) each representing the boundaries where the network is reaching a saturated tree (compare to Figure 2.15-2.18). Another comparison is presented with the result of average time delay in the following.

### Average Time Delay

One other interesting performance measure is the average time delay for a packet. In the buffered MIN, once a packet enters the network, it is guaranteed to reach its destination. But in addition to the necessary service time at each stage, one cycle per stage, buffering introduces waiting delays to the total system time. By using Little's formula, we get the average time delay. After reaching steady state, we calculate the average total number of packets in the MIN. Therefore, by applying Little's result, we get the average time delay for a packet:

$$T = \frac{\bar{N}_{MIN}}{N \cdot q \cdot PA} \quad (2.11)$$

The product  $N \cdot q$  is the total input rate generated. This value multiplied by PA is the actual input rate to the MIN in steady state. The  $\bar{N}_{MIN}$  represents the average total number of packets in the MIN. The result is shown in Figure 2.21.

For the uniform traffic case, the average time delay for small offered traffic is close to the necessary service time of 9 cycles. When the offered traffic increases to 1.0, due to buffering, the time delay increases to 40 cycles. Refer to Figure 2.18, when  $q=1.0$ , the average number of busy buffers is roughly 4.5. We notice that there is a trend to the curves. At first, the time delay rises as the offered traffic increases. At a certain value of offered traffic, the curve begins to level off. For example, this critical value of  $q$  for the  $r=0.9$  case is 0.2. After  $q=0.2$ , the time delay seems to reach a steady state. Even though the offered traffic is increased, the time delay does not increase much. The reason for this behavior is that the tree has been formed when  $q=0.2$  for the  $r=0.9$  case. Which means that,

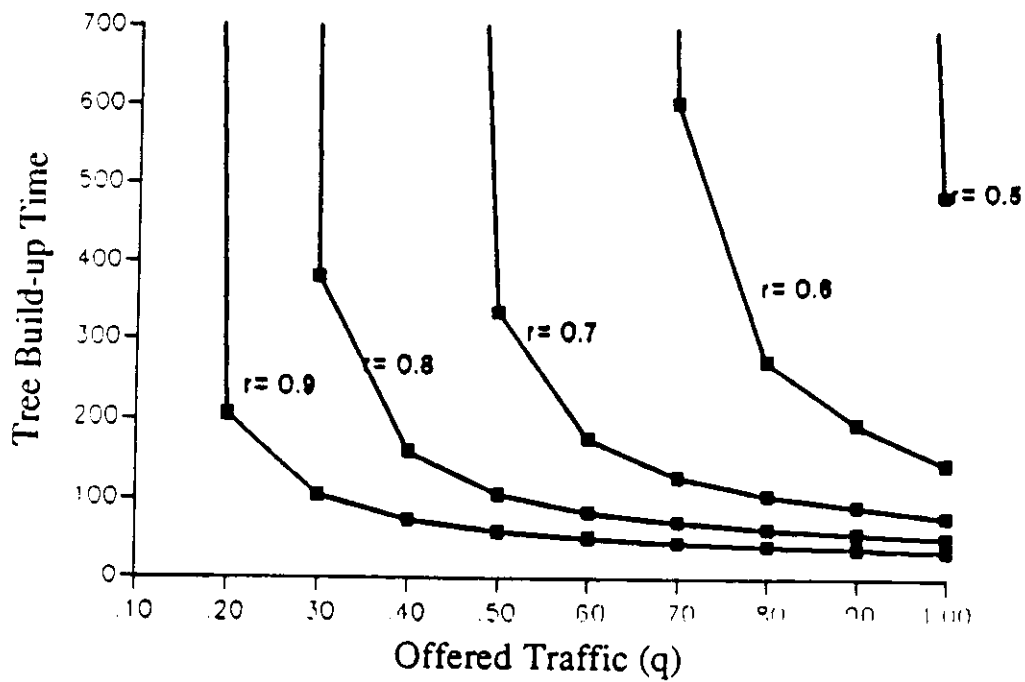


Figure 2.20: The upper bound of the tree build-up time for a 9-stage, 8 buffered interconnection network

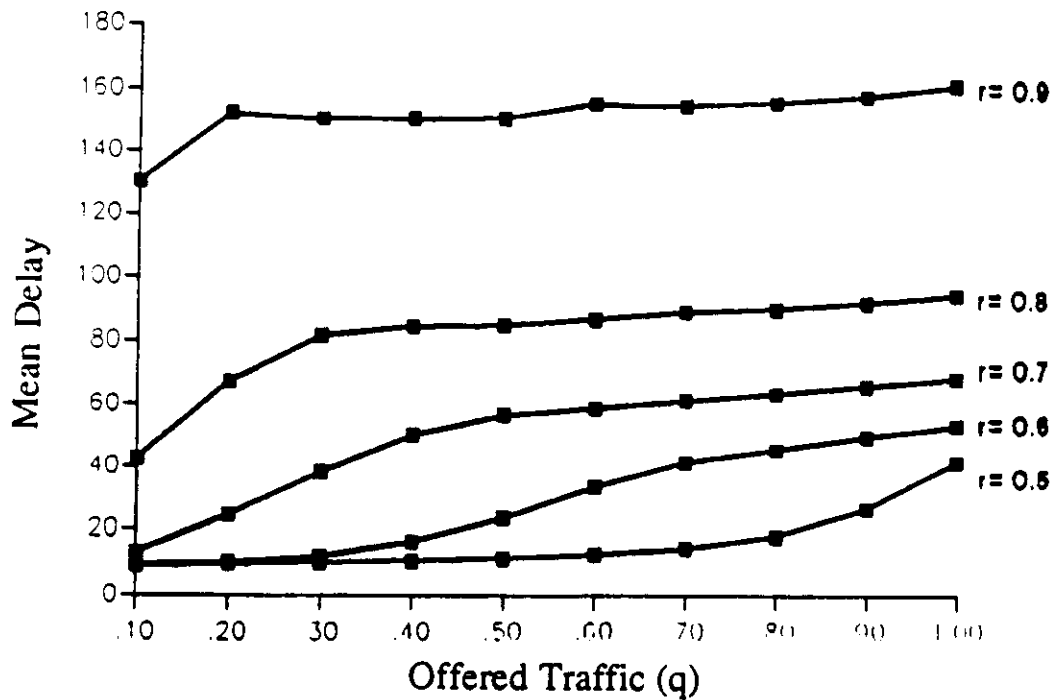


Figure 2.21: The mean delays for a 9-stage, 8 buffered interconnection network

starting from  $q=0.2$ , most of the packets are blocked at the first stage, and then discarded. Therefore the time delay does not change much by the increase of offered load. Similar critical points for the cases of  $r=0.8$ ,  $0.7$  and  $0.6$  are  $q=0.3$ ,  $0.5$  and  $0.7$ , respectively (compare to the points in Figure 2.20). There is a small amount of increase in time delay after the critical points for these values of  $r$  due to the further decrease in PA. These critical values are those points when PA is getting steady from a dramatic decrease due to tree saturation. We can predict for a large value of  $r$ , i.e. if the hot spot is very serious, that the tree is quickly saturated and reaches a steady state. Hence, the delay is not further increased due to the increase in offered traffic. With a small  $r$ , the tree is saturated slowly. PA can be further decreased by increasing  $q$  after the critical point is reached. The time delay is increased with a small amount after the critical point. The slope for this increase is larger for  $r=0.6$  than for  $r=0.7$ , and so on. This behavior is typical of queueing systems with finite buffers.

### 2.1.5 Model Verification

Our approach to analyzing the unbuffered MIN was to calculate the probability of a non-empty output port using a recurrence equation. No approximation method was used to solve for PA. Therefore, it is an exact solution to the model. Since the model is not used to approximate any real world application, no simulation was implemented to verify the error range of the model. If we were to write a simulation based on the same assumption, both results should be identical due to the exact solution.

However, the model of the buffered MIN is an approximate model. The approximation comes from the decomposition of coupled queues. Therefore, it is necessary to verify how good the model is to the real case. In our simulation, we



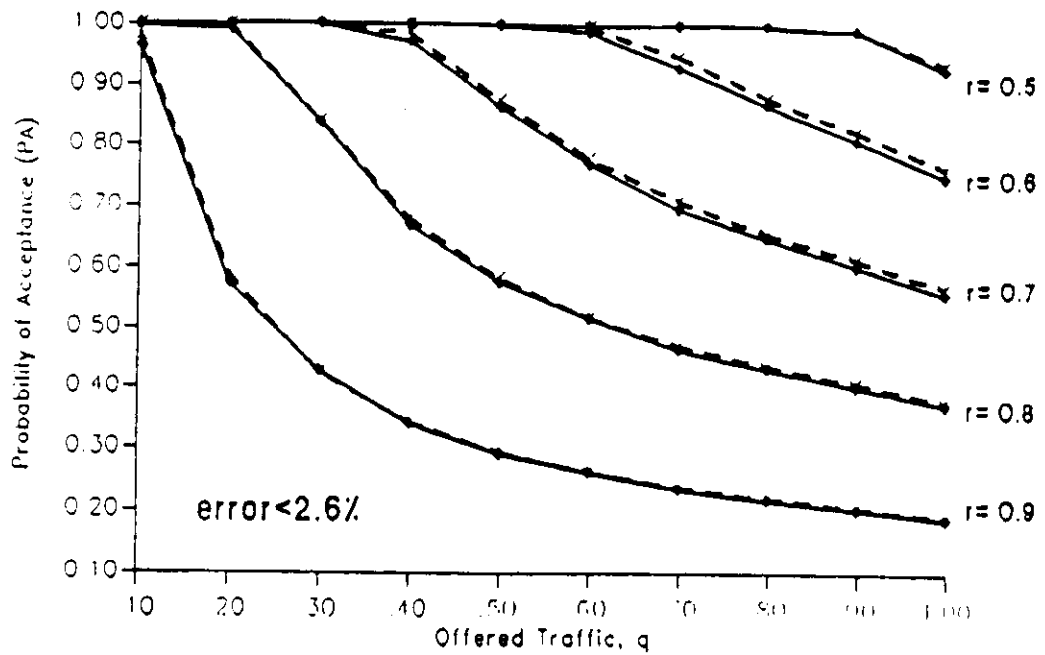


Figure 2.22: The analytical model vs. the simulation for a 9-stage, 8 buffered interconnection network. The dotted lines are simulations.

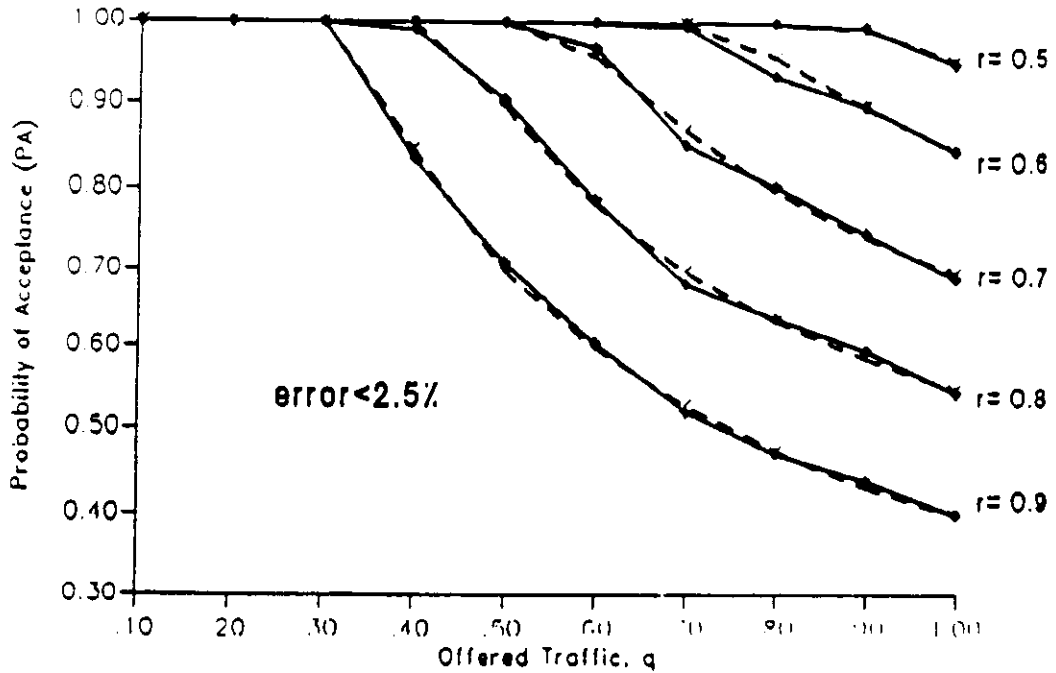


Figure 2.23: The analytical model vs. the simulation for a 2-stage, 8 buffered interconnection network

keep track of the state change of each queue at every stage. We begin with the output to MM's, then change the state of those queues in the last stage according to input process and output process at this stage. The procedure continues to the succeeding stages until stage 1. This procedure can be viewed as a time-sliced calculation of all queues in one cycle. We repeat this procedure for the first 2000 cycles in order to let the MIN reach steady state. Then we count the input packets and output packets starting with the 2001th cycle for another 3000 cycles, and calculate PA by dividing the number of output packets by the number of input packets. The results of simulation, as compared to the model results, are shown in Figure 2.22 for 9-staged, 8-buffered MIN. The error range for the model results are within 2.6%. Most points are within 1% for a broad range of values of  $r$  and  $q$ . A similar comparison for 2-staged, 8-buffered MIN is shown in Figure 2.23, with error range less than 2.5%. The simulation results show that the approximation is very good. For the small buffer case, the error range is within 2% for both large and small MIN. From the simulation, we are confident that the proposed model is a very good approximation to the coupled system.

### 2.1.6 Conclusion

We proposed models to solve for the unbuffered MIN and the buffered MIN. A routing model was suggested which makes systematic analysis possible for the performance of a MIN under hot spot traffic. The routing model covers uniform and non-uniform traffic cases, and provides a broad range of hot spot traffic patterns. The hot spot traffic pattern created from the routing model is different patterns studied by other researchers. Asymptotic behavior of the probability of acceptance for the unbuffered MIN was studied. An explicit expression was

given which governs the asymptotic behavior. The performance of the unbuffered MIN and the buffered MIN were compared for various offered traffic, stages and patterns. We also studied the evolution of tree saturation in terms of the average number of busy buffers in the queues of the tree. The average time delay of a packet was then found. The model of the buffered MIN was verified through simulation. The model result shows a very good agreement with the simulation data.

## 2.2 General Traffic Pattern Model

### 2.2.1 Introduction

One limitation of the previous model is that only specific output traffic patterns can be evaluated. This limitation is due to the choice of using a single routing probability  $r$  for all switching elements. Although the model can give us some insight as to how hot spots affect the system performance, it is not good enough. In reality, we are usually given a traffic pattern which does not conform to our special traffic patterns. Indeed, the traffic pattern can have two or three hot spots. In this section, we investigate the particular characteristics of the previous model in order to extend it to handle general traffic patterns.

The way that we evaluated the finite-buffered MIN in section 2.1 was to decompose the network of blocking queues into independent queues with equivalent input rates and blocking probabilities. The final solution was found by repeating this decomposition process for all switches until the value of PA converged. Since each queue is evaluated independently, the restriction of using a single routing probability  $r$  for all switches in all stages seems unnecessary. Hence let us now relax this requirement by allowing the value of  $r$  to vary in every stage. In section

2.2.2, we present the method to transform a given traffic pattern into a set of values of  $r_{i,j}$  such that after traffic is fed into the MIN, the routing probabilities in different switches and stages creates the exact given traffic pattern. However, we will not show the resulting model at this point since there are some other limitations that need to be relaxed. We will incorporate this transformation method with a superposition method (to be discussed in section 3.1) to model the situation where processing elements have different traffic patterns. Then an analysis of the NUTS traffic pattern (to be discussed in section 3.2) is presented as an example.

### 2.2.2 Transformation Method

The model assumptions are the same as in section 2.1 except that each queue can have a different routing probability. Since we want to transform a given traffic pattern into routing probabilities, the components of the pattern must be given in a numerical form such that they can be computed and transformed into routing probabilities in the switches. Hence, we specify the given traffic pattern to be the set of MM accessing probabilities of a processing element. The sum of these accessing probabilities, from MM(0) to MM(N-1), is equal to one. With the assumption that all PEs have the same traffic pattern, we only need one transformation to solve for the routing probabilities. If PEs have different traffic pattern, then the transformation is performed as many times as the number of different patterns(see 3.1.2).

#### 2.2.2.1 The First Attempt

One way to approach the generalization is to let the switching elements in each stage  $i$  have a routing probability  $r_i$  of choosing the 0-output port. A packet,

with  $n$  0's in its destination address, will choose  $n$  consecutive 0-output ports to go to MM(0). In terms of routing probabilities, the probability of accessing MM(0), namely  $A_0$ , is :

$$A_0 = \prod_{i=1}^n r_i$$

Similarly, the probability of accessing MM(1) is :

$$A_1 = (1 - r_n) \cdot \prod_{i=1}^{n-1} r_i$$

Therefore given  $A_0$  and  $A_1$ , we are able to solve for  $r_n$  by dividing the two accessing probabilities :

$$\frac{A_0}{A_1} = \frac{r_n}{1 - r_n}$$

$$r_n = \frac{A_0}{A_1 + A_0}$$

Given the accessing probabilities  $A_0$  to  $A_{N-1}$ , we choose the  $n$  largest ones among these  $N$  values ( $N = 2^n$ ), and transform them using the same technique as shown to solve for routing probabilities in every stage. The limitation of this method is that we have to choose the  $n$  largest among the  $N$  values. This implies that this method is only an approximation. When the  $N$  values are very close, there will be a significant discrepancy. Hence the question remains as how good is the approximation. There will be certain conditions under which the approximation works well. We must adjust the values of  $r_i$  to further reduce the discrepancy. From this argument we simply state that this method extends the specific traffic pattern somewhat further, but not to a very general extent.

Although this attempt did not solve the problem properly, it did inspire us to the next approach which uses a similar transformation method but with more generality.

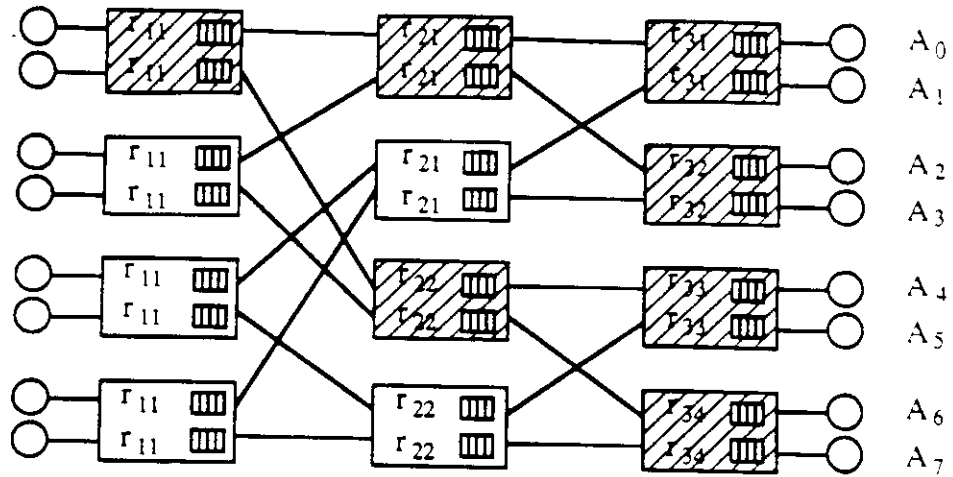


Figure 2.24: A General memory referencing pattern shown in terms of accessing probabilities  $A_j$

### 2.2.2.2 The General Traffic Pattern (GTP) Transformation

One problem of the first attempt is that we neglected the other  $N-n$  accessing probabilities, i.e. not all of the given constraints are satisfied. One characteristic of the routing is that before the first stage, all packets look the same. After the first stage routing, they break into two different groups of packets. One has the first address bit 0 and the other has the first address bit 1. The group with the first address bit 0 is routed to the upper output ports of switching elements in the first stage. According to Banyan interconnection, they are fed to the first half of the switching elements in the second stage. The group with the first address bit 1 is fed to the second half of the switching elements in the second stage. After being routed in the second stage, packets are divided into four groups, and so on. Hence, we put two routing probabilities in second stage, four routing probabilities in the third stage, and so on. In such a way, we identify  $N$  accessing probabilities (actually  $N-1$  values since the sum of  $N$  accessing probabilities is 1) to solve for  $N-1$  unknowns.

Let us take a 3 stage Banyan network as an example, as shown in Figure 2.24. Observe the path taken by a packet, generated by processing element 0. If there exists a steady state output referencing pattern, given in terms of accessing probabilities  $A_j$ , then a packet chooses memory module 0 with probability  $A_0 = r_{11} \cdot r_{21} \cdot r_{31}$ . Similarly, a packet chooses memory module 1 with probability  $A_1 = r_{11} \cdot r_{21} \cdot (1 - r_{31})$ . Using these two equations, we can find  $r_{31}$  in terms of  $A_0$  and  $A_1$ .

$$r_{31} = \frac{A_0}{A_0 + A_1}$$

The other routing probabilities can be found in a similar way :

$$r_{32} = \frac{A_2}{A_2 + A_3}$$

$$r_{33} = \frac{A_4}{A_4 + A_5}$$

$$r_{34} = \frac{A_6}{A_6 + A_7}$$

$$r_{21} = \frac{A_0 + A_1}{A_0 + A_1 + A_2 + A_3}$$

$$r_{22} = \frac{A_4 + A_5}{A_4 + A_5 + A_6 + A_7}$$

$$r_{11} = \frac{A_0 + A_1 + A_2 + A_3}{A_0 + A_1 + A_2 + A_3 + A_4 + A_5 + A_6 + A_7}$$

Thus we solve the simultaneous equations for  $r_{i,j}$ , we implement the GTP transformation method in the following recursive PASCAL program :

```
procedure traffic-trace(s, posi, count : integer; var psum : real);
```

```
var
```

```
    fl, portt : integer;
```

```
    denom, nom : real;
```

```

begin
    count:=count+1;
    s:=s div 2;
    if posi > (penum div 2) then fl:=1 else fl:=0;
    posi:=port(posi);
    if fl=1 then portt:=posi-1 else portt:=posi;
    if count < stagen then
        begin
            traffic-trace(s, portt, count, psum);
            denom := psum;
            traffic-trace(s, portt+1, count, psum);
            nom:=denom+psum;
        end
    else
        begin
            denom:=pac[portt];
            nom :=denom+pac[portt+1];
        end;
        if denom <> 0 then r[count,posit]:=denom/nom;
        psum:=nom;
    end;

```

Thus we fully utilize all the given information to solve for the routing probabilities. By applying this transformation method on top of the framework of section 2.1, we can evaluate any given output pattern for the probability of ac-



ceptance and the average system time. However, we are still restricted to the case where all PE's use the same MM referencing pattern. This we relax in the next chapter.

## CHAPTER 3

### Modeling of Processing Elements with Different Traffic Patterns

The analytical model in Chapter 2 assumes that all processing elements have the same traffic pattern. However, it might not always be true in real world applications. It is more likely that each processing element has its own traffic pattern and offered load. In section 2.2, we presented a transformation method to model a general output traffic pattern where all processing elements have the same general output traffic pattern. In this Chapter, we generalize the traffic conditions such that each processing element has its own output traffic pattern and offered load. The basic model is presented in section 3.1 where a superposition method is proposed to model different traffic patterns for processing elements with the same offered load. Section 3.2 applies both methods (transformation from section 2.2 and superposition from section 3.1) to evaluate system performance of the Not Uniform Traffic Spot traffic pattern [Lang 88]. In section 3.3, we propose a method, which incorporates a weighting factor into the transformation and the superposition methods, to model the case where each processing element has its own offered load. This represents the most general case in traffic condition.

#### 3.1 Basic Model

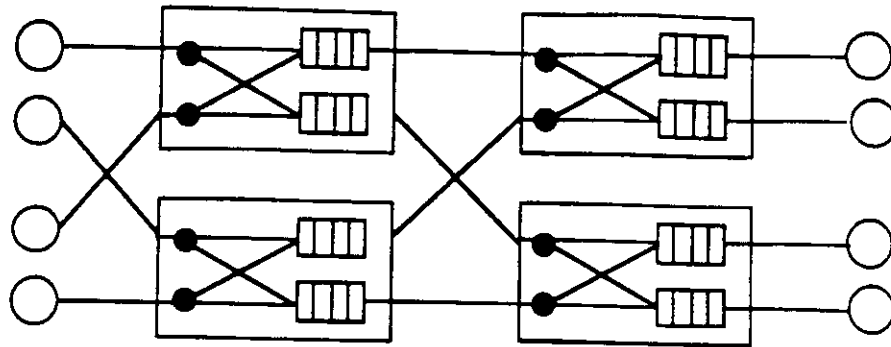
##### 3.1.1 Problem Characteristics

Using the transformation method in section 2.2 to transform each traffic pattern individually for all  $N$  PE's, we get  $N$  sets of routing probabilities. Depending

on where it comes from, a packet chooses the output port according to the specific routing probability. Therefore, the modelling approach for the Different Traffic Pattern Model is to determine the appropriate overall routing probability for each switching point. Since there are many traffic flows, each with its pattern, we keep track of all possible paths taken by each traffic flow. Take a 2-stage, 4x4 interconnection network for example, as shown in Figure 3.1(a). Let  $r_k[i, j]$  be the routing probability of the  $j$ th switching point at stage  $i$  for packets generated from the processing element  $k$ . Using the processing element 1's traffic pattern, the transformation method that we proposed in section 2.5 generates  $r_1[i, j]$ 's. These  $r_1[i, j]$  are assigned to the switching points along the paths that packets from the processing element 1 take (as shown in Figure 3.1(b)). The traffic pattern of the processing element 2 is then transformed into a set of routing probabilities,  $r_2[i, j]$ 's. They are assigned to the proper switching points accordingly (as shown in Figure 3.1(c)). The routing probability sets of the processing elements 3 and 4 are solved for and are assigned to the routing matrix in Figure 3.1 (d) and (e). Next we calculate the routing probabilities of each switching point,  $r[i, j]$ , as follows :

$$\begin{array}{ll}
 r[1,1] = r_1[1, 1] & r[2,1] = \frac{r_1[2,1]+r_3[2,1]}{2} \\
 r[1,2] = r_3[1, 2] & r[2,2] = \frac{r_2[2,2]+r_4[2,2]}{2} \\
 r[1,3] = r_2[1, 3] & r[2,3] = \frac{r_1[2,3]+r_3[2,3]}{2} \\
 r[1,4] = r_4[1, 4] & r[2,4] = \frac{r_2[2,4]+r_4[2,4]}{2}
 \end{array}$$

The routing probability matrix in Figure 3.1(f) is the overall routing probability matrix for the switching elements.



(a)

	stage 1	stage 2
1	$r_1[1,1]$	$r_1[2,1]$
2		
3		$r_1[2,3]$
4		

(b)

	stage 1	stage 2
1	$r_1[1,1]$	$r_1[2,1]$
2		$r_2[2,2]$
3	$r_2[1,3]$	$r_1[2,3]$
4		$r_2[2,4]$

(c)

	stage 1	stage 2
1	$r_1[1,1]$	$r_1[2,1] \ r_3[2,1]$
2	$r_3[1,2]$	$r_2[2,2]$
3	$r_2[1,3]$	$r_1[2,3] \ r_3[2,3]$
4		$r_2[2,4]$

(d)

	stage 1	stage 2
1	$r_1[1,1]$	$r_1[2,1] \ r_3[2,1]$
2	$r_3[1,2]$	$r_2[2,2] \ r_4[2,2]$
3	$r_2[1,3]$	$r_1[2,3] \ r_3[2,3]$
4	$r_4[1,4]$	$r_2[2,4] \ r_4[2,4]$

(e)

	stage 1	stage 2
1	$r[1,1]$	$r[2,1]$
2	$r[1,2]$	$r[2,2]$
3	$r[1,3]$	$r[2,3]$
4	$r[1,4]$	$r[2,4]$

(f)

Figure 3.1: (a) A 4x4 network with 4 switching points (•) in each stage. (b)-(f) routing probability matrix

Unlike the result in section 2.2 where there is only one routing probability for each switching element, we now have two routing probabilities for each switching element, each for one switching point. The reason is that when all PEs have the same pattern, the routing probabilities for both switching points in the switching element are the same. Therefore there is no need to distinguish between them. However, when the processing elements have different traffic patterns, the situation above is no longer true, and we have to distinguish between them in order to keep track of the correct traffic flows.

### 3.1.2 Superposition Method

According to the discussion in the previous section, we take a traffic pattern of a PE and route a packet through all possible paths. Then the routing probabilities are determined using the transformation method. The routing probabilities are accumulated at switching points along the paths that packets take. By repeating this transformation and accumulation process for all PEs (for their traffic patterns, respectively), we then take the mean of the routing probabilities to find the proper routing probabilities set to approximate the given traffic patterns. We present the superposition method in the following program :

```
procedure traffic-trace(s, posi, count : integer; var psum : real);  
var  
    f1, portt : integer;  
    denom, nom : real;  
begin  
    count:=count+1;  
    s:=s div 2;
```

```

if posi > (penum div 2) then fl:=1 else fl:=0;
posi:=port(posi);
if fl=1 then portt:=posi-1 else portt:=posi;
if count < stagen then
    begin
        traffic-trace(s, portt, count, psum);
        denom := psum;
        traffic-trace(s, portt+1, count, psum);
        nom:=denom+psum;
    end
else
    begin
        denom:=pac[portt];
        nom :=denom+pac[portt+1];
    end;
if denom <> 0 then
    begin
         $c[count, posi] := c[count, posi] + 1;$ 
         $r[count, posi] := r[count, posi] + denom/nom;$ 
    end ;
if (denom=0) and (nom <> 0) then  $c[count, posi] := c[count, posi] + 1;$ 
    psum:=nom;
end;

```

This procedure is basically the same as the one we presented in section 2.5 except that we put in some extra lines (shown in *italic font*) to account for the accumulation of the routing probabilities. Each switching point has two variables:  $c[i,j]$  is the counter for the  $j$ th switching points in stage  $i$  and  $r[i,j]$  is the accumulated routing probability whose mean will be the probability for a packet choosing the 0-output queue in steady state.  $c[i,j]$  is increased by one if there is traffic from a processing element passing through this switching point. However,  $r[i,j]$  is accumulated only when the 0-output queue is on the path taken by the packets from a processing element whose traffic pattern is currently being transformed. The variable *denom* is the amount of traffic from the previous stage passing through a 0-output queue, and the variable *nom* is the traffic coming from the previous stage (not necessarily passing through this queue). If *denom* is not 0, then there is traffic from the previous stage passing through this 0-output queue. We increment the counter by 1 and accumulate the routing probability. If *denom* is 0 and *nom* is not 0, then all traffic coming from the previous stage goes to the 1-output queue in the switching element, we only increment the counter without accumulating any routing probability. If both variables are 0, then there is no traffic coming from the previous stage, and hence there is no need to increment the counter or to accumulate the routing probability.

For every PE we call the procedure **traffic-trace** to trace all possible paths and accumulate the routing probability along the path. After each PE's traffic pattern is transformed, we divide the accumulated routing probability by the counter for each switching point individually. The control flow is represented in the following program :

```
for i:=1 to penum do
```

```

begin
    pattern-bitr (i-1);
    s:=i-1;
    psum:=0;
    posi:=i;
    traffic-trace (s,posi,0,psum);
end;
for j:=1 to penum do
begin
    for i:=1 to stagen do
    begin
        if tc[i,j]=0 then r[i,j]:=0 else r[i,j]:=tr[i,j]/tc[i,j];
    end;
end;
end;

```

The procedure **pattern-bitr** contains the MM accessing probabilities of each processing element. This given traffic pattern of a processing element is transformed (by using procedure **traffic-trace**) into a routing probabilities set. The routing probabilities are accumulated at switching points which the traffic from the PE passes. When all the processing elements' traffic patterns are transformed into routing probabilities and these values are accumulated at switching points, we take the mean of these routing probabilities to determine the routing probabilities matrix which approximates the steady state behavior of the given traffic patterns. Then this routing probability matrix is used in the decomposition and iteration model that we proposed in Chapter 2 to evaluate any general traffic



condition. We shall use this method in the next section to solve for the NUTS traffic patterns.

## 3.2 Not Uniform Traffic Spots model

### 3.2.1 Introduction

The model in section 2.1 deals mainly with a hot spot traffic pattern in which a favorite MM creates a saturated tree. However, if we let each PE have a different traffic pattern, it is possible to have a system whose pattern shown at the MM side is uniform, but with some congested spots inside the interconnection network. This behavior was first discussed in [Lang 88], and was named the Not Uniform Traffic Spots (NUTS). The problem is completely different from the traditional hot spot problem in that there is no hot MM which attracts traffic. It is the pattern of each PE that creates conflicting paths inside the network. Among the possible patterns, a few were discussed as examples in [Lang 88]: bit reversal, Even-First, Odd-Second (EFOS). In a bit reversal pattern, each processing element send all its traffic to a memory module whose address is the bit-reversed address of the PE. In an EFOS pattern, all even-addressed PE's send packets uniformly to the first half of the MM's and all odd-addressed PE's send their packets uniformly to the second half of the MM's. The simulator result of these patterns are shown in the paper for 6 stage Omega-MIN with buffer size 4. However, due to the limited speed and capacity, the simulator is not suitable for analyzing a large network.

In section 3.2.2, we present an analytical model using the transformation method and superposition method to evaluate the NUTS patterns. The analytical results of EFOS, bit-reversal and uniform traffic are compared to Lang's

simulator data. Model verification is discussed in section 3.2.3. The simulation indicates that the analytical model is too optimistic. The cause of the discrepancy is discussed and verified through simulation. In Chapter 4, an approximation method is used to resolve the discrepancy.

### 3.2.2 Modelling Approach

The NUTS traffic pattern is basically a general traffic pattern such that each PE has a different traffic pattern. The control flow of the model is shown as follows :

```

PROGRAM General Traffic Pattern;
begin
    input stage number, buffer size;
    for all PEs do
        call procedure pattern-bitr; /*assign traffic patterns*/
        call procedure traffic-trace; /*transformation method*/
    endfor;
    for all ij do /*superposition method*/
         $r[i,j] := r[i,j] \text{ div } c[i,j]$ ;
    endfor;
    call procedure buffer-min /*solve for PA, time delay*/
end;
```

The procedure **pattern-bitr** and **traffic-trace** are explained in previous sections. The procedure **buffer-min** is the model in section 2.1.4 with routing probability  $r$  substituted by  $r[i, j]$  to handle the general traffic conditions. The

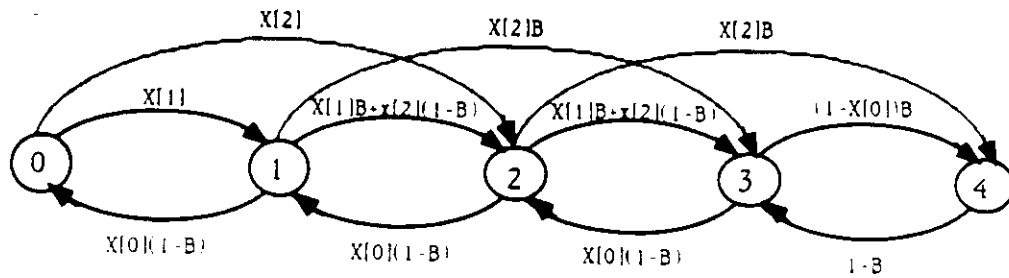


Figure 3.2: Markov chain of the queuing process where the state variable represents the number of packets in the queue

program begins with initializing system parameters. Then we apply the given traffic patterns of all PEs to create the routing probabilities for the model. Using the parameters and the routing probabilities, we decompose the network of blocking queues and solve for their steady state probabilities. This process is iterated until PA converges.

In order to compare the result of the analytical model with the simulator data in [Lang 88], we have to make sure the assumptions in both cases are the same. There are two different assumptions :

(1) **input process** In section 2.1.4, a concurrent handling of input/output was assumed, in which two input packets and an output packet can move in and out of the queue at the same time. However, the simulator assumes that if two incoming packets find only one available buffer space, only one is accepted regardless of what the output process might be (blocked or not). If the buffer is full, no packets will be accepted at any time. These rules, together with the blocking probability and the input probabilities (see section 2.1.4.1), determine the Markov chain of a queue decomposed from the network. The Markov chain is shown in Figure 3.2 where  $B$  is the blocking probability and  $X[i]$  is the probability that  $i$  packets are entering the queue.

(2) **output choice** The model in section 2.1.4 assumes that in every cycle, the packet at the head of the queue chooses output ports according to the routing probability  $r_{i,j}$  independent of the choice in the previous cycle. A new decision is made in every cycle. However, it is not the case in a real system. The simulator model assumes that, if a packet is blocked in the previous cycle, it will choose the same output in the following cycles until it gets through. This implies that a blocked packet has a "memory" in choosing the output. In the following model, we temporarily make the renewal assumption. This renewal assumption makes the model result optimistic, and thus causes a discrepancy when compared to the simulation. Further analysis and modeling of this problem is to be discussed in Chapter 4.

### 3.2.2.1 Bit-Reversal Model

When every source sends its packets to the destination whose address is its own address bit-reversed, it creates a traffic pattern that is uniform at the memory modules, but congested inside the network. A source with address 0011010, for instance, sends all its packets to the destination with address 0101100. From the MM sides, it looks like a uniform pattern since no hot MM exists. Every MM receives the same amount of traffic. But the routing paths resulting from this particular pattern are overlapped. Several hot traffic spots are formed inside the switches. The throughput is thus significantly reduced. Given the source address  $s$ , the following procedure **pattern-bitr** creates the bit reversal pattern. It first transforms a PE's address into binary form bit by bit. The destination address is calculated using these bits, and the value is assigned to the variable *temp*. Then the accessing probability  $pac[temp + 1]$  is set to one, with all other accessing probabilities set to zero. This procedure is used in the control flow

shown in section 3.1.2 to set the accessing probabilities for the specific pattern that we want to model.

```

procedure pattern-bitr(s : integer); /* Bit-Reversal Pattern */
var
    i, bit, temp : integer;
begin
    for i:= 1 to penum do
        pac[i]:=0;
    temp:=0;
    for i:=1 to stagen do
        begin
            bit:=s-(s div 2) *2;
            s:=s div 2;
            temp:=temp+expni(2,stagen-i)*bit;
        end;
        pac[temp+1]:=1;
    end;

```

### 3.2.2.2 EFOS Model

The EFOS traffic pattern is created by routing packets from the even-addressed source to the first half of the destinations uniformly and packets from the odd-addressed source to the second half of the destinations uniformly. For example, in a 64x64 MIN, the processing elements with the addresses 2, 4, ... 64 send packets to the memory modules 1, 2, ... 32 with an accessing probability  $\frac{1}{32}$ . The processing elements with the addresses 1, 3, ..., 63 send packets to the the

memory modules 33, 34 ..., 64 with an accessing probability  $\frac{1}{32}$ . The following procedure creates the EFOS traffic pattern. By placing this procedure in the General Traffic Pattern program, we are able to evaluate system performance of the EFOS traffic pattern.

```
procedure pattern-bitr(s : integer);
var
    i, temp : integer;
begin
    for i:= 1 to penum do
        pac[i]:=0;
    temp:=penum div 2;
    if (s mod 2)=0 then
        begin
            for i:=1 to temp do
                pac[i]:=1/temp;
            end
        else
            begin
                for i:=temp+1 to penum do
                    pac[i]:=1/temp;
                end;
            end;
    end;
```

### 3.2.3 Model Results and Verifications

Since the proposed analytical model employs several approximate methods, it is important to study how these approximations affect the model accuracy. There are two approximations in the modelling approach :

- decomposing a queue from a network of queues with blocking to be an independent queue.
- using a general routing matrix to model the steady state flows.
- renewal routing probabilities.

The first approximation is obvious since dependent queues are decomposed into equivalent independent queues and solved individually. Some accuracy is lost because our model neglects the dependency and coupling among the queues. The second approximation method simulates the steady state behavior of a traffic pattern with a routing probability set. Hence packets are routed according to this routing probability set in the model, instead of their address tags in the real world. The third approximation allows packets to choose their output ports independently at every cycle according to the routing probabilities. This renewal routing choice allows a blocked packet to choose a different output port in the next cycle. This renewal assumption renders the analytical model optimistic since it "allows" blocked packets to be routed around a congested queue. In the real world, blocked packets repeatedly access the same destination, and most likely, these blocked packets will be blocked again (especially when the traffic is not uniform).

The results of the analytical model for a 64 x 64 Omega MIN with buffer size 4 are plotted in Figure 3.3 along with the simulator data from [Lang 88].

All curves are shown with the input rate starting from 0.1 to 1.0 except for the bit-reversal case where extra 9 points between 0.1 and 0.2 are plotted. Both the analytical result and the simulator result for the bit-reversal pattern are nearly the same. The bit-reversal pattern severely reduces system throughput to 0.125. As predicted, the analytical model is very optimistic due to the independent routing choices it allows. When severe blocking is presented due to contention, the blocked packets will choose the same output queues repeatedly in the real world while the renewal choice in the analytical model allows the blocked packets to choose other queues. This inherited "memory" structure in blocking switches severely degrades the performance since it is likely to have persistent contention for a queue once a contention occurs. The discrepancy between the analytical result and the simulation data is caused mainly by this memory characteristic of the blocking switch. In the simulator data, however, a packet blocked in a previous cycle will be sent to the same output port. And thus the time delay will be higher, and the throughput will be lower than the analytical model with the renewal assumption.

To verify the conjecture in explaining the discrepancy, we first verify the correctness of the analytical model. We then change the assumption regarding the output choice in the simulation (i.e. to account for the "memory" behavior of a packet). If the revised simulation agrees with the simulator data, then we can be sure that the discrepancy indeed comes from the different assumption about the output choices. To verify the general traffic pattern model, the simulation program in section 2.1.5 is not sufficient since only one value of the routing probability was used. With the simulation program revised according to the assumptions in section 3.2.2, we will be able to verify the analytical model by adding a 2-dimensional array of the routing probabilities  $r[i, j]$ . For detailed



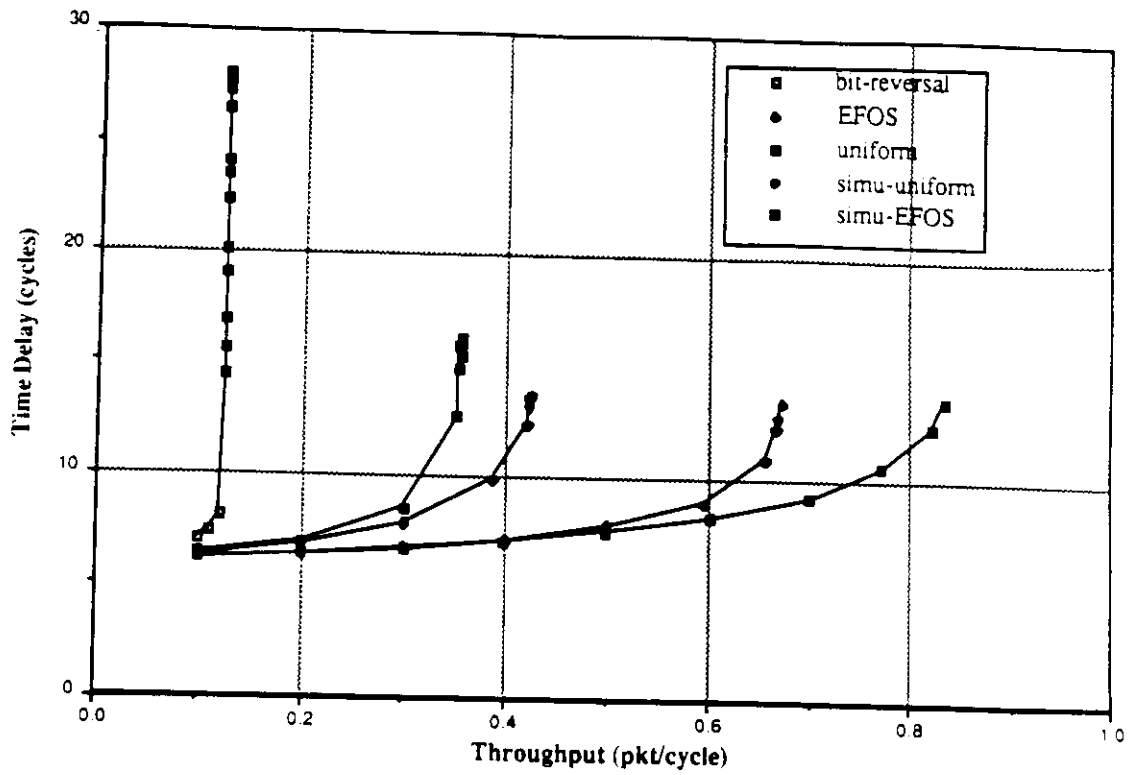


Figure 3.3: Throughput-Delay performance from analytical model vs. simulator data

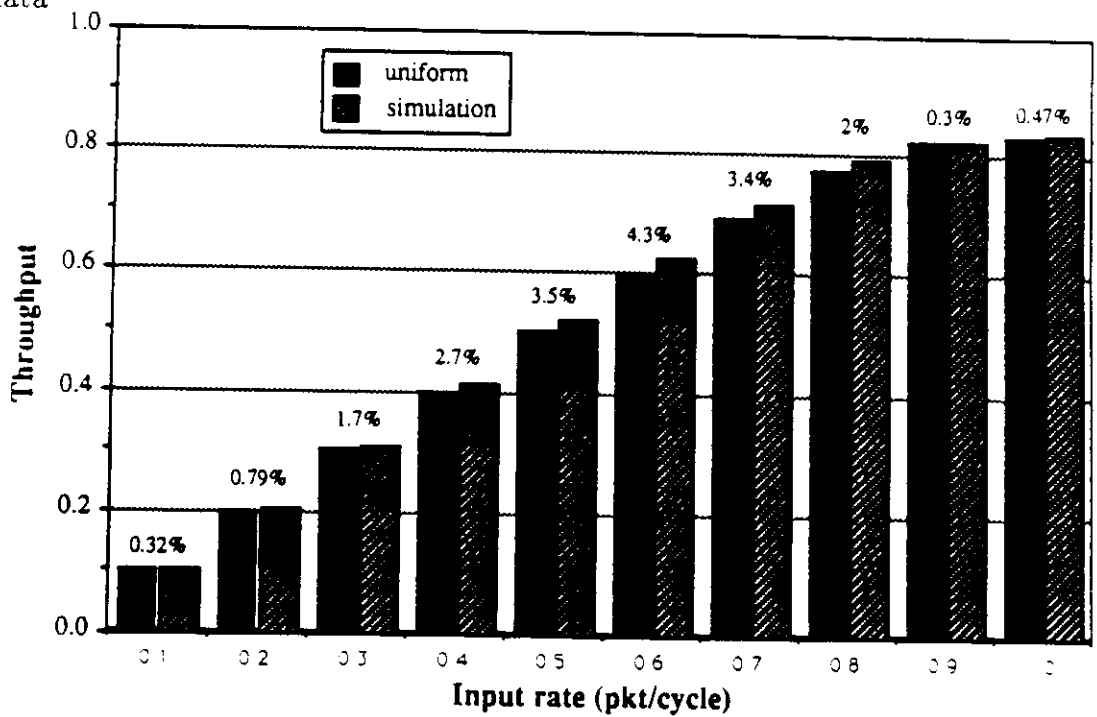


Figure 3.4: Throughput comparison for a 64x64 Omega MIN under uniform traffic

algorithm of how the simulation works, refer to section 2.1.5.

Figure 3.4 shows the throughput comparison of the analytical model and the simulation program based on the same assumption. The simulation results are shown to be in good agreement with the analytical model. The correctness of the model has been confirmed. Next, we change the output choice assumption in the newly revised simulation to account for the fact that packets do memorize their previous destination addresses. Again, good agreement was obtained in comparing the simulation and the simulator result. This proves the conjecture that we made. As a result, we claim that the analytical result in section 3.2.2 is optimistic in heavy load. We would try to use another approach in Chapter 4 to model the switch with "memory".

### 3.3 Different Input Rate Model

This is the most general model of traffic patterns in that every PE has its own input rate. In real world, some PEs may be very active while some are not active at all. Hence, different input rates and different traffic patterns of the PE's may give a completely random form of generalization. The modelling approach considers the weighting factor when we compute the routing probabilities. The following procedure **traffic-trace** is taken directly from section 3.1.2 with the addition of the weighting factor  $rate(i)$ , the new variable introduced representing the input rate of PE(i) :

```
procedure traffic-trace(s, posi, count : integer; var psum : real);  
var  
    fl, portt : integer;  
    denom, nom : real;
```

```

begin
    count:=count+1;
    s:=s div 2;
    if posi > (penum div 2) then f1:=1 else f1:=0;
    posi:=port(posi);
    if f1=1 then portt:=posi-1 else portt:=posi;
    if count < stagen then
        begin
            traffic-trace(s, portt, count, psum);
            denom := psum;
            traffic-trace(s, portt+1, count, psum);
            nom:=denom+psum;
        end
    else
        begin
            denom:=pac[portt];
            nom :=denom+pac[portt+1];
        end;
    if denom <> 0 then
        begin
            c[count, posi]:=c[count, posi]+rate(i);
            r[count, posi]:=r[count, posi]+(denom/nom)*rate(i);
        end ;
    if (denom=0) and (nom <> 0) then c[count, posi]:= c[count, posi]+rate(i);
    psum:=nom;

```

**end;**

The weighting factor  $rate(i)$  is shown in bold face. The reason we choose  $rate(i)$  as the weighting factor is that the amount of the traffic affects the routing probability. A traffic from a heavy loaded PE is an influential factor in the routing probability. A traffic from a lightly loaded PE is less influential than the former. Hence, it becomes an appropriate choice for the weighting factor. Whenever we compute a routing probability from a particular source, before accumulating the value, we multiply the value by the weighting factor to reflect the appropriate share of the load it brings to this switch. After we repeat this procedure for all PEs, we get the appropriate routing probabilities by dividing the accumulated values by the counter. Hence, by placing this procedure in the general traffic pattern model in 3.1, we can solve for the different input rate model.

The method for modeling different input rates will simply be listed here without any example. The reason is that this method is straight forward and there is no need to model any real system to prove its correctness.

## CHAPTER 4

### Persistent Blocking Model

#### 4.1 Modelling Approach

Since the basic model is a renewal process, we continue to model the memory behavior as a renewal process. However, the behavior of a blocked packet, after its first blocking, is such that the routing choice no longer obeys the renewal probability  $r_{i,j}$ . Biasing the routing probabilities to account for this does not help since it changes the memory referencing pattern. The routing probabilities were created to reflect the steady state memory referencing pattern; therefore, it is necessary to keep the values unchanged.

Although an exact model of this persistent blocking behavior would require that we keep track of how many times a packet has been blocked at a given node, we choose an approximation which captures the "first order" effect of this persistent blocking using the following two state model : when the queue is not empty, we model the server as being either in the "new" state or the "blocked" state. When a packet first comes into the server, the server is in the new state. The server enters the blocked state when the packet is blocked, and it remains in the blocked state until the blocked packet finally goes through to the next stage. This cycle repeats until the server empties the queue and becomes idle. Observe that the server is inactive when it is in blocked state. During the new state, the server obeys the renewal behavior choosing an output port according to the routing probability  $r_{i,j}$ . Hence, we can approximate a blocking switch

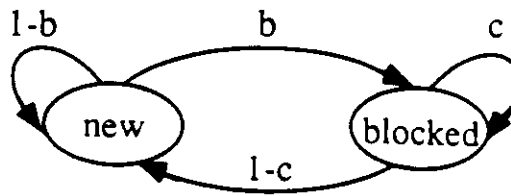


Figure 4.1: The states of a server during its busy period.

with "memory" characteristics by a finite buffer queue with a reduced service rate. The reduced portion is the probability that the server is in the blocked state.

The diagram in Figure 4.1 shows how the server alternates between the new state and the blocked state during its busy period. Let  $b$  be the probability that a new packet is blocked when it tries to go to the next stage. Let  $c$  be the probability that a blocked packet is blocked again when it tries to go to the same destination. Then for our approximation, the steady state probability that the server is in the blocked state,  $P_{blocked}$ , can be solved in terms of  $b$  and  $c$  :

$$P_{blocked} = \frac{b}{1 - c + b}$$

where  $b$  is the blocking probability (for which we used the notation  $B_{i,j}$  in section 2.1.3). Once blocked, it is more likely that a blocked packet gets blocked again; therefore, the value of  $c$  is selected to be larger than the value of  $b$ . In fact, when a packet is in the blocked state, the length of the destination queue in the next cycle will be either  $K$  (full buffer) or  $K-1$  (only one space available). If we disregard how many times it has been blocked previously, there will be only two cases : either the blocked packet faces a full queue or a queue with one space left. In the first case, with probability  $\frac{P[K]}{P[K]+P[K-1]}$ , the packet will be blocked again. In the second case, with probability  $\frac{P[K-1]}{P[K]+P[K-1]}$ , the packet will face possible contention from the other queue in the same stage which feeds this destination

queue. Incorporating these two probabilities in equation (2.7),  $c$  can be found in a similar way :

$$c = r \cdot C_{i+1,j1} + (1 - r) \cdot C_{i+1,j2}$$

where  $C_{i+1,j1}$  is the conditional probability that the  $j1$ -th queue in stage  $i+1$  is blocking a packet in stage  $i$  given that it blocked a packet in a previous cycle :

$$C_{i+1,j1} = \frac{P_{i+1,j1}(K)}{P_{i+1,j1}(K) + P_{i+1,j1}(K-1)} + \frac{r}{2} \cdot (1 - P_{i,j}(0)) \cdot \frac{P_{i+1,j1}(K-1)}{P_{i+1,j1}(K) + P_{i+1,j1}(K-1)}$$

$P_{blocked}$  is the probability that the server is in the blocked state. During this period, the server is inactive. Therefore, we may use this probability to approximate the blocking switch with "memory" characteristic. At the beginning of each cycle, the server tosses a coin which comes up heads with probability  $P_{blocked}$ , in which case the server will be blocked (inactive). If there is a packet at the server, it stays idle until the next cycle when the coin will be tossed again. With probability  $1 - P_{blocked}$ , the server will be active. The queue length then determines whether the server will send a packet or not. If there are packets in the queue, the server takes the first packet and routes it according to the routing probability.

Incorporating the probability  $P_{blocked}$  into our basic iteration model, the approach is then similar to the one that was described in section 2.1.4 except that the equivalent input rates and blocking probability are different from equations (2.6) and (2.7). Note that the effective input rate to a queue will be changed when  $P_{blocked}$  is included in the model. In the original model, when a queue is not empty (probability  $1 - P_{i,j}(0)$ ), it tries to transmit a packet to the destination in stage  $i+1$ . When  $P_{blocked}$  is used to approximate the persistent blocking, the probability that a queue will try to transmit a packet to the next stage is now

$(1 - P_{blocked}) \cdot (1 - P_{i,j}(0))$ . When the server is "active" (probability  $1 - P_{blocked}$ ) and is not empty (probability  $1 - P_{i,j}(0)$ ), it will transmit a packet to the next stage.

Let us define  $P_{i,j}^{eff}(0)$  to be the effective probability that  $Q_{i,j}$  will not send a packet (either the server is empty or the server is blocked). Let  $P_{i,j,blocked}$  to be the probability that  $Q_{i,j}$  is in the blocked state. Then the effective input rates for a queue,  $Q_{i,j}$  in this persistent blocking model are similar to the ones in section 2.1.4.1 :

$$P_{i-1,j1}^{eff}(0) = 1 - (1 - P_{i-1,j1,blocked}) \cdot (1 - P_{i-1,j1}(0))$$

$$P_{i-1,j2}^{eff}(0) = 1 - (1 - P_{i-1,j2,blocked}) \cdot (1 - P_{i-1,j2}(0))$$

$$X[1] = r[P_{i-1,j1}^{eff}(0)(1 - P_{i-1,j2}^{eff}(0)) + P_{i-1,j2}^{eff}(0)(1 - P_{i-1,j1}^{eff}(0))]$$

$$+ 2r(1 - r)(1 - P_{i-1,j1}^{eff}(0))(1 - P_{i-1,j2}^{eff}(0))$$

$$X[2] = [r(1 - P_{i-1,j1}^{eff}(0))] \cdot [r(1 - P_{i-1,j2}^{eff}(0))]$$

$$X[0] = 1 - X[1] - X[2] \quad (4.1)$$

And the equivalent blocking probability  $B_{i,j}$  can be found as follows :

$$B_{i,j} = r \cdot C_{i+1,j1} + (1 - r) \cdot C_{i+1,j2}$$

$$\text{where } C_{i+1,j1} = P_{i+1,j1}(K) + \frac{r}{2} \cdot (1 - P_{i,j}^{eff}(0)) \cdot P_{i+1,j1}(K - 1) \quad (4.2)$$

Using these equivalent input rates and blocking probability in the model in section 2.1.4, we can evaluate a persistent blocking MIN with a special hot spot traffic pattern. Incorporating the transformation method and the superposition method (i.e. replace  $r$  with  $r_{i,j}$ 's), we can evaluate any general traffic pattern with a persistent blocking behavior.



## 4.2 Model Results

We ran our model incorporating this new technique to handle the memory behavior for the same 6-stage Omega network with buffer size 4 under both the uniform traffic and the EFOS traffic pattern. The result is shown in comparison with the former model in Figure 4.2. The improved model greatly reduces the discrepancy between the simulation and analytical model results.

For a detailed study, we compare the improved analytical results with simulation in Figures 4.3-4.10. The confidence range of these simulations is 95%. The first case shown in Figures 4.3 and 4.4, is for a 4-buffered, 6 stage Omega network with a uniform traffic pattern. In Figure 4.3, the throughput is compared with various offered loads and in Figure 4.4, the average time delay is compared. The offered load is varied from 0.1 pkt/cycle to 1.0 pkt/cycle for each processing element. For offered loads within the range of small loss probability ( $q \leq 0.7$  pkt/cycle), the simulations verify the accuracy of the analytical results. Beyond this load (when packets begin to be discarded), the analytical results are slightly optimistic.

Figures 4.5 and 4.6 show the case of a 4-buffered, 6 stage Omega network with an EFOS traffic pattern. Throughput and average time delay are plotted against offered load, respectively. The non-uniformity of this pattern severely degrades the performance. The throughput graph shows very good correspondence between analytical results and simulations. For offered load within the low-loss range ( $q \leq 0.4$  pkt/cycle), delay performance of the analytical result is very accurate. However, delay performance of analytical results are still slightly optimistic in heavy load cases.

The third case is included to determine whether the analytical model per-

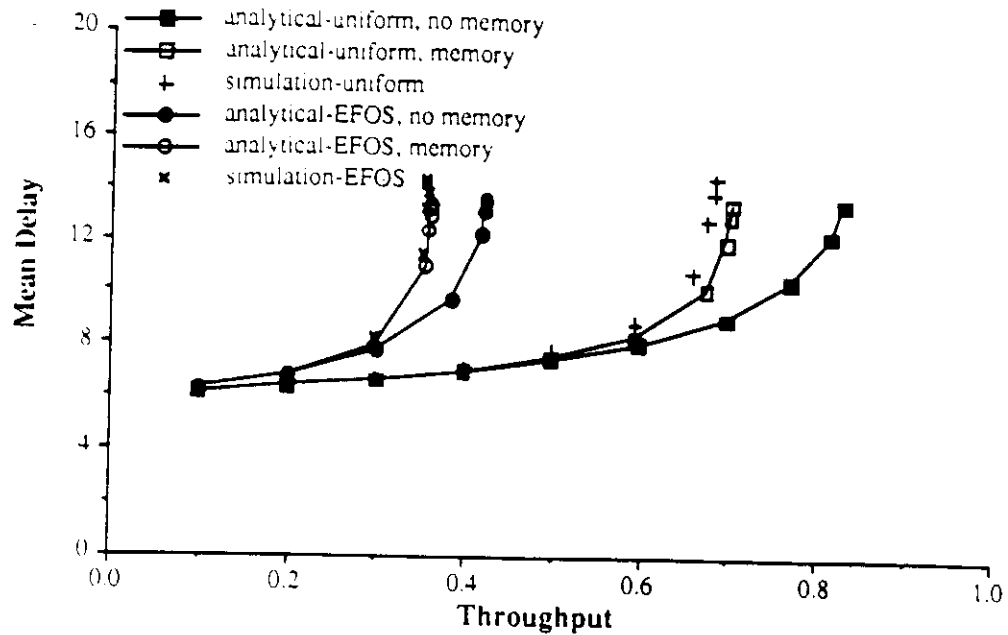


Figure 4.2: Comparison of results for a 6-stage, 4-buffered Banyan network with and without the "memory" behavior improvement

forms well with a larger buffer. The result for an 8-buffered, 6 stage Omega network with a uniform traffic pattern is shown in Figures 4.7 and 4.8. Except in heavy load ( $\rho=0.9$  and  $1.0$ ), analytical results measure well when compared to simulations. The throughput and delay performance are optimistic when the total input enters the range of heavy load. This simulation indicates that the analytical model performs well for networks with other buffer sizes.

We show the analytical results of a large sized network in Figures 4.9 and 4.10, namely a 4-buffered, 10 stage ( $1024 \times 1024$ ) Omega network with a uniform traffic pattern. Again, the analytical model is slightly optimistic when offered load exceeds the maximal attainable output. This indicates that the model is suitable for large sized networks as well.

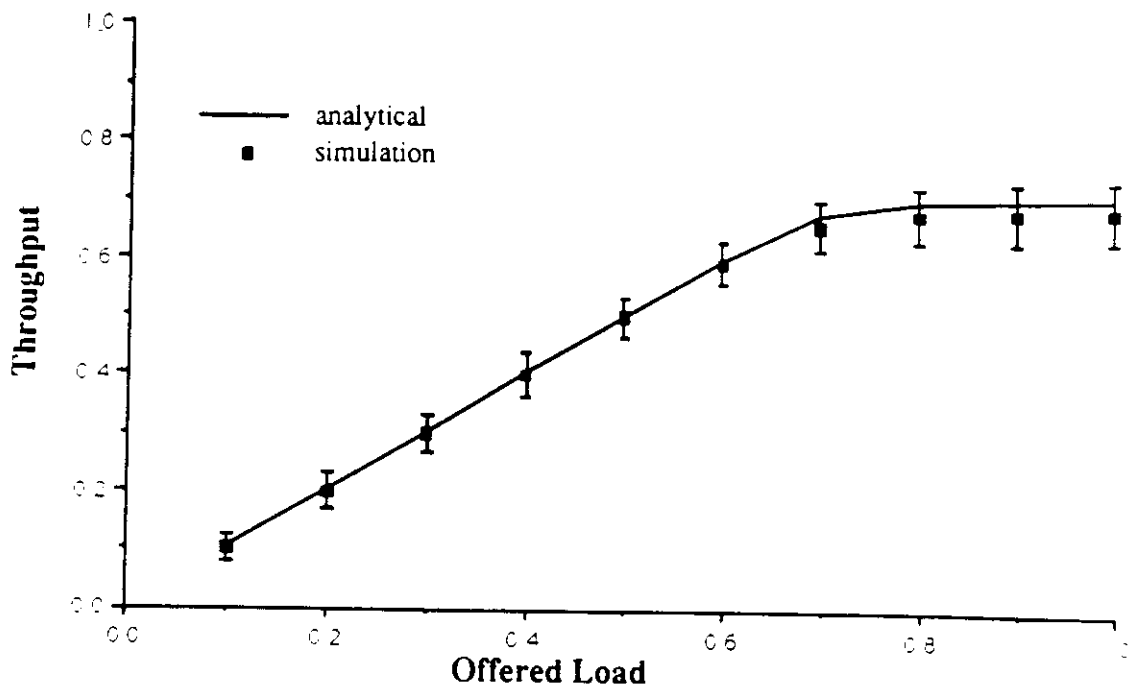


Figure 4.3: Throughput comparison for a 4-buffered, 6 stage Omega MIN with a uniform traffic pattern

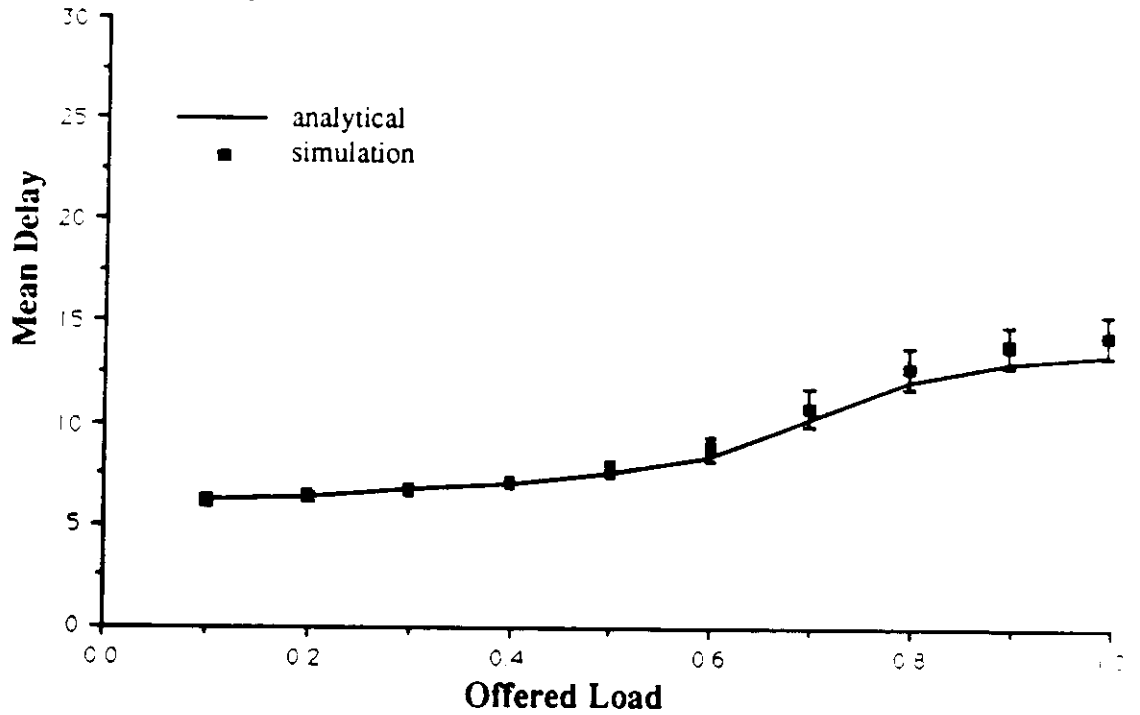


Figure 4.4: Mean delay comparison for a 4-buffered, 6 stage Omega MIN with a uniform traffic pattern

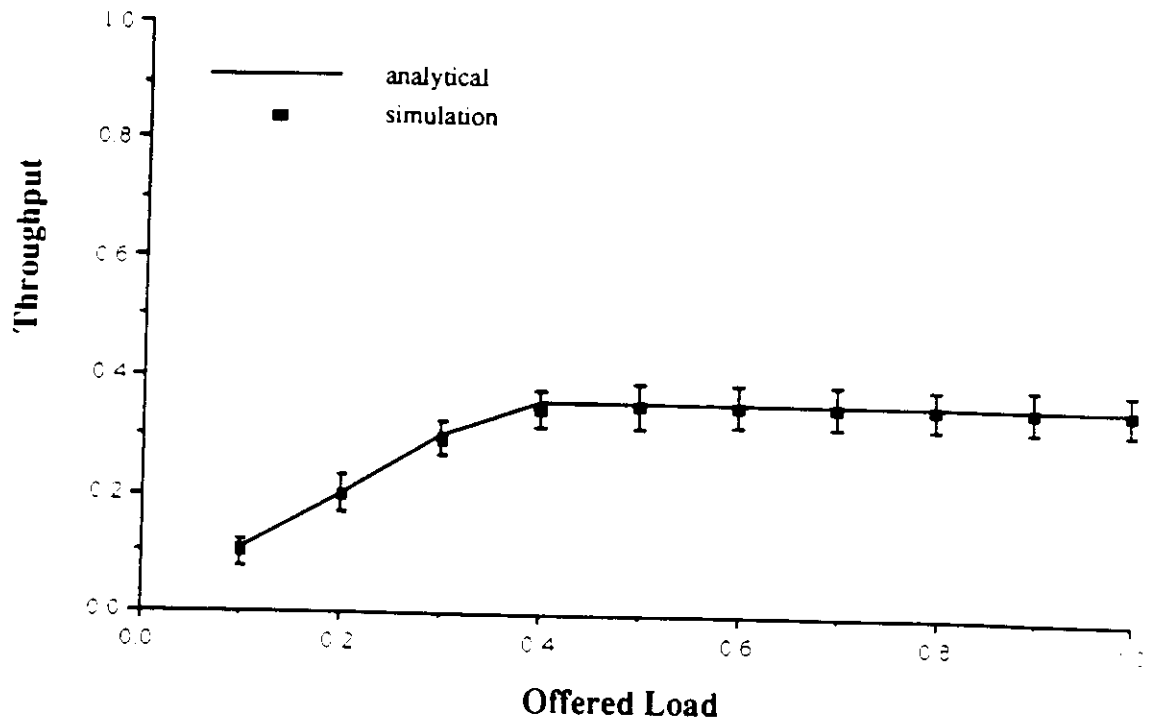


Figure 4.5: Throughput comparison for a 4-buffered, 6 stage Omega MIN with EFOS traffic pattern

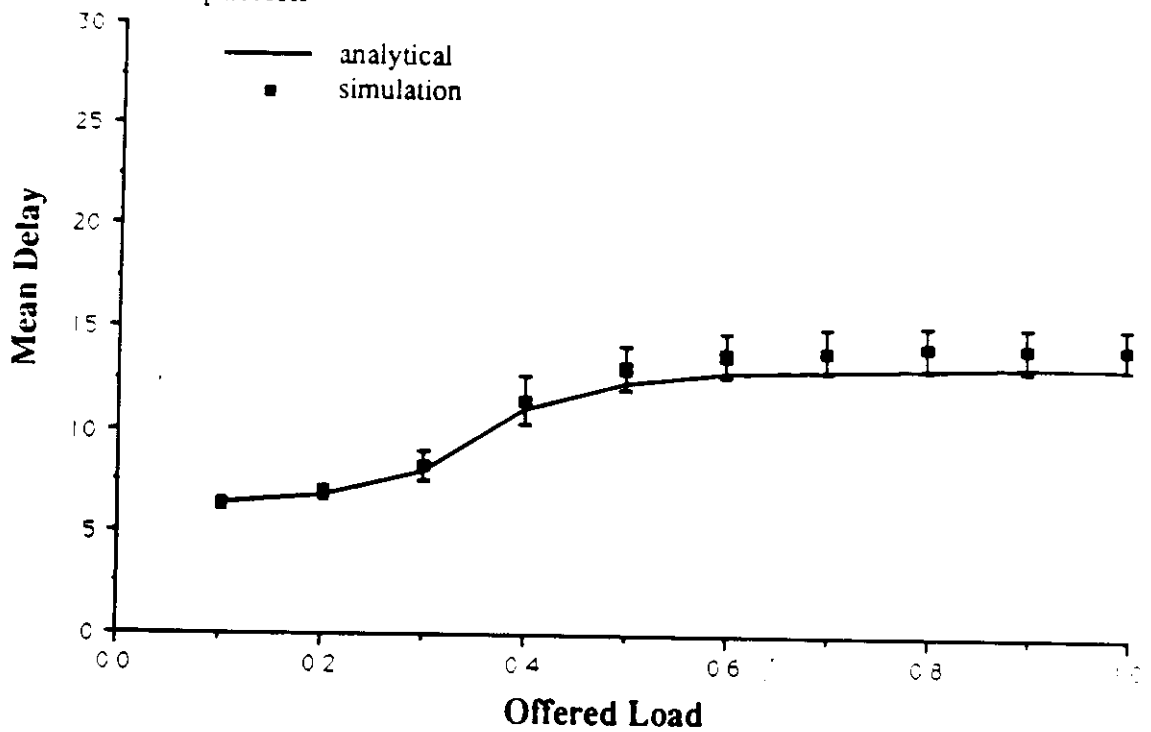


Figure 4.6: Mean delay comparison for a 4-buffered, 6 stage Omega MIN with EFOS traffic pattern

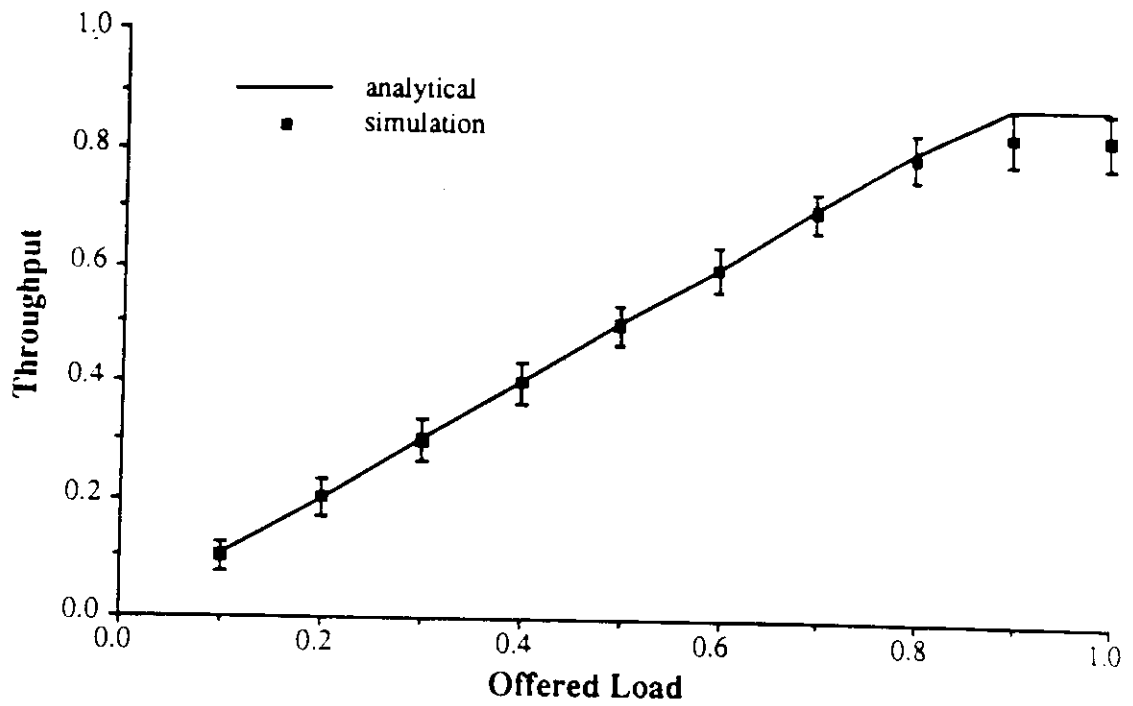


Figure 4.7: Throughput comparison for an 8-buffered, 6 stage Omega MIN with a uniform traffic pattern

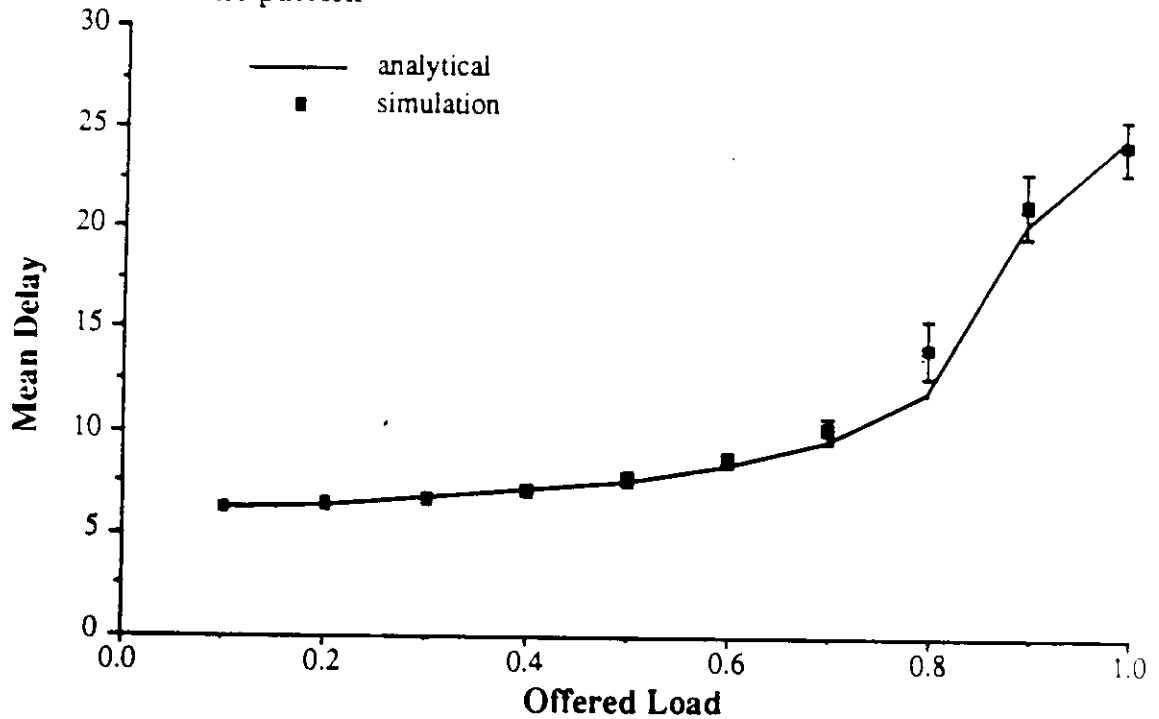


Figure 4.8: Mean delay comparison for an 8-buffered, 6 stage Omega MIN with a uniform traffic pattern

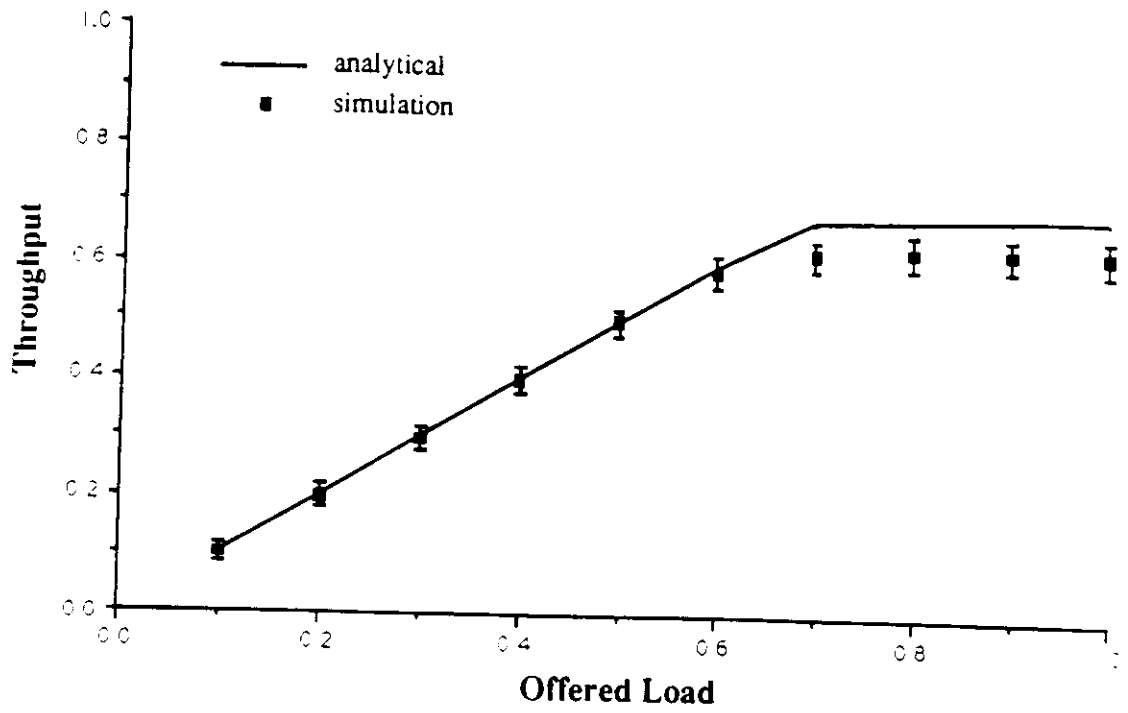


Figure 4.9: Throughput comparison for a 4-buffered, 10 stage Omega MIN with uniform traffic pattern

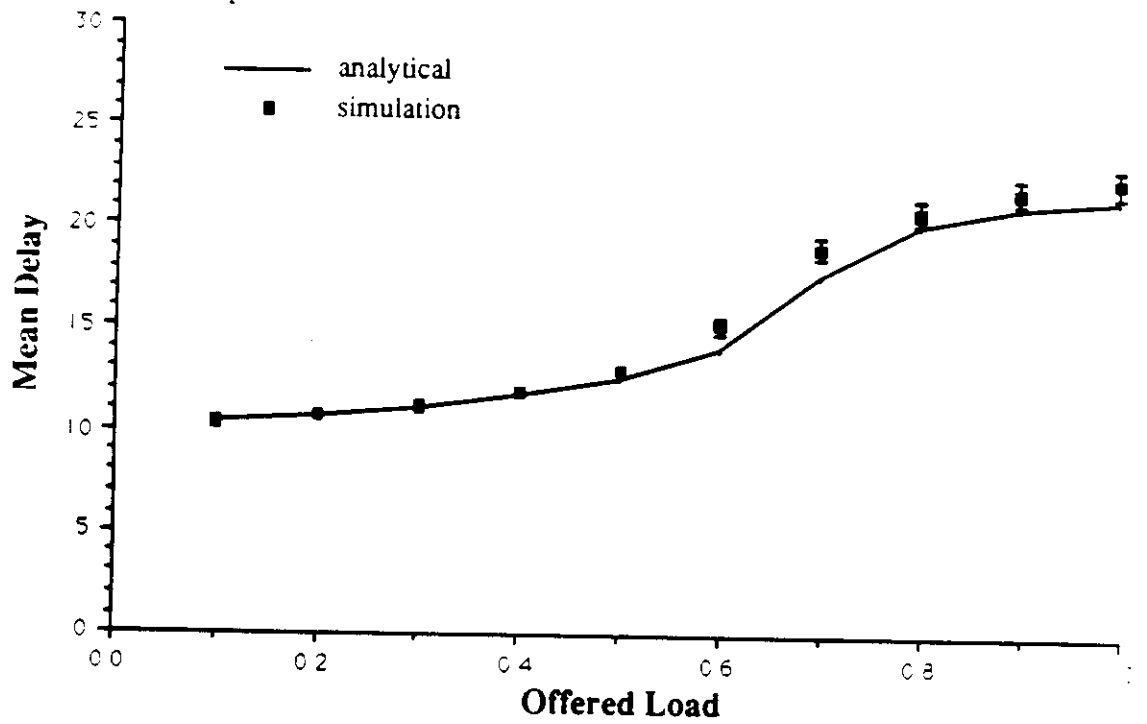


Figure 4.10: Mean delay comparison for a 4-buffered, 10 stage Omega MIN with uniform traffic pattern

### 4.3 Model Limitation

Even though the approximation methods that we proposed for a General Traffic Pattern model provides a very good analysis of systems performance with the uniform and EFOS general traffic patterns, the methods do not perform well when the traffic pattern is a severe hot spot traffic pattern. This indicates that the model works well only for a traffic pattern with a moderate persistent blocking. It still does not capture the true effect of persistent blocking. We leave this as a future research topic.

## CHAPTER 5

### Re-submission Model

In the previous model, we made the assumption that discarded packets are not re-submitted. Although this assumption did simplify our attempt to evaluate a complicated system, but it is not always realistic. In the ISDN application, for example, the voice packets, if discarded due to contention, are not re-submitted. But in many multiprocessor applications, a discarded packet will re-enter the MIN trying to reach its destination. In the real world, packets discarded before they enter the network will be queued. This queueing period introduces an extra delay over that experienced in the analytical model without re-submission. In addition to the extra delay, a model without re-submission avoids the stability problem. Therefore it is important to understand the delay performance and the stability problem. An analytical model which re-submits the discarded packets is needed. The modelling approach is described in section 5.1. The analytical model is verified in section 5.2. Several extensions based on the analytical model are discussed in section 5.3. A model which adjusts its input rate according to the system status is discussed in section 5.3.1. We show that proper adjustment of the input rate can reduce delay while maintaining the throughput at the same level. An extended model which determines the operating point under a given loss probability is discussed in section 5.3.2.



## 5.1 Modelling Approach

A finite number of buffers are placed at each processing element for the purpose of re-submission. Following Kurisaki & Lang's terminology [Lang 38], let us define this queue as a **c-link** (communication link). When a new packet is generated, it is placed at the end of the c-link. The c-link server tries to send the head-of-line packet to the first stage. If the queue in the first stage is full, then the packet is blocked. It will try to send it to the same queue in the next cycle. If the queue in the first stage has one buffer space available, and there is another processing element trying for the same queue, a contention occurs. If the packet loses the contention, it is blocked. The processing element generates a new packet in a cycle with probability  $q$ . If the processing element generates a new packet and takes up the last buffer space of the c-link, then the c-link has reached its full capacity. The c-link will send a signal to the processing element to notify it that the buffer is full. On receiving the full-buffer signal, the processing element stops generating new packets, and waits for further signals from the c-link to resume the new packet generation. When the c-link finally outputs a packet, a buffer space becomes available. The c-link sends a resume signal to the processing element. The processing element then resumes the new packet generation in the next cycle with probability  $q$ .

We extend the General Traffic Pattern Model that we proposed in Chapter 4 such that a c-link queue is accounted for. Instead of  $n$  queues, there are  $n+1$  queues in the tandem structure. The processing element generates a packet with probability  $q$ . Notice that there is no need to model the stopping-resuming process of the processing element. When the c-link is full, it will not take any more packets. Those packets generated by the processing element during the

time when the c-link is full are simply ignored and thrown away. When the c-link becomes available again, it will accept new packets that are generated by the processing element at a rate  $q$ . Therefore, the situation is the same whether we allow the processing element to stop or not. The modelling approach is similar to the General Traffic Pattern Model. The process of evaluating the Markov chain of the c-link is incorporated in each iteration. If we let the c-link size to be the same as the finite buffer size in each switching element, then the Markov chain of the c-link is identical to the one that we show in Figure 3.2.

The model begins with the evaluation of the c-link's Markov chain. The offered load  $q$  is taken as input process. The blocking probability is taken from the first stage (in the first iteration, this blocking probability is 0 since the queues in the network are yet to be evaluated). The Markov chain of the c-link is evaluated. The steady state probability of the c-link is taken as the input process to the queue in the first stage. The Markov chain of the queue in the first stage is evaluated. We evaluate the Markov chain of all queues in each stage. Then we iterate the evaluation process until the throughput converges.

## 5.2 Model Results

We let the c-link size be equal to the finite buffer size 4 in the network. A 5-stage Omega network with EFOS traffic pattern is modelled. The result is shown in Figure 5.1 and Figure 5.2 in comparison with the simulation. Figure 5.1 is the mean delay comparison to a simulation with a 95 % confidence range. The analytical model is slightly optimistic with a heavy system load. The throughput comparison indicates that the model is very good at predicting the performance.

We ran the model for a 6-stage Omega under a uniform traffic pattern, the

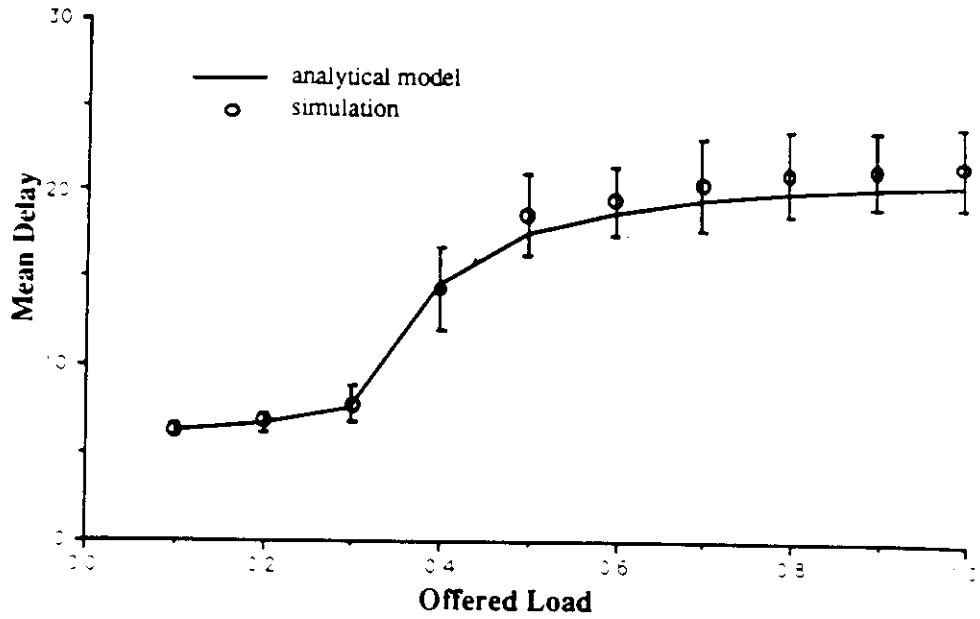


Figure 5.1: Analytical model vs. simulation for a 5 stage Omega under EFOS traffic pattern

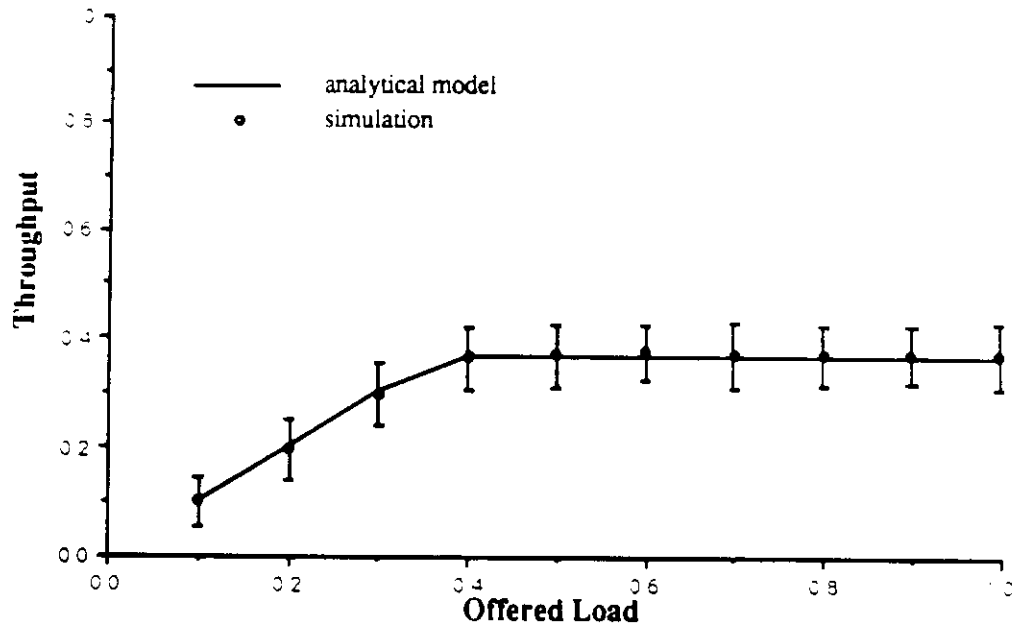


Figure 5.2: Analytical model vs. simulation for a 5 stage Omega under EFOS traffic pattern

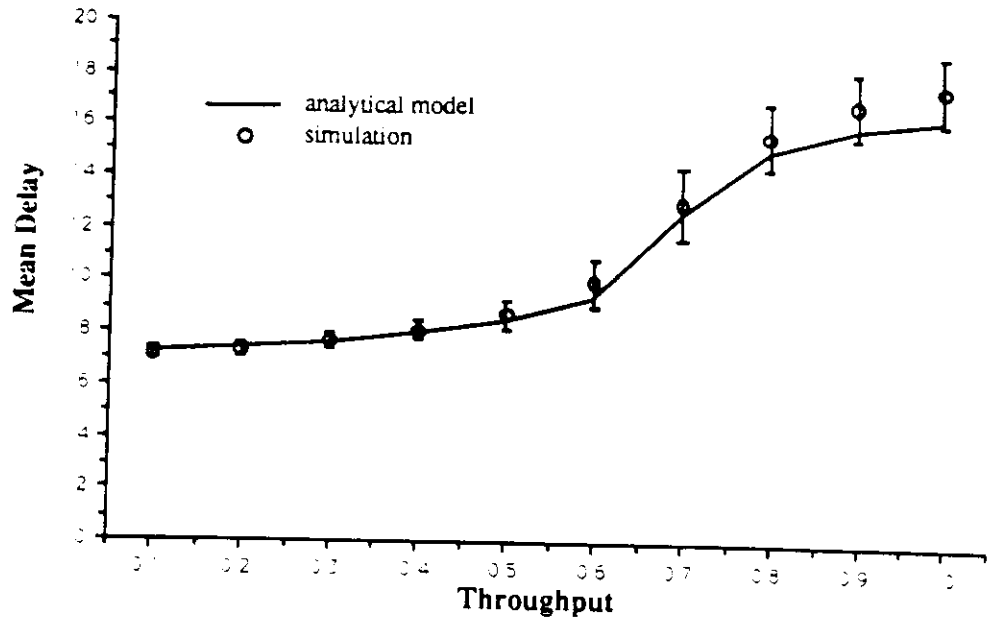


Figure 5.3: Analytical model vs. simulation for a 6 stage Omega under uniform traffic pattern

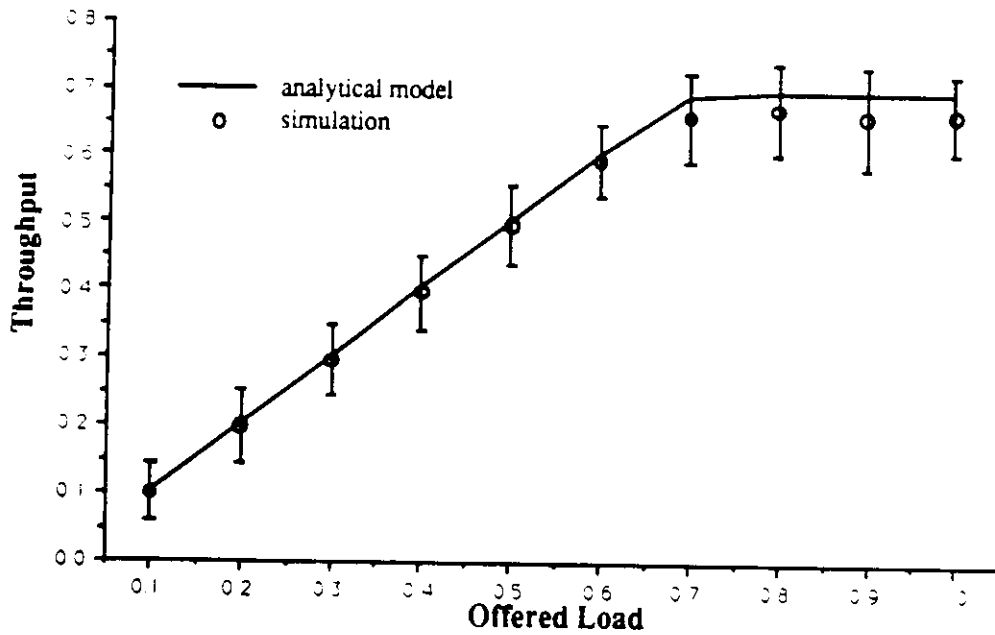


Figure 5.4: Analytical model vs. simulation for a 6 stage Omega under uniform traffic pattern

results are compared to the simulation in Figures 5.3 and 5.4. Both the throughput and the mean delay are slightly optimistic with a heavy system load. The simulation indicates that the analytical model is very accurate in the stable range of operations.

Both the mean delay curves in Figures 5.1 and 5.3 rise rapidly and level off gradually. The quick rise is the point when the offered load is approaching the maximal capacity. Therefore the queues in the network and the c-link are quickly filled up. When the new packets begin to be discarded due to lack of room in the c-link, the delay curve starts to level off. If the c-link is an infinite queue, the curve explodes to infinity. We will show the c-link length and the c-link delay in a heavy traffic case in the next section.

### **5.3 Extended Model**

We would like to focus on several questions regarding the operating points, the loss probability and the stability problem.

#### **5.3.1 Rate Adjusted Model**

In general, when the mean delay rises rapidly, the c-link is most likely saturated. At this point, it is meaningless to have a large offered load. Even though the processing element is signaled to stop generating new packets when the c-link is full, as soon as the c-link is able to accept packets, the processing element resumes the generation of the packets again at a rate greater than the system capacity. The c-link quickly saturates again, and thus creating a long delay.

In reality, when the offered load exceeds the system capacity, a rate adjustment mechanism must be employed to reduce the offered load. In this section,

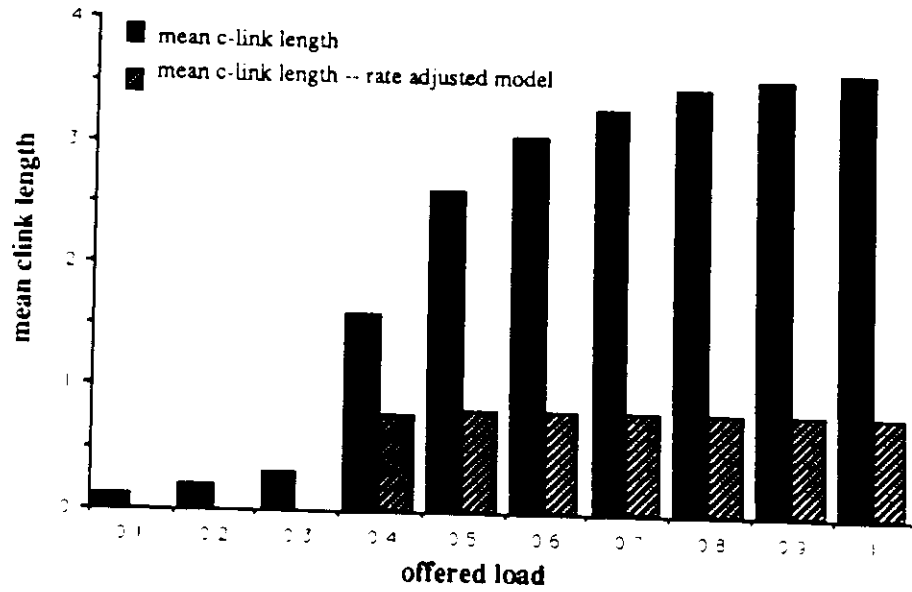


Figure 5.5: Mean c-link length comparison for rate adjusted model

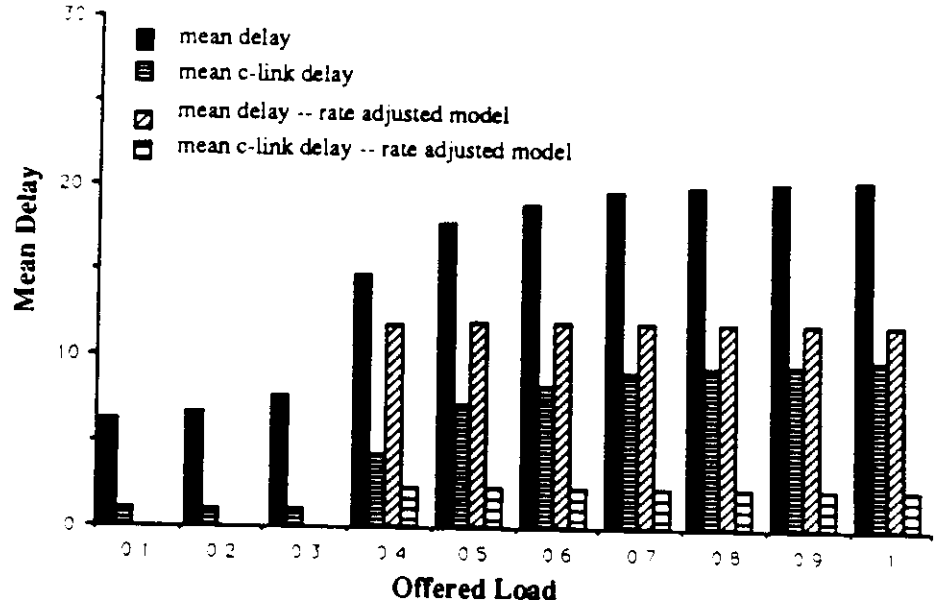


Figure 5.6: Mean delay comparison for rate adjusted model

we would change the offered load when it exceeds the system capacity. The following is an iteration mechanism that reduces the offered load repeatedly until the loss probability is below a certain value.

```

PA :=1
REPEAT
    q:=q · PA
    procedure General Traffic Pattern:
    begin
        input stage number, buffer size:
        for all PEs do
            call procedure pattern-bitr; /*assign traffic patterns*/
            call procedure traffic-trace; /*transformation method*/
        endfor;
        for all ij do /*superposition method*/
            r[ij]:=r[ij] div c[ij];
        endfor;
        call procedure buffer-min /*solve for PA, time delay*/
    end;
UNTIL 1 - PA < 0.01

```

The assignment,  $q:=q \cdot PA$ , adjusts the offered load gradually. For example, if the model begins with an offered load 0.9 in an EFOS traffic pattern, then the model first solves for the PA (Probability of Acceptance) as 0.4. Obviously, the loss probability (1-PA) is too high. Therefore, we adjust the offered load to be  $0.4 \times 0.9 = 0.36$ , which is the throughput value. We iterate the General Traffic

Pattern Model again, and keep adjusting the value of the offered load until the loss probability is below a certain value.

A 5-stage Omega network with a c-link size of 4 and a finite buffer size of 4, with an EFOS traffic pattern is analyzed and the result is shown in Figure 5.5. The model terminates when the loss probability is below 0.01. The mean c-link lengths of the initial offered load and the adjusted load are compared. For the light-load cases (0.1 - 0.3), the loss probability is less than 0.01 when the first iteration of the General Traffic Pattern Model terminates, hence the rate is not adjusted. When the system load exceeds the system capacity ( $q \geq 0.4$ ), the offered load is adjusted. Since the EFOS traffic pattern is a severely congested traffic pattern, the model quickly converges. All cases ( $1 \geq q \geq 0.4$ ) terminate in the second iteration when the loss probability is less than 0.01. The mean c-link length is shown in Figure 5.6, which compares the values of the first iteration (original values) and the second iteration (terminal values). If the rate is not adjusted, the mean c-link length is almost full in every case. The length increases sharply for  $q=0.4$  and  $0.5$ . Then, it slowly levels off. The mean delay is thus high. If we adjust the offered load, the mean c-link length becomes very small. For an offered load greater than 0.3, the mean c-link length is less than 1. Therefore by adjusting the offered load, we are able to reduce the mean delay significantly while maintaining a high throughput.

The mean delay and the c-link delay of the original model and the rate adjustment model are compared in Figure 5.6. Note that the difference between the mean delay and the c-link delay is the delay experienced in the network. The delay experienced in the network for the original model and for the rate adjusted model are nearly the same. This indicates that the delay we manage to reduce is the c-link delay. The mean delay is cut from 20 to 12 for an offered load of 1.0.



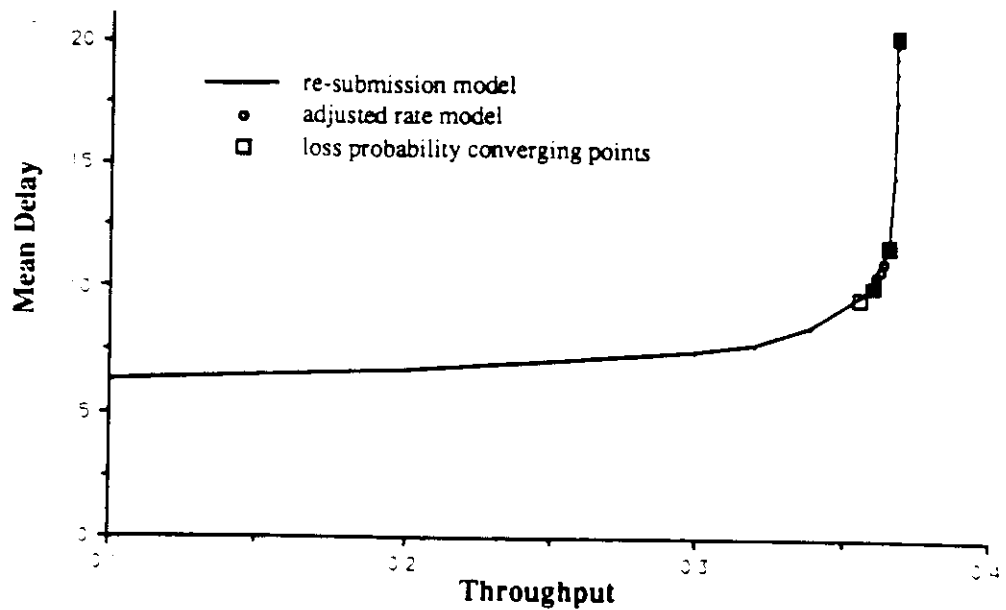


Figure 5.7: Throughput-delay curve for rate adjusted model, 5 stage, EFOS pattern

### 5.3.2 Maximal Input Rate for a Given Loss Probability

There is a maximal system capacity for a given traffic pattern, a c-link size and a finite buffer size. Depending on the requirement of loss probability and the mean delay, an operating point can be determined such that a desirable system performance is achieved. We ran the rate adjusted model with the convergence condition  $1 - PA < 0.0001$ . We begin with an offered load of 0.9; the Loss probability, the throughput and the mean delay are recorded in every iteration. We choose three sets of data which represent the cases where the loss probability is less than 0.01, 0.001 and 0.0001, respectively. They are listed as follows.

loss probability	0.589	< 0.01	< 0.001	< 0.0001
offered load	0.9	0.37	0.361	0.357
throughput	0.37	0.367	0.361	0.357
delay	20.367	11.862	10.277	9.802
iteration numbers	1	2	10	30

On the second iteration, the loss probability converges within 0.01. However, it takes 10 iterations to converge to 0.001 with the throughput 0.361 and the mean delay 10.277. It takes 30 iterations to converge within 0.0001. These points are plotted in Figure 5.7. We show the original model as a solid line. The results of the first 10 iterations are plotted in small circles. Those four points where loss probability converges within 0.01, 0.001 and 0.0001 are plotted in small boxes. If an application requires a minimum loss probability, then the maximal input rate can be determined by using the rate adjusted model. If the performance objective is to find the operating point where the Power [Klei 79],  $\frac{\text{throughput}}{\text{delay}}$ , is maximized, the convergence condition can be changed such that it converges when the Power,  $\frac{\text{throughput}}{\text{delay}}$ , cannot be improved.

## CHAPTER 6

### Delay Model Analysis

The **blocking switch** models were discussed in Chapter 2 to Chapter 5. Both the throughput and the overall mean delay were found. In this Chapter, we concentrate on the analysis of the **turn back switch**. A turn back switch differs from a blocking switch in that when a contention occurs where there is not enough buffer space for contending packets, instead of being blocked, packets are *turned back* to be re-submitted at the processing elements. This removes the persistent blocking behavior of the blocking switch, thus we predict that the system performance of a turn back switch should be better than a blocking switch in heavy loads. When a rejected packet is not turned back to the original source (processing element), instead, it is *rotated* to join the end of the local queue, we call this new configuration a **rotating switch**. The motivation for the rotating switch (to be discussed in section 6.4) comes from the performance comparison of the blocking switch and the turn back switch .

In section 6.1, we first propose an analytical model for the turn back switch with a uniform traffic pattern. The analytical model is based on a recurrence equation. The mean delay of a particular path is determined by using the recurrence equation. Different Markov chains, resulting from different assumptions of the queue, are discussed. Two different approaches are proposed : a Combined Rate Model with an explicit analytic solution and a Feedback Model with an iterative approach. Both models are analyzed and compared. The advantages of both models are discussed. In section 6.2, we extend the Feedback Model to

analyze a turn back switch with a non-uniform traffic pattern. The exact feedback traffic is calculated. Since the feedback of a non-uniform traffic pattern may change the original traffic pattern, we update the traffic pattern in each iteration with the exact feedback traffic. Two example traffic patterns : a hot spot and a EFOS traffic pattern are analyzed. We compare the system performance of the turn back switch to the blocking switch in section 6.3. The result shows that each switch is better than the other switch in a certain range of system load. This fact leads us to the new configuration, the rotating switch. The rotating switch model is proposed and the result is compared to the turn back switch and the blocking switch in section 6.4.

### **6.1 The Turn Back Switch Model with a Uniform Traffic Pattern**

A turn back switch differs from a blocking switch in that when a contention occurs where there is not enough buffer space for conflicting packets, instead of being blocked, packets are rejected and are re-submitted at the processing elements (see belows). This assumption allows the queue at each switching element to output one packet per cycle when it is not empty, regardless of the status of queues in the next stage. Therefore a queue is independent of conditions at the next stage. This independence simplifies the modelling of a turn back switch since there is no interdependence among queues in different stages. The only information we need to solve for each queue is the input rates from the previous stage.

For multistage interconnection networks using the turn back switch as the basic building block, if the original traffic pattern is not uniform, re-submitting the rejected packets into the network changes the traffic pattern. The re-submitted packets are more likely to go to the favorite memory modules. The modelling

approach must then keep track of each rejected traffic flow from different queues at different stages to assure that they are re-submitted at the correct processing elements. We shall discuss the model approach (with a non-uniform traffic pattern) in section 6.2.

In this section, we propose an analytical model for a multistage interconnection network using turn back switches with a uniform traffic pattern. The model takes advantages of a recurrence equation of the mean system time. This approach is extended to model the general traffic patterns in a later section.

We first describe a turn back switch. The Markov chain of the queue in a single switching element is found in terms of the input parameters. We then develop an analytical solution based on a recurrence equation to solve for the mean system time and the throughput for uniform traffic. An iterative solution for the non uniform traffic is then given. We also discuss two different Markov models regarding the assumptions of the queue. The analytical results are verified through simulation for different network sizes.

### 6.1.1 Turn Back Model

In this section, we discuss a class of  $n$  stage interconnection networks (e.g. Banyan, Omega) with  $K$  finite buffers at each of its switching element's output ports (see section 2.1.2). The difference from the previous chapters is that these switching elements are  $2 \times 2$  turn back switches. Each processing element has an infinite queue in order to accept rejected packets such that any possible loss of rejected traffic is prevented.

Each processing element generates a new packet according to a Bernoulli arrival process. Each packet is assigned a destination address according to a destination distribution (traffic pattern). This new packets is then submitted

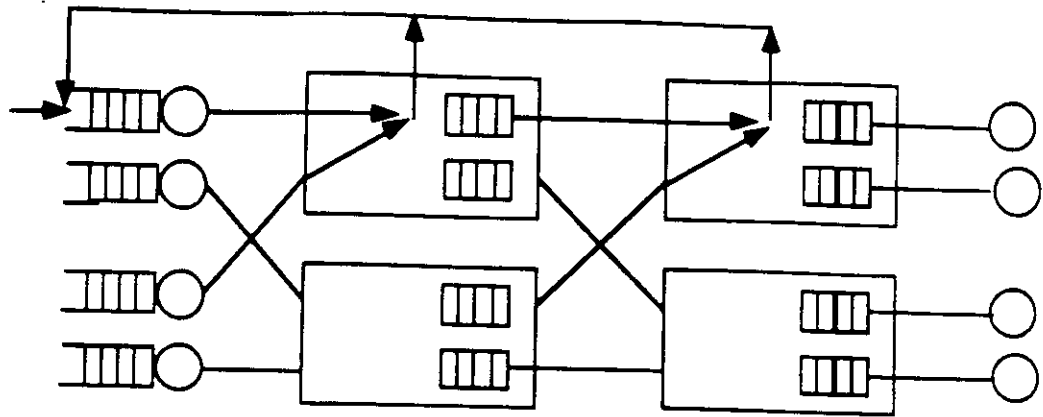


Figure 6.1: 4x4 interconnection network with turn back switch as building block to the infinite queue at the processing element. The packet in the server of the infinite queue is submitted into the network at the end of a cycle. The maximum number of packets that can arrive at a finite-buffered queue in stage 1 is two because there are only two input links (see Figure 6.1). A packet entering the first stage selects an output port according to its address tag, and joins the end of the destination queue. However, if there are two packets heading for the same queue, a possible contention occurs. If there is enough space, the queue accepts both packets. If there is only one space left, then one packet is randomly chosen, and the other is rejected. The rejected packet returns to the originating source (processing element) immediately and joins the tail of this infinite queue. Since a turn back switch always outputs a packet as long as it is not empty, it can always accept at least one packet per cycle. Hence the situation when both packets find no space and are both rejected is not possible in a turn back switch. The infinite queue is assumed to be able to accept as many rejected packets as necessary in one cycle, in addition to accepting newly generated packets.

## 6.1.2 Modeling Approach

We discuss the analytical model by focussing on the following 3 different components of the system : the finite buffered queue inside the network, the infinite queue at the PE's and the path that packets take to reach their destination.

### 6.1.2.1 The Markov Model for Finite-Buffered Queue

Depending on the assumption of the operations on sending and receiving packets, there are two Markov chains for the finite-buffered queue inside the network. When the queue is in state 0, it either moves to state 1 if there is only one incoming packet at the end of the cycle, or moves to state 2 if there are two incoming packets. While in state 1, the queue moves back to state 0 when there is no input packet. The queue remains at state 1 when there is only one incoming packet. This is due to the fact that a queue outputs a packet as long as it is not empty. It only moves to state 2 when there are two incoming packets. All other states have the same transitions except state  $K-1$  and state  $K$ . At the end of a cycle in state  $K-1$ , two incoming packets find only one buffer space left. At the same time, the server is sending the first packet to the next stage. We develop two models with different assumptions regarding the way a queue handles packets. The different assumption is whether the queue can perform simultaneous operations to remove one packet at the head of the queue and to accept two incoming packets at the same time while in state  $K-1$ .

**Simultaneous Model** When the queue is in state  $K-1$ , if there are two packets coming in at the end of a cycle, the packet in the server is transmitted to the next stage, thus creating one more space in the buffer. Then both incoming packets are accepted and the queue reaches state  $K$ . When the queue is in state

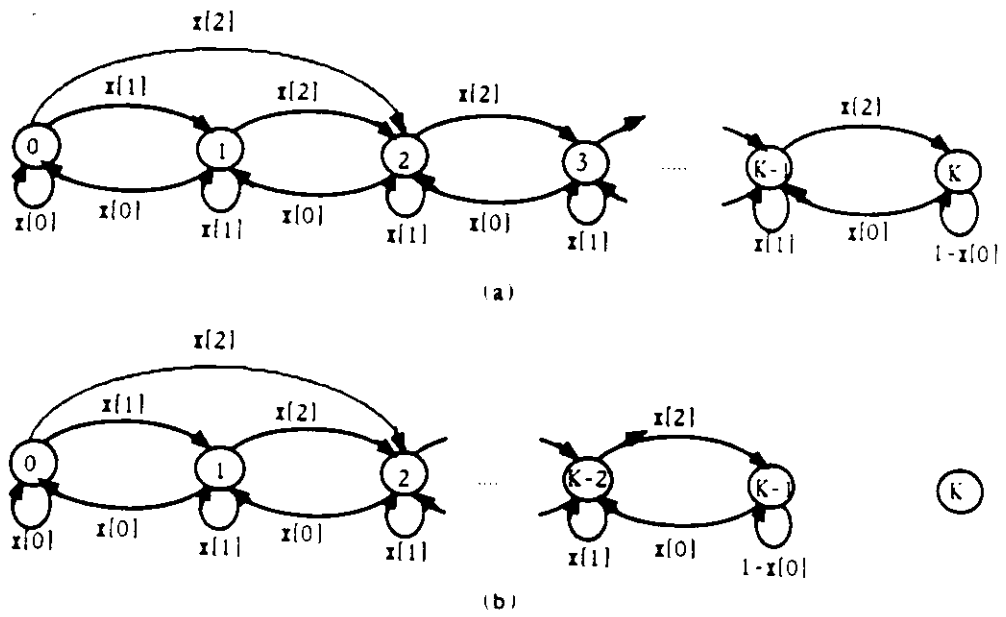


Figure 6.2: Markov chain for (a) simultaneous operation (b) no simultaneous operation.

$K$  and there are two incoming packets, simultaneous operation guarantees that one packet can be accepted while the other one must be rejected. While in state  $K$ , the queue moves back to state  $K-1$  if there is no input at the end of the cycle. It remains in state  $K$  when there is more than one incoming packet. We define  $x[i]$  to be the probability of  $i$  incoming packets. The Markov chain for this simultaneous model is shown in Figure 6.2 (a). The steady state probability  $P[i]$  can be found as follows :

$$\begin{aligned}
 P[0] &= \frac{x[0] - x[2]}{1 - \left(\frac{x[2]}{x[0]}\right)^K} \\
 P[1] &= \frac{1 - x[0]}{x[0]} \cdot P[0] \\
 P[i] &= \left(\frac{x[2]}{x[0]}\right)^i \cdot \frac{1}{x[2]} \cdot P[0] \quad 2 \leq i \leq K
 \end{aligned}$$



**Non-simultaneous Model** If the queue cannot perform the simultaneous operation as described, then when the queue is in state  $K-1$ , it can only accept one packet. Therefore if there are two incoming packets at the end of the cycle, one packet must be rejected. In this case, the state  $K$  can never be reached because the queue always outputs one packet in a cycle. While in state  $K-1$ , a queue either remains at state  $K-1$  (when there is at least one incoming packet) or moves back to state  $K-2$  (when there is no incoming packet) in the next cycle. The non-simultaneous assumption is more realistic, and is the same assumption we made in the blocking switch model we discussed in Chapters 3 through 5. The Markov chain is shown in Figure 6.2 (b). The steady state probabilities can be solved as follows :

$$\begin{aligned}
 P[0] &= \frac{x[0] - x[2]}{1 - \left(\frac{x[2]}{x[0]}\right)^{K-1}} \\
 P[1] &= \frac{1 - x[0]}{x[0]} \cdot P[0] \\
 P[i] &= \left(\frac{x[2]}{x[0]}\right)^i \cdot \frac{1}{x[2]} \cdot P[0] \quad 2 \leq i \leq K - 1 \\
 P[K] &= 0
 \end{aligned}$$

In order to compare the result of the turn back switch to that of the blocking switch, we choose the same assumption (i.e. no simultaneous operations) for the turn back switch model. Throughout this chapter, unless otherwise mentioned, the finite-buffered queue in the turn back switch will assume the basic operation that we described in the non-simultaneous model. We shall compare the performance of a MIN employing these two queuing disciplines in their switching elements.

### 6.1.2.2 The Input Model for the Infinite Queue at PE

The arrival process to the infinite queue at PE is assumed to be a bernoulli process with parameter  $q$ . The server is a deterministic server with service time one. This makes the infinite queue a discrete time, Geo/D/1 queue with feedback from the network. It is complicated to calculate the exact transition probability from one state to another, therefore the resulting Markov chain is difficult to solve for. Hence, we use a continuous time M/D/1 queue with feedback to approximate this Geo/D/1 queue with feedback. The Poisson arrival rate to this infinite queue is  $q$  pkt/cycle and the deterministic service time is 1. The stable condition for this infinite queue is that the combined rate of new packets,  $q$ , plus the feedback rate of rejected packets is less than 1.

We propose two modelling approaches to calculate the combined input rate : the Combined Rate Model and the Feedback Model.

**Combined Rate Model** For a uniform traffic pattern, the re-submitted packets still have the same traffic pattern as do the newly generated packets. Therefore, it is reasonable to combine them into a total input rate of  $q$ . This assumption simplifies the modelling approach such that there is no need to calculate the exact feedback rate. When a contention occurs, the rejected packet is thrown away since we assume that it will join the new packet generating process at a rate  $q$ . For the stable range of operation ( $0 \leq q < 1$ ), we have the mean delay,  $T_0$ , for passing through this infinite queue as follows :

$$\begin{aligned}\rho &= q \\ P_0[0] &= 1 - \rho \\ T_0 &= \frac{2 - \rho}{2 - 2 \cdot \rho}\end{aligned}$$

However, if the given pattern is not uniform, then the feedback packets may not see the same traffic pattern as do the new packets. For instance, in the case of a hot spot, most of the resubmitted packets are packets that are heading for the hot spot. Therefore, the traffic pattern of these resubmitted packets is completely different from the given pattern. The disadvantage of this combined rate approach is that it can not properly reflect the changes of the traffic pattern when the given pattern is not uniform. Hence for the non-uniform traffic pattern, we need a modelling approach that accounts for the feedback traffic.

**Feedback Model** The second approach calculates the exact feedback traffic. This is especially useful when the traffic pattern is not uniform. Let the rate of the feedback traffic be  $d$  and let the new packet generation rate be  $q$ . For the stable range of operation ( $0 \leq q + d < 1$ ), the mean delay,  $T_0$  can be found as follows :

$$\begin{aligned}\rho &= q + d \\ P_0[0] &= 1 - \rho \\ T_0 &= \frac{2 - \rho}{2 - 2 \cdot \rho}\end{aligned}$$

For uniform traffic patterns, the throughput-delay curve of the Feedback Model should be the same as the one of the Combined Rate Model.

In this section, both approaches will be analyzed and compared. It shows that the Combined Rate Model is simple. However, the Feedback Model is more

general and can be extended to analyze the case with a non-uniform traffic pattern.

### 6.1.2.3 Recurrent Path

Unlike the blocking switch where interdependence among queues in different stages makes the model solvable only through iterations, the turn back switch does not have a similar interdependence because it returns the rejected packets to the processing element. Therefore the steady state probability of a queue in the network can be solved directly. Observing the relationship among queues in different stages, we see that a queue depends only on its previous queue (for input), and is independent of the queues in the next stage since there is no blocking. Therefore, the probability that there is no incoming packets to the infinite queue at stage 0 (i.e. PE),  $x[0]$ , is :

$$x[0] = 1 - q$$

Let  $P_i[j]$  be the probability that a queue in stage  $i$  contains  $j$  packets. Since we approximate the discrete time Geo/D/1 queue as a continuous time M/D/1 queue, the probability that this queue is empty,  $P_0[0]$ , is thus  $1 - q$ . This probability is used to calculate the input process to a queue in the first stage :

$$\begin{aligned} x[2] &= (r \cdot (1 - P_0[0]))^2 \\ x[1] &= 2 \cdot r \cdot (1 - P_0[0]) \cdot (1 - r \cdot (1 - P_0[0])) \\ x[0] &= 1 - x[1] - x[2] \\ &= (1 - r \cdot (1 - P_0[0]))^2 \end{aligned}$$

Using these probabilities, we can solve for the steady-state probabilities,  $P_1[i]$  of the first stage queue (the Markov chain is shown in Figure 6.2(b)) as discussed in section 6.1.2.1.

Similarly, we use  $1 - P_1[0]$  as the input parameter of the queue in the second stage, and so on. In general, the input process to a queue in stage  $i$  is :

$$\begin{aligned}
 x[2] &= (r \cdot (1 - P_{i-1}[0]))^2 \\
 x[1] &= 2 \cdot r \cdot (1 - P_{i-1}[0]) \cdot (1 - r \cdot (1 - P_{i-1}[0])) \\
 x[0] &= 1 - x[1] - x[2] \\
 &= (1 - r \cdot (1 - P_{i-1}[0]))^2
 \end{aligned}$$

The steady state probabilities for queues in all stages can be solved accordingly.

For a uniform traffic pattern, all queues in the same stage are identical. Therefore, we concentrate the calculation on one path, as shown in Figure 6.3. Let  $T_i$  be the mean of the random variable representing the time needed to pass through a queue in stage  $i$ . Let  $d_i$  be the conditional rejecting probability for a packet given that it attempts to enter stage  $i+1$  and is rejected due to a contention. Let  $S_i$  be the mean of the random variable representing the accumulated system time after a packet enters stage  $i+1$ . A packet first spends  $T_0$  cycles in the infinite queue at a PE, then it is transmitted to the queue in the first stage. However, there might be a contention such that the packet is rejected due to no buffers. If the packet is rejected, it returns to join the end of the queue in stage 0, and spends another  $T_0$  before the queue tries to transmit it again. A packet may repeat this process many times until it finally meets no contention or wins a contention, and enters the first stage (the packet now accumulates a system time

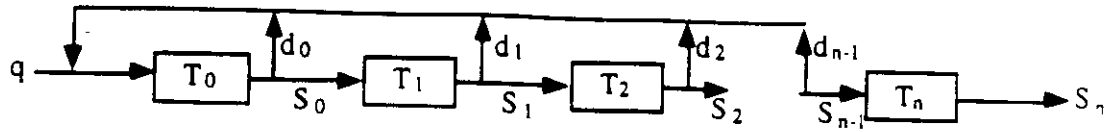


Figure 6.3: The recurrent relationship among variables in a particular path

$S_0$ ). A packet repeats this process stage by stage until it reaches stage  $n$ . Since there is no contention for the outputs in stage  $n$ , packets reach the destination after spending  $T_n$  cycles in stage  $n$ . This repeating process of rejecting and re-submitting can be represented in the following recurrent equations in terms of the accumulated system time  $S_i$ , the conditional rejecting probability  $d_i$ , and the queueing time  $T_i$  :

$$\begin{aligned}
 S_0 &= \frac{T_0}{1 - d_0} \\
 S_i &= \frac{S_{i-1} + T_i}{1 - d_i} \quad 1 \leq i < n \\
 S_n &= S_{n-1} + T_n
 \end{aligned}$$

The explicit expression for  $S_i$  in terms of  $T_i$  and  $d_i$  is as follows :

$$S_i = \sum_{m=1}^i \frac{T_m}{\prod_{j=0}^{i-m} (1 - d_{i+1-j})}$$

Therefore, the average system time for passing through the network,  $S_n$  is :

$$\begin{aligned}
 S_n &= S_{n-1} + T_n \\
 &= \sum_{m=1}^{n-1} \frac{T_m}{\prod_{j=0}^{n-m-1} (1 - d_{n-j})} + T_n
 \end{aligned}$$

Once the values of  $d_i$ 's and  $T_i$ 's are determined, the average system time for a turn back switch can be solved.  $T_i$  is the mean time a packet spends in stage  $i$ .

Since we have the state probabilities for each queue, we use the Little's result to solve for  $T_i$  :

$$T_i = \sum_{k=1}^K \frac{k \cdot P_i[k]}{1 - P_i[0]}$$

The rejecting probability  $d_i$  can be found as follows :

$$d_i = \frac{1}{2} \cdot P_{i+1}[K-1] \cdot r \cdot (1 - P_i[0])$$

which is the case when there is a contention and there is not enough space to take both, the packet is rejected with probability 0.5.

### 6.1.3 Performance Measure

The performance measures that we are interested in are the throughput and the mean delay. The mean delay of a turn back switch,  $S_n$  is shown in the previous section. The throughput can be calculated as follows :

$$throughput = 1 - P_n[0]$$

### 6.1.4 Solution Algorithm

For different modelling approaches (i.e. the Combined Rate Model and the Feedback Model) the solution algorithm is different.

#### 6.1.4.1 Combined Rate Model

We take the combined offered load  $q$  and solve for  $P_0[0]$  and  $T_0$  of the infinite queue (approximated as a continuous time M/D/1 as discussed in section 6.1.2.2). Then  $1 - P_0[0]$  is used as an input parameter to calculate the steady-state probability of the queue in the first stage,  $P_1[j]$ . We then calculate  $T_1$ ,  $d_1$  and  $S_0$  using the equations in section 6.1.2.3.  $1 - P_1[0]$  is then used as an input

parameter to calculate another set of variables, and so on. The algorithm is shown as follows :

```

solve  $P_0[0], T_0$ 
FOR  $i:=1$  to  $n$  DO
    solve  $P_i[j], 0 \leq j \leq K$ 
    solve  $T_i, d_{i-1}$ 
    solve  $S_{i-1}$ 
ENDFOR
solve  $S_n$  and  $1 - P_n[0]$ 

```

We vary the value of  $q$  to plot the throughput-delay curve.

#### 6.1.4.2 Feedback Model

We concentrate the feedbacks on processing element 1 in the following. Let  $Q_{i,j}$  be the  $j$ th queue in stage  $i$ . Note that for packets in the first stage queue can come from two different sources (see Figure 6.1). If a packet is rejected before it enters  $Q_{1,1}$ , it either came from  $Q_{0,1}$  (PE 1) or  $Q_{0,3}$  (PE 3). Hence with probability 0.5, the rejected packet returns to  $Q_{0,1}$ . The same argument applies to a packet rejected at  $Q_{1,2}$ . Therefore the total returning rate from stage 1 back to  $Q_{0,1}$  is just the rejecting probability at stage 1. Let  $d[i,j]$  be the probability that a packet is rejected at the  $j$ th queue in stage  $i$ . Then the rejecting probability for the uniform traffic case is as follows :

$$d[i,j] = P_i[K] \cdot \left(\frac{1}{2} \cdot (1 - P_{i-1}[0])\right)^2$$

For a packet rejected at  $Q_{2,1}$ , it either came from  $Q_{1,1}$  or  $Q_{1,3}$ . A packet from  $Q_{1,1}$  came from either  $Q_{0,1}$  or  $Q_{0,3}$ . Hence with probability 0.25, the rejected



packet returns to  $Q_{0,1}$ . The path originates from  $Q_{0,1}$  have 4 possible rejecting points at stage 2. Therefore the total rejected traffic from stage 2 back to  $Q_{0,1}$  is the rejecting probability at stage 2. Similarly, the path has 3 possible rejecting points in stage 3, and so on. To incorporate this feedback traffic to the input process, we add an iterative process to compute the rejecting probability in each stage. A new offered load is calculated at the end of each iteration. The iterative process terminates when the rejected traffic converges. This iterative algorithm is as follows :

```

ρ := q
REPEAT
    for stage 0, solve  $P_0[0], T_0$ 
    FOR i:=1 to n DO
        solve  $P_i[j], 0 \leq j \leq K$ 
        solve  $T_i, d_{i-1}$ 
        solve  $S_{i-1}$ 
    ENDFOR
    solve  $S_n$  and  $1 - P_n[0]$ 
     $\rho = \rho + \sum_{i=1}^n d[i, j]$ 
UNTIL ρ converges

```

For a non-uniform traffic pattern, queues in the same stage are not necessarily identical. Hence a more complicated tracing algorithm is needed to trace each branching flow. This will be discussed in Section 6.2.

### 6.1.5 Results

In this section, we would like to compare our analytical model to simulation results. A medium sized network (6 stage) and a large sized network (10 stage) will be compared to simulations. The Feedback Model and the Combined Rate Model is compared against each other. A brief summary of their advantages and disadvantages is given. The simultaneous model and the non-simultaneous model are also compared.

#### 6.1.5.1 Analytical Model vs. Simulation

We ran the Feedback Model for a 6-stage Omega network with buffer size 4 with a uniform traffic pattern. The results are plotted against simulations in Figure 6.4. The throughput-delay curve climbs slowly until it reaches the saturation point, at which point the delay increases rapidly. The maximum throughput we can get for the 6 stage Omega is 0.769 when the combined rate reaches 0.99. In Figures 6.6 and 6.7, we show throughput vs. offered load and mean delay vs. offered load, respectively with a 95 % confidence range simulation result. The simulation shows that the analytical model is very close to the simulation.

For a larger sized network, we ran the Combined Rate Model for a 10-stage Omega network with finite-buffer size 4 with a uniform traffic pattern. The result is shown in Figure 6.5. The maximum throughput the 10-stage Omega can achieve is 0.721 which is lower than the 6-stage Omega. The reason is that packets suffer more contention since there are 4 more stages. The simulation verifies the analytical model in a large sized network.

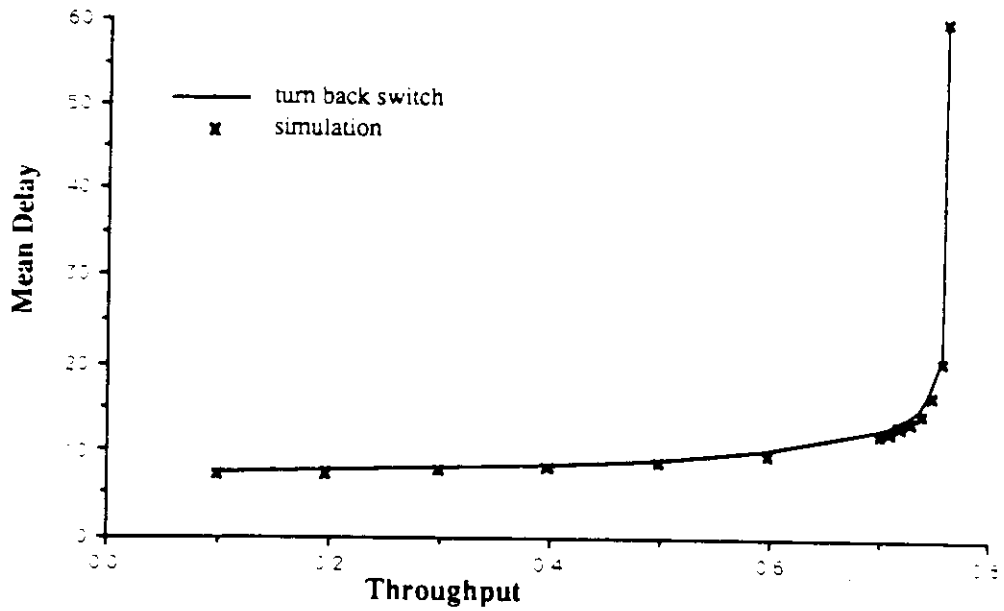


Figure 6.4: Analytical model vs. simulation for a 6-stage Omega

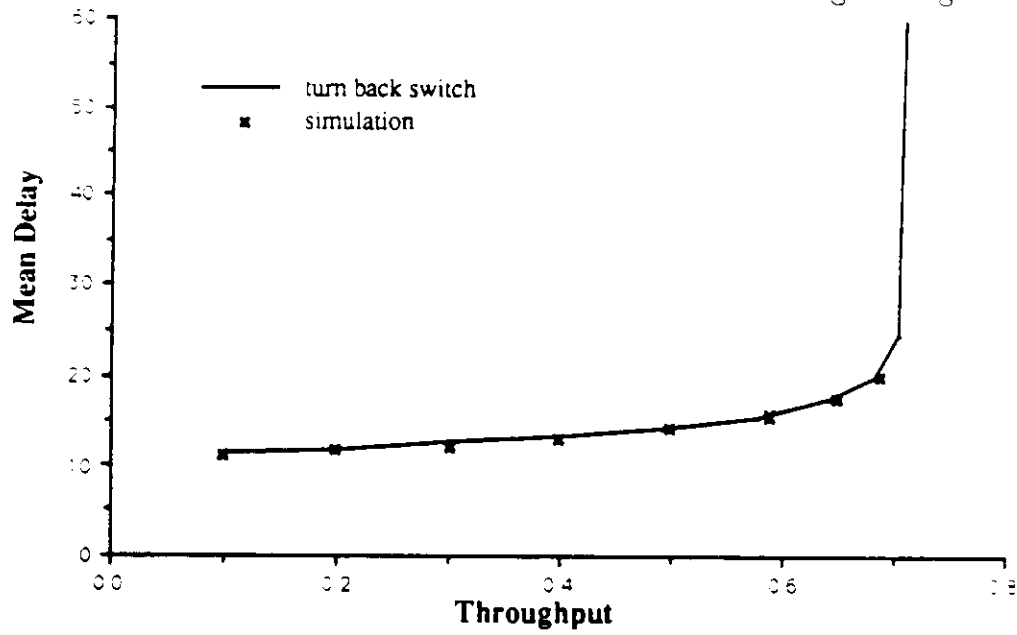


Figure 6.5: Analytical model vs. simulation for a 10-stage Omega

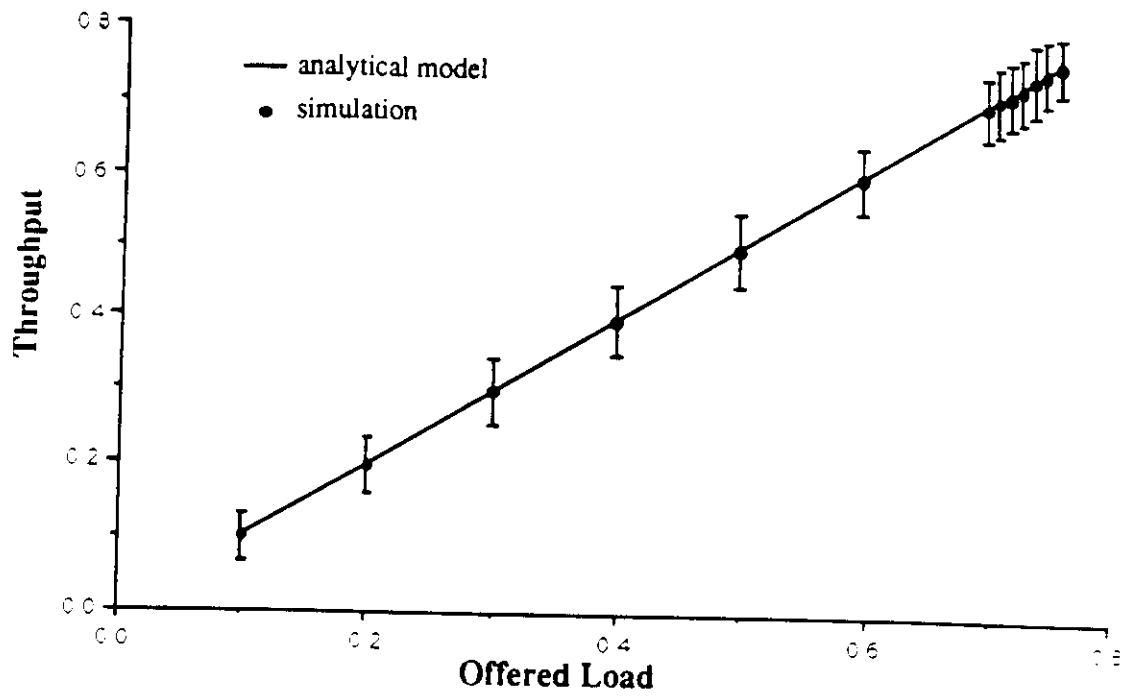


Figure 6.6: Analytical model vs. simulation for a 6 stage, 4 buffer Omega with infinite buffer at PE under uniform traffic pattern

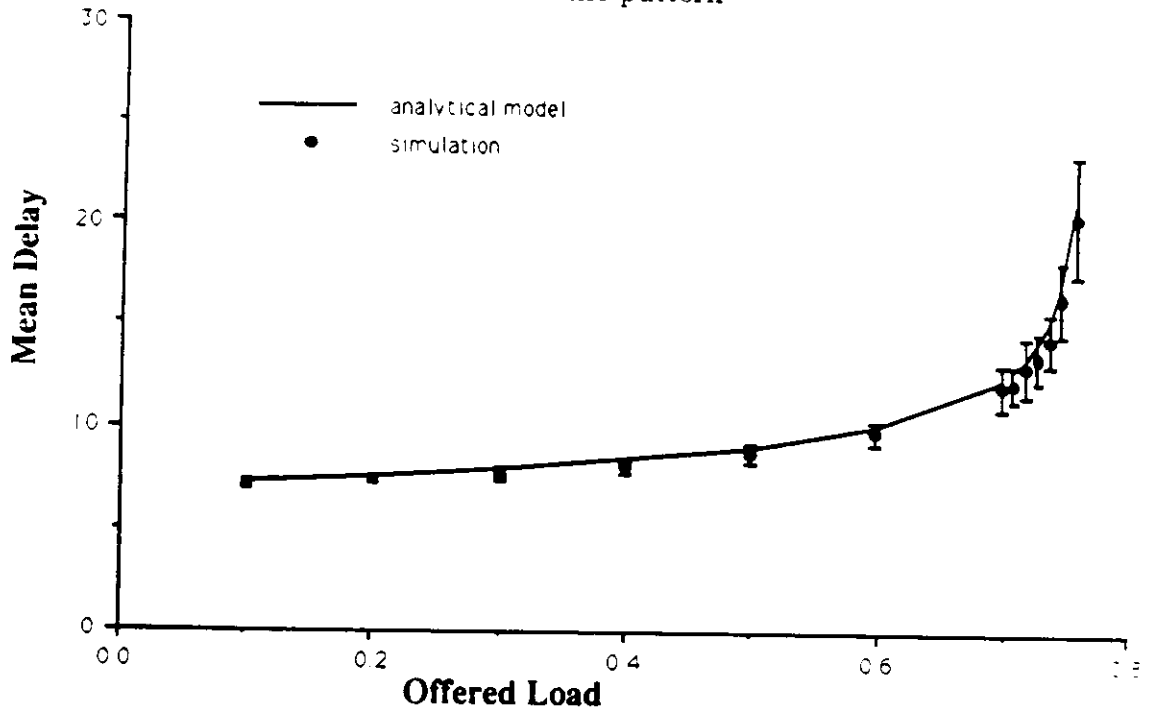


Figure 6.7: Analytical model vs. simulation for a 6 stage, 4 buffer Omega with infinite buffer at PE under uniform traffic pattern

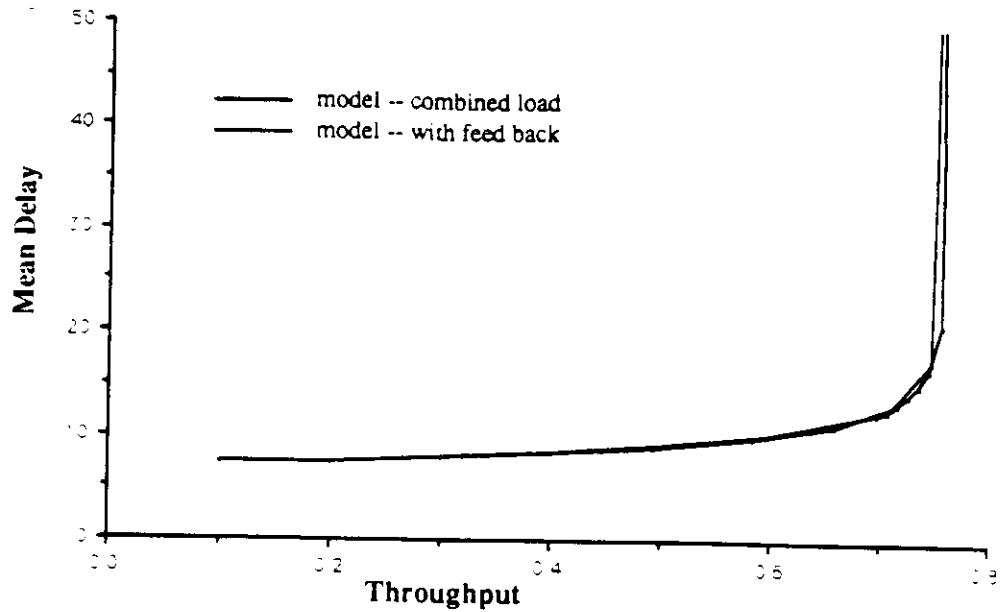


Figure 6.8: Feedback Model vs. Combined Rate Model

#### 6.1.5.2 Feedback Model vs. Combined Rate Model

Next, we compare the performance of models using different assumptions. The Feedback Model is compared to the Combined Rate Model in Figure 6.8. There is not much difference until late in the saturation area. The curve is basically the same for the uniform traffic pattern as predicted. The reason that those 2 curves do not match near the saturation area is that the combined rate model jumps from 0.9 to 0.99. If we were to run more detailed rates (0.91 to 0.99), then these 2 curves would likely match. The advantage of the Combined Rate Model is its simplicity and its straightforward solution algorithm. However, the exact amount of feed back traffic can not be determined. The advantage of the Feedback Model is that it gives more information about the resubmitted traffic. The corresponding amount of rejecting traffic from a given offered load can be solved for. In addition, the Feedback Model can be used to determine the maximal allowable input rate to operate within the stable range. This model

can also be extended to evaluate non-uniform traffic patterns.

### 6.1.5.3 Simultaneous Model vs. Non-simultaneous Model

In this section, the simultaneous model is compared to the non-simultaneous model for different assumptions on sending and receiving packets. Two different sized networks, 6 stage and 10 stage, are evaluated. The modelling approach we use is the Combined Rate Model. The results are shown in Figures 6.9 and 6.10. For light load, the throughput-delay curve for both models are almost the same. For medium to heavy load, the simultaneous model yields higher throughput. This behavior is more obvious when the network size gets larger. The reason behind this behavior is that when the queue reaches state  $K-1$ , the non-simultaneous model rejects one packet and accepts another if there are two incoming packets. The Simultaneous Model, however, accepts both, and therefore increases its maximum queue length to  $K$ .

## 6.2 The Turn Back Switch Model with a Non-uniform Traffic Pattern

In section 6.1, we proposed a simple modelling approach for performance modelling of a turn back switch with a uniform traffic pattern. The simplicity is due to the fact that the rejected packets have the same traffic patterns as the new packets do. Hence we can combine them into a single input source without distinguishing them. However, for non-uniform traffic pattern, this technique cannot be employed. The feedback traffic (rejected packets) does not have the same non-uniform traffic pattern as the new packets do. The contention and rejecting place more emphasis on the heavily referred memory models. Hence the rejected packets tend to have a more severe non-uniform pattern than the

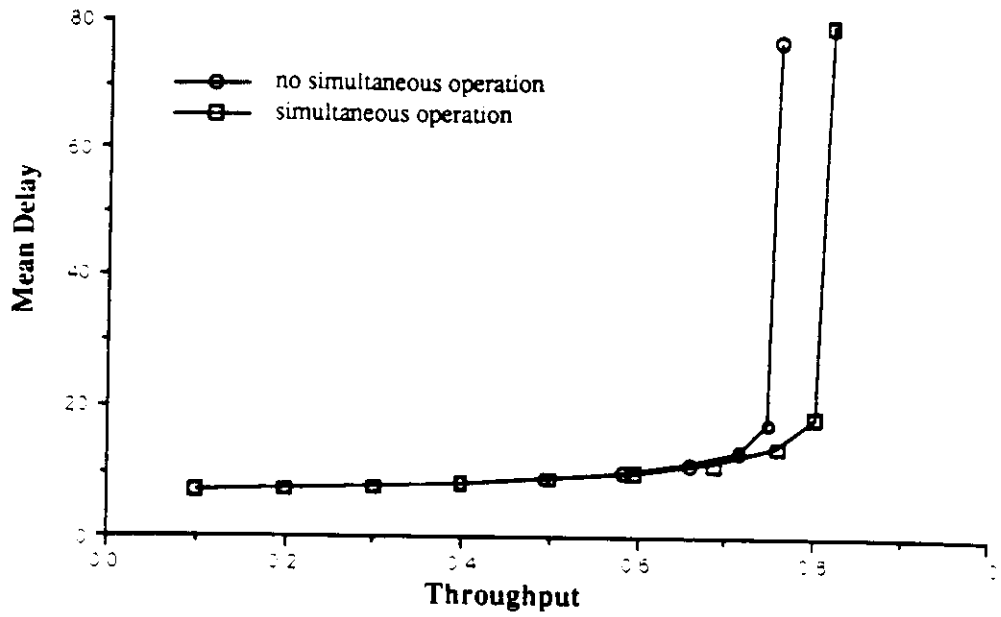


Figure 6.9: Simultaneous model vs. non-simultaneous model for a 6 stage Omega

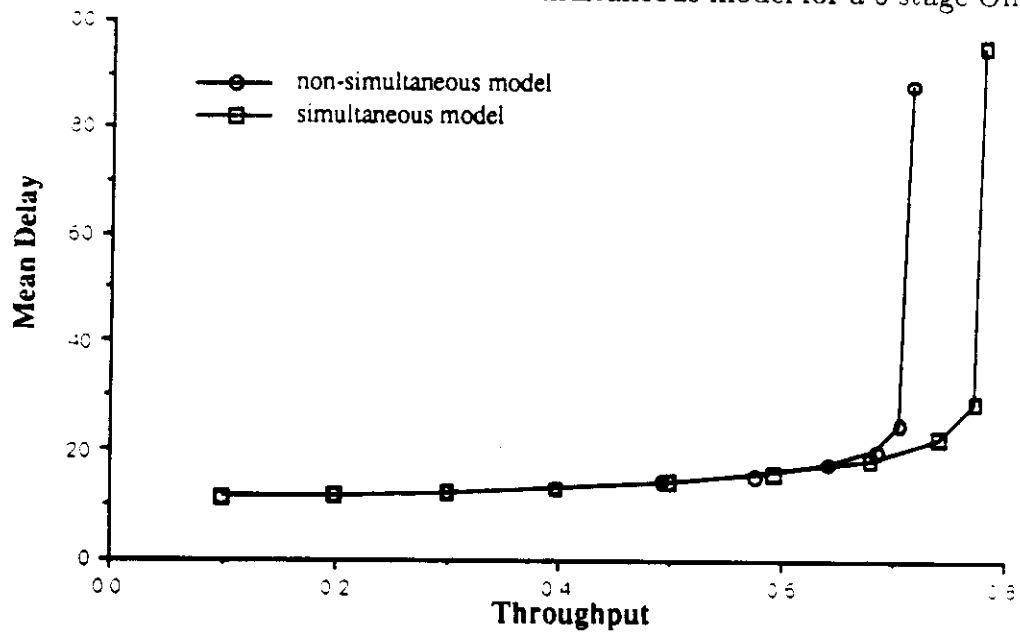


Figure 6.10: Simultaneous model vs. non-simultaneous model for a 10 stage Omega

new packets do. We must combine these two different sources to create a new traffic pattern.

The Feedback Model, as discussed in section 6.1.4.2, distinguishes the feedback packets from the new packets. However, each path originating from stage 0 to stage  $n$  of a general traffic pattern can be different, thus the modelling approach using the Feedback Model needs to be extended to a more general one. An extension of the basic Feedback Model to handle the general traffic model is discussed in section 6.2.1. For a particular path, general equations of the mean system time  $S_i$ 's are given. The rejected packets from this particular path is calculated such that a total feed back traffic can be found. We propose a recursive algorithm to find all possible paths. Results are verified through the simulation for 2 general traffic patterns : an EFOS pattern and a hot spot pattern.

## 6.2.1 Modelling Approach

### 6.2.1.1 Mean Delay Equations for a Particular Path

For the non-uniform traffic pattern, each path originated from a processing element to memory modules can be different. Depending on the address tag, packets may experience different delay by taking different paths. In the following, we concentrate on a particular path that a packet takes,  $j(0), j(1), \dots, j(n)$ , where  $j(i)$  is a variable representing the position of the queue in stage  $i$  along the path.

Let  $T_{i,j(i)}$  be the random variable representing the time spent in  $j(i)$ 's queue in stage  $i$ . Let  $d_{i,j(i)}$  be the conditional rejecting probability for the packet after it leaves stage  $i$  and before it enters stage  $i+1$ . Let  $S_i$  be the random variable representing the accumulated system time after a packet enters stage  $i+1$ . For a



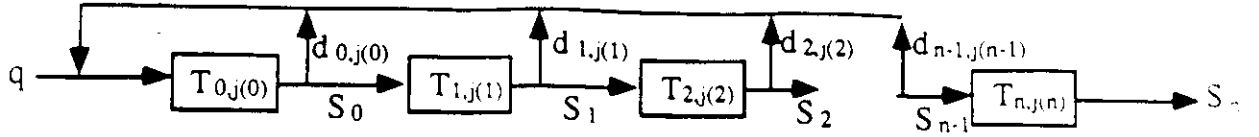


Figure 6.11: The recurrent relationship among variables in a particular path

packet taking this particular path, its mean delay can be calculated as

$$\begin{aligned}
 S_0 &= \frac{T_{0,j(0)}}{1 - d_{0,j(0)}} \\
 S_i &= \frac{S_{i-1} + T_{i,j(i)}}{1 - d_{i,j(i)}} \quad 1 \leq i < n \\
 S_n &= S_{n-1} + T_{n,j(n)}
 \end{aligned}$$

The explicit expression for  $S_i$  in terms of  $T_{i,j(i)}$  and  $d_{i,j(i)}$  can be shown as follows:

$$S_i = \sum_{m=1}^i \frac{T_{m,j(m)}}{\prod_{k=0}^{i-m} (1 - d_{i+1-k,j(i+1-k)})}$$

Therefore, the average system time for passing through this particular path is

$$\begin{aligned}
 S_n &= S_{n-1} + T_{n,j(n)} \\
 &= \sum_{m=1}^{n-1} \frac{T_{m,j(m)}}{\prod_{k=0}^{n-m-1} (1 - d_{n-k,j(n-k)})} + T_{n,j(n)}
 \end{aligned}$$

The algorithm to calculate  $S_n$  is given as follows :

**procedure**PATH-DELAY

**begin**

for stage 0, solve  $P_{0,j(0)}[0], T_{0,j(0)}$

**FOR**  $i:=1$  to  $n$  **DO**

**solve**  $P_{i,j(i)}[k], 0 \leq k \leq K$

**solve**  $T_{i,j(i)}, d_{i-1,j(i-1)}$

```

        solve  $S_{i-1}$ 
    ENDFOR
    solve  $S_n$  and  $1 - P_{n,j(n)}[0]$ 
end

```

### 6.2.1.2 Path Finding Algorithm

Once the address of a path is given, the mean delay of this particular path can be calculated using **PATH-DELAY**. To calculate the mean delay for the overall traffic, the mean delays for all possible paths are calculated and averaged over their proportions. A recursive algorithm that traces all possible paths starting from one processing element is outlined as follows :

```

procedure PATH-TRACE (i,j,portion)
begin
    j(i):=k;
    IF  $i < n$  THEN
        use j(i) to find its destination, des, in stage i+1 (i.e. j(i+1))
        use des to find its routing probability routing
        PATH-TRACE (i+1, des, portion · routing)
        PATH-TRACE (i+1, des+1, portion · (1-routing))
    ELSE
        use the path information j(i)'s as inputs, call PATH-DELAY
        accumulate total delay by  $S_{n,j(n)} \cdot$  portion
    end

```

To find all possible paths that originate from a processing element  $k$ , we call the above algorithm, **PATH-TRACE**(0, $k$ ,1). It is a depth-first algorithm such that the path from processing element  $k$  to memory module 1 is reached first. It then reaches memory module 2, and so on. After the trace reaches memory module  $2^n$ , the algorithm terminates. The mean delays along different paths are properly accumulated and calculated according to their contribution to the total delay. The variable **portion** is the product of the routing probabilities from stage 1 to stage  $n$  along the path. So, when the path reaches stage  $n$ , the mean delay is calculated and multiplied by **portion**. The weighted delay is then added to the total delay. This algorithm calculates the mean delay for packets originating from processing element  $k$ .

### 6.2.1.3 Updating Traffic Pattern

The main difference between the modelling of a non-uniform traffic pattern and the modelling of a uniform traffic pattern is the changes of the pattern. The network begins with a given traffic pattern. After packets being rejected and re-submitted, the overall pattern for all packets (new and resubmitted) is changed. This process repeats until it reaches a steady state. The non-uniform traffic pattern evolves from the original traffic pattern to some intermediate patterns and finally settles down to a new steady-state pattern. Unlike the Feedback Model in uniform traffic pattern modelling, each feedback traffic in the non uniform traffic case may be different. Therefore, the modelling approach for non-uniform traffic pattern is more complicated than the Feedback Model we discussed in section 6.1.4.2.

To understand the total feedbacks on PE 1, let us examine a 3-stage,  $8 \times 8$  MIN. Let  $d[i,j]$  be the rejecting probability of  $j$ th queue at stage  $i$ , as we defined

	PE	stage 1		stage 2				stage 3								
		q	d[1,1]	d[1,2]	d[2,1]	d[2,2]	d[2,3]	d[2,4]	d[3,1]	d[3,2]	d[3,3]	d[3,4]	d[3,5]	d[3,6]	d[3,7]	d[3,8]
pac[1]	a	$\frac{a}{a+3b}$		$\frac{a}{a+b}$				1								
pac[2]	b	$\frac{b}{a+3b}$		$\frac{b}{a+b}$					1							
pac[3]	b	$\frac{b}{a+3b}$			$\frac{1}{2}$					1						
pac[4]	b	$\frac{b}{a+3b}$			$\frac{1}{2}$						1					
pac[5]	b		$\frac{1}{4}$			$\frac{1}{2}$						1				
pac[6]	b		$\frac{1}{4}$			$\frac{1}{2}$							1			
pac[7]	b		$\frac{1}{4}$				$\frac{1}{2}$							1		
pac[8]	b		$\frac{1}{4}$				$\frac{1}{2}$								1	

Figure 6.12: The effect of feed back from different queues and different stages to PE 1

in section 6.1.4.2. Let  $\text{pac}[i]$  be the probability of accessing memory module  $i$ . A traffic pattern is defined through the values of  $\text{pac}[i]$ . The impact the rejected traffic has on the traffic pattern of PE 1 of a 3-stage MIN is shown in Figure 6.12. At stage 0, PE 1 submits the new packets to the network with a traffic pattern that sends a portion  $a$  of the new traffic to memory module 1 and sends a portion  $b$  of the new traffic to each of the other memory modules ( $a + 7 \cdot b = 1$ ). A rejecting may occur at  $Q_{1,1}$  with probability  $d[1,1]$ . Memory modules 1 to 4 are the possible destinations for a rejected packet at  $Q_{1,1}$ . The rejected packet may head for memory module 1 with probability  $\frac{a}{a+3b}$ . With probability  $\frac{b}{a+3b}$ , the rejected packet is destined for memory modules 2, 3 or 4. Therefore, the feedback traffic resulting from contention at queue 1.1 (with probability  $d[1,1]$ ) only contributes to the traffic pattern of memory module 1 to 4. A packet rejected at  $Q_{1,2}$  (with probability  $d[1,2]$ ) has memory modules 5 to 8 as its possible destinations. The rejected packet is destined to one of these four

memory modules with an equal probability  $\frac{1}{4}$ . Therefore, a rejecting that occurs at  $Q_{1,2}$  contributes uniformly to memory module 5 to 8. The effect of rejecting packets from different queues at different stages on the traffic pattern for a 3 stage network is outlined in Figure 6.12. However, a rejected packet at stage 1 does not necessarily come from PE 1. Since the contention involves 2 different sources, the rejected packet returns to PE 1 with probability  $\frac{1}{2}$ . Similarly, a rejected packet at  $Q_{2,1}$  may be destined for memory module 1 with probability  $\frac{a}{a+b}$  or destined for memory module 2 with  $\frac{b}{a+b}$ . Since packets from 4 PE's may possibly be involved with a rejecting that occurs at stage 2, the rejected packet returns to PE 1 with probability  $\frac{1}{4}$ . Therefore, with the given traffic pattern, the resulting new patterns for PE 1 are :

$$\begin{aligned}
pac[1] &:= a \cdot q + \frac{a}{a+3b} \cdot d[1,1] \cdot \frac{1}{2} + \frac{a}{a+b} \cdot d[2,1] \cdot \frac{1}{4} + 1 \cdot d[3,1] \cdot \frac{1}{8} \\
pac[2] &:= b \cdot q + \frac{b}{a+3b} \cdot d[1,1] \cdot \frac{1}{2} + \frac{a}{a+b} \cdot d[2,1] \cdot \frac{1}{4} + 1 \cdot d[3,2] \cdot \frac{1}{8} \\
pac[3] &:= b \cdot q + \frac{b}{a+3b} \cdot d[1,1] \cdot \frac{1}{2} + \frac{1}{2} \cdot d[2,2] \cdot \frac{1}{4} + 1 \cdot d[3,3] \cdot \frac{1}{8} \\
pac[4] &:= b \cdot q + \frac{b}{a+3b} \cdot d[1,1] \cdot \frac{1}{2} + \frac{1}{2} \cdot d[2,2] \cdot \frac{1}{4} + 1 \cdot d[3,4] \cdot \frac{1}{8} \\
pac[5] &:= b \cdot q + \frac{1}{4} \cdot d[1,2] \cdot \frac{1}{2} + \frac{1}{2} \cdot d[2,3] \cdot \frac{1}{4} + 1 \cdot d[3,5] \cdot \frac{1}{8} \\
pac[6] &:= b \cdot q + \frac{1}{4} \cdot d[1,2] \cdot \frac{1}{2} + \frac{1}{2} \cdot d[2,3] \cdot \frac{1}{4} + 1 \cdot d[3,6] \cdot \frac{1}{8} \\
pac[7] &:= b \cdot q + \frac{1}{4} \cdot d[1,2] \cdot \frac{1}{2} + \frac{1}{2} \cdot d[2,4] \cdot \frac{1}{4} + 1 \cdot d[3,7] \cdot \frac{1}{8} \\
pac[8] &:= b \cdot q + \frac{1}{4} \cdot d[1,2] \cdot \frac{1}{2} + \frac{1}{2} \cdot d[2,4] \cdot \frac{1}{4} + 1 \cdot d[3,8] \cdot \frac{1}{8}
\end{aligned}$$

The new traffic patterns for other PE's can be calculated in a similar way.

The new traffic pattern is then used to generate another new traffic pattern. This process repeats until it converges. The complete algorithm that determines the temporary, new traffic pattern is as follows :

```

procédure ROUTE-TRACE (i,j)
begin
    use j to find its destination, des, in stage i+1
    calculate d[i+1, des] and d[i+1, des+1]
    IF  $i < n - 1$  THEN
        ROUTE-TRACE (i+1, des)
        ROUTE-TRACE (i+1, des+1)
    calculate new accessing probabilities
end

```

We call this procedure **ROUTE-PATH**(0,1) to begin the calculation for the new pattern of PE 1. If the processing elements are not identical, **ROUTE-PATH**(0,j) are called for all processing elements j's. If processing elements are identical, then **ROUTE-PATH**(0,1) is called only once to find the new traffic pattern.

### 6.2.2 Solution Algorithm

The turn back switch model uses a routing probability set to represent the steady state traffic flow. The transformation and the superposition methods that we proposed in Chapter 2 and 3 are used to transform a given non-uniform traffic pattern to a set of routing probabilities. Then, the modelling approaches that we propose in this section are incorporated into the following complete solution algorithm to solve for a turn back switch model:

#### INITIALIZATION

$\rho := q$

```

repeat
  for all  $j$ 's  $1 \leq j \leq 2^n$ , solve  $P_{0,j}[0]$  and  $T_{0,j}$ 
  for  $i:=1$  to  $n$  do
    for  $j:=1$  to  $2^n$  do
      solve  $P_{i,j}[k]$ ,  $0 \leq k \leq K$ 
      solve  $T_{i,j}$ 
    endfor
  endfor
  for  $j:=1$  to  $2^n$  do
    call PATH-TRACE ( $0j,1$ )
    accumulate mean delay and throughput
  endfor
  for  $j:=1$  to  $2^n$  do
    call ROUTE-TRACE ( $0j$ )
    accumulate new accessing probabilities and new input load  $\rho$ 
  endfor
  use new traffic pattern, solve new routing probabilities
until throughput converges or  $\rho \geq 1$ 

```

The algorithm begins with an initialization program that sets the initial values of variables, and uses the transformation and the superposition methods to calculate the routing probability set. Then the iterative process begins with the initial load and initial traffic pattern to calculate the mean delay and the throughput for the network. A new traffic pattern and a new offered load (combine both the new packets and the rejected packets) are calculated. This iterative process

stops when the throughput value converges or when the offered load exceeds 1. The first condition is the case when the system reaches steady state. The second condition is the case when the system reaches the unstable region.

This analytical model for the non-uniform traffic pattern incorporates the transformation and the superposition methods that we proposed for the general traffic pattern modelling in Chapter 2 and 3; therefore it not only is suitable for non-uniform traffic patterns, but also can be used to evaluate any general traffic conditions, for examples, different traffic patterns for processing elements, different input rates for processing elements, etc. We shall show the flexibility of this analytical model by evaluating two traffic patterns, a hot spot traffic pattern and an EFOS traffic pattern, and verify them with simulations.

### 6.2.3 Results

The analytical model for the non-uniform traffic pattern is used to evaluate a hot spot traffic pattern which sends  $\frac{3}{66}$  of the traffic to memory module 1 and  $\frac{1}{66}$  of the traffic uniformly to other memory modules in a 6 stage Omega network. The result is shown in Figure 6.13 with simulation data. The comparison shows good correspondence between analytical results and simulation data.

For a general traffic pattern, we choose the EFOS traffic pattern. Analytical results for turn back switch with an EFOS pattern are verified through simulation in Figure 6.14. The comparison indicates that the analytical model is quite accurate.

From these 2 examples, we see that the analytical model for a MIN using turn back switch as basic building block is a very good model under any general traffic patterns. For a uniform traffic pattern, a simpler model can be found in section 6.1.



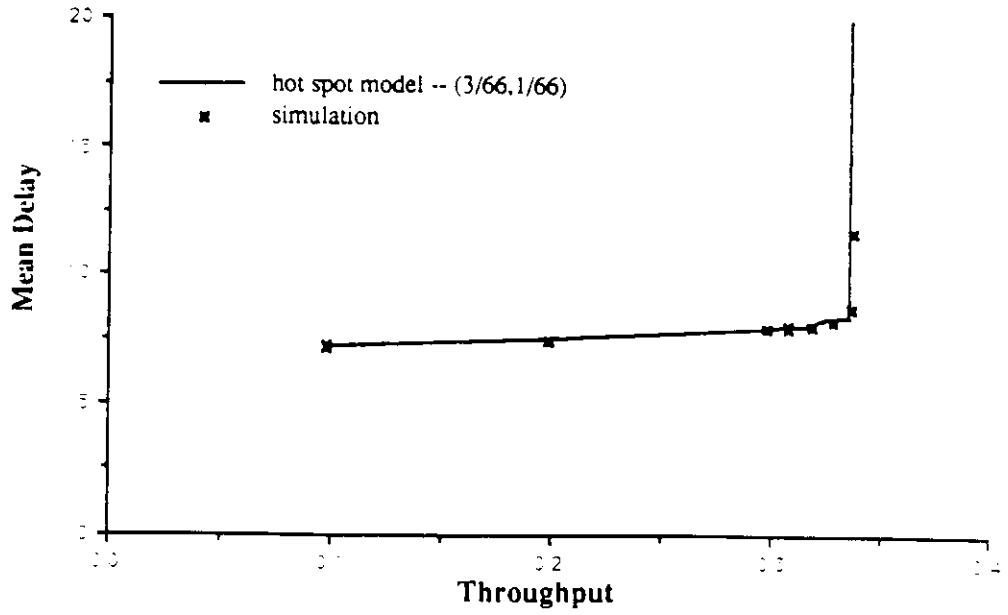


Figure 6.13: Analytical model vs. simulation for a 6 stage Omega under hot spot traffic pattern

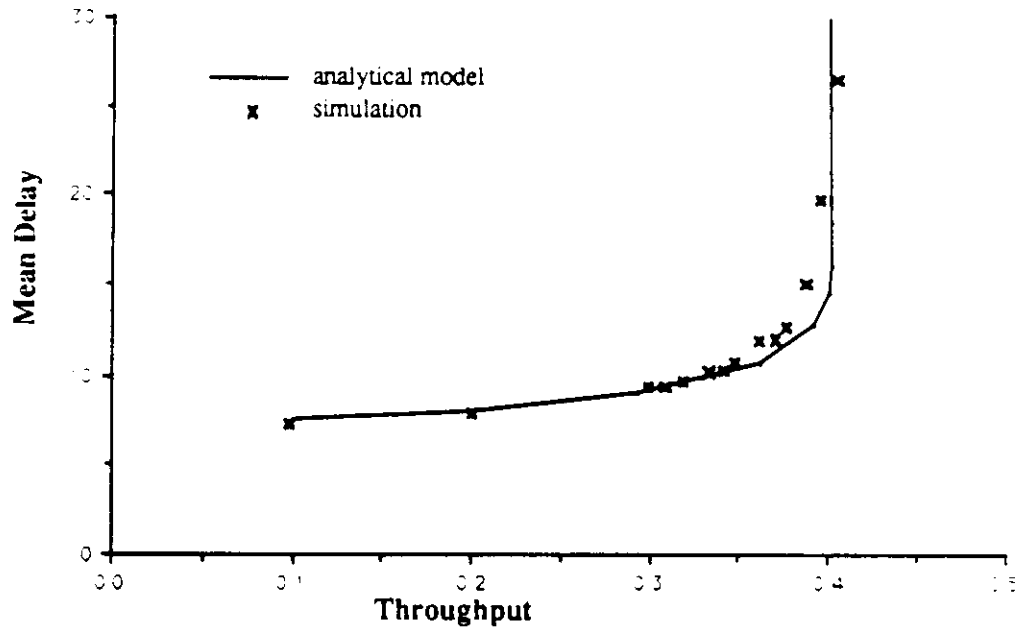


Figure 6.14: Analytical model vs. simulation for a 6 stage Omega under EFOS pattern

### 6.3 The Blocking Switch

The blocking switch model has been discussed in Chapters 2 and 3. However, the model did not include an infinite queue at the PE. The model only deals with packets that enter the network. Packets which are unable to enter the network are rejected. In order to compare the performance of the blocking switch with the turn back switch, the blocking switch model should be modified such that it includes an infinite queue at PE to accommodate those rejected packets. The modelling approach for this modification is described in section 6.3.1. A method to calculate the mean delays of a particular path is also discussed. The performance comparison of the blocking switch and the turn back switch is discussed in section 6.3.2.

#### 6.3.1 Modelling Approach

##### 6.3.1.1 Infinite Queue Model

We modify the general traffic pattern model that we proposed in section 3.2.5 to incorporate an infinite queue at the processing element. The PE generates a new packet and places it at the end of the infinite queue. The server of the infinite queue tries to submit the head-of-line packet into the network in every cycle. If there is buffer space in stage 1, the packet is accepted by the queue in stage 1. If there is no space available, the packet stays in the infinite queue and tries again in the next cycle. When the network reaches a steady state, the infinite queue server will have a steady state blocking probability that a given packet is blocked in a cycle. Let  $q$  be the probability that a new packet is generated in one cycle. Let  $b$  be the blocking probability that a packet in the server is blocked. The infinite queue begins at state 0. It moves to state 1 when there is a new

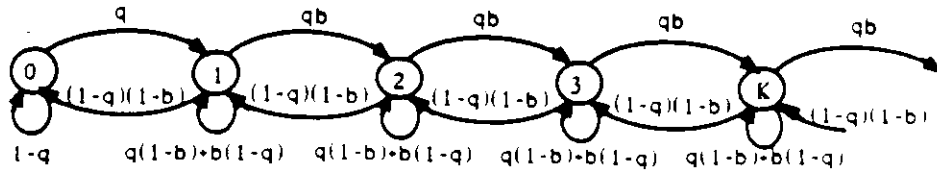


Figure 6.15: Markov chain for a discrete time queue at PE

packet being generated (with probability  $q$ ). The queue moves from state 1 to state 2 if there is one incoming packet and the packet in the server is blocked. The transition probability for this case is  $q \cdot b$ . The queue returns to state 0 if there is no new packet coming and the packet in the server is not blocked. The transition probability is  $(1 - q) \cdot (1 - b)$ . Since this discrete time, discrete state queue can only make unit step transitions, it remains at state 1 with probability  $1 - q \cdot b - (1 - q) \cdot (1 - b)$ . The other states have the same transitions as the state 1 does. The Markov chain for this discrete time, discrete state queue is shown in Figure 6.15. The steady state probabilities can be solved as follows :

$$P_0 = 1 - \frac{q}{1 - b}$$

$$P_k = \frac{1}{b} \cdot \left( \frac{qb}{(1 - q)(1 - b)} \right)^k \cdot P_0 \quad k \geq 1$$

The mean queue length is :

$$\bar{V} = \sum_{k=1}^{\infty} k \cdot P_k = \frac{q(1 - q)}{1 - b - q}$$

Using Little's result, we get the mean delays :

$$T = \frac{\bar{V}}{q} = \frac{1 - q}{1 - b - q}$$

Including this delay in the blocking switch model that we proposed in Chapter 2 and 3, we get the mean delay for a packet passing through the queue at PE and the interconnection networks.

### 6.3.1.2 Delay Calculation along a Particular Path

In section 6.2.1.1, we proposed a method to calculate the mean delay along a particular path in an interconnection network using turn back switches. It provides more information in addition to overall system delays. The method is especially useful if the traffic pattern is not uniform. The delay of a particular path of interest can be determined.

The mean delay of a particular path in an interconnection network using the blocking switch can also be determined. Given a path,  $j(0), j(1), \dots, j(n)$  where  $j(i)$  represents the index of a queue in stage  $i$ , the mean delay can be determined by summing the delays in each stage. When the system reaches steady state, the state probabilities of each queue are known. We first calculate the mean queue length using the steady state probabilities. The throughput of each queue can be calculated using  $P_0$  and the queue's blocking probability. We then apply Little's result to each queue to solve for the mean delay. Summing the delays along the particular path, we get the mean delay for a particular path.

### 6.3.2 Results

We compare the blocking switch model to the turn back switch model in Figure 6.16. For light load, the blocking switch has lower delay than the turn back switch does. When the system load is small, the contention probability is very small; therefore persistent blocking does not cause severe system degradation. If a contention occurs, the turn back switch rejects the packet and resubmits it at PE while the blocking switch simply blocks it. Since the contention probability is small, the blocked packet is most likely to get through in next cycle while in the turn back switch, it has to begin from stage 0. Hence, a packet suffers more delay

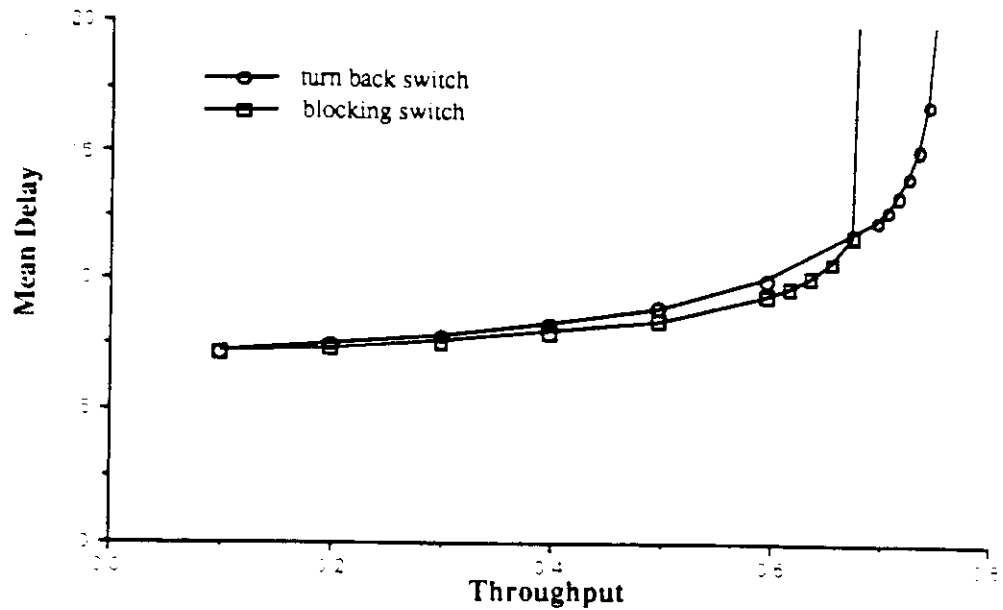


Figure 6.16: Blocking switch vs. turn back switch for a 6 stage Omega under uniform traffic pattern

in a turn back switch when the system load is small. However, the throughput-delay curve of blocking switch jumps up rapidly when the system load reaches its maximal capacity. When the system load reaches 0.68, the throughput-delay curve of the blocking switch crosses the curve of the turn back switch and quickly reaches saturation. The turn back switch performs better under heavy system load. The maximal throughput the turn back switch can achieve is 0.760. The reason that the turn back switch performs better in heavy load is due to rejecting the collided packets. Rejecting packets removes the persistent blocking behavior, the main reason behind the saturated tree. A blocked packet keeps blocking other packets as long as it occupies the server. A rejected packet returns to PE, thus the packet that was behind it will not be persistently blocked as the blocking switch does. Therefore a higher throughput can be achieved.

## 6.4 The Rotating Switch Model

Observing the different advantages the blocking switch and the turn back switch have in different load ranges, we propose a new switching element that shares the advantages of the both blocking switch and the turn back switch. The advantage of the blocking switch lies in the light system load range. During this range, a packet losing a contention is blocked locally, instead of being rejected and returned to the processing element. Thus a lower delay can be achieved. The advantage of the turn back switch is the removal of persistent blocking. The new switching scheme we propose combines both advantage. We would like to retain the locality principle of the blocking switch and the rejecting scheme of the turn back switch.

A rotating switch is basically a kind of turn back switch. When a contending packet finds no buffer space or loses a contention, it is rejected. The difference between a rotating switch and a turn back switch is where to re-submit the rejected packet. The turn back switch resubmits it to processing element queue. The rotating switch rotates the rejected packet from the server to the end of the queue. This rejecting scheme not only removes the persistent blocking, but also re-submits a rejected packet in a local queue. The two different advantages that a blocking switch and a turn back switch each has are now combined in the rotating switch. Therefore the performance of the rotating switch should be better than the performance of both the blocking switch and the turn back switch. A rotating switch retains the rejecting operation (removing the persistent blocking) and rotates the rejected packet to the end of the queue (locality principle). The configuration of a rotating switch is shown in Figure 6.17.

The Markov chain of a rotating switch is exactly the same one as discussed in

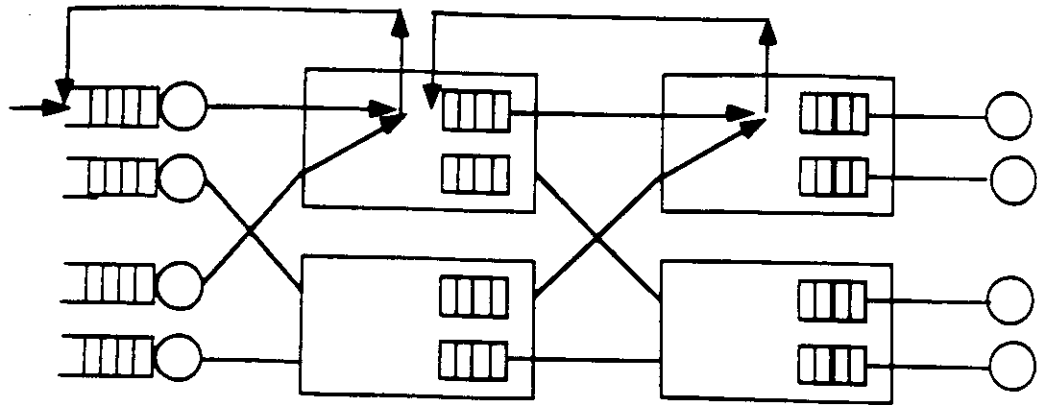


Figure 6.17: A 2 stage 4x4 interconnection network with rotating switches

Figure 3.2. Instead of being blocked, a packet is rotated to the end of the queue. For a uniform traffic pattern, the rotating scheme means that in each cycle a packet in the server has a renewal choice of destination. The rotated packet is treated as a new packet being submitted to the end of the queue. Therefore the modelling of the rotating switch is simple : we use the analytical model proposed in Chapter 3, with the addition of an infinite queue at the processing elements. The infinite queue is modelled as a discrete time queue as we discussed in section 6.3.1.1. The model in Chapter 3 assumes that a packet has a renewal output choice in every switch. The analytical result is shown in Figure 6.13. The network that we modelled is a 6 stage, finite buffer size 4 in each switching element and an infinite queues at PE's.

As predicted, the throughput-delay curve is better than the curves of the blocking switch and the turn back switch. For light load cases ( $\rho \leq 0.5$ ), the rotating switch is almost the same as the blocking switch. A packet losing a contention is either blocked locally or rotate to the end of local queue. Since the blocking probability for these two models are the same, re-inserting the packet at the server or at the end of the queue has the same delay performance. However,

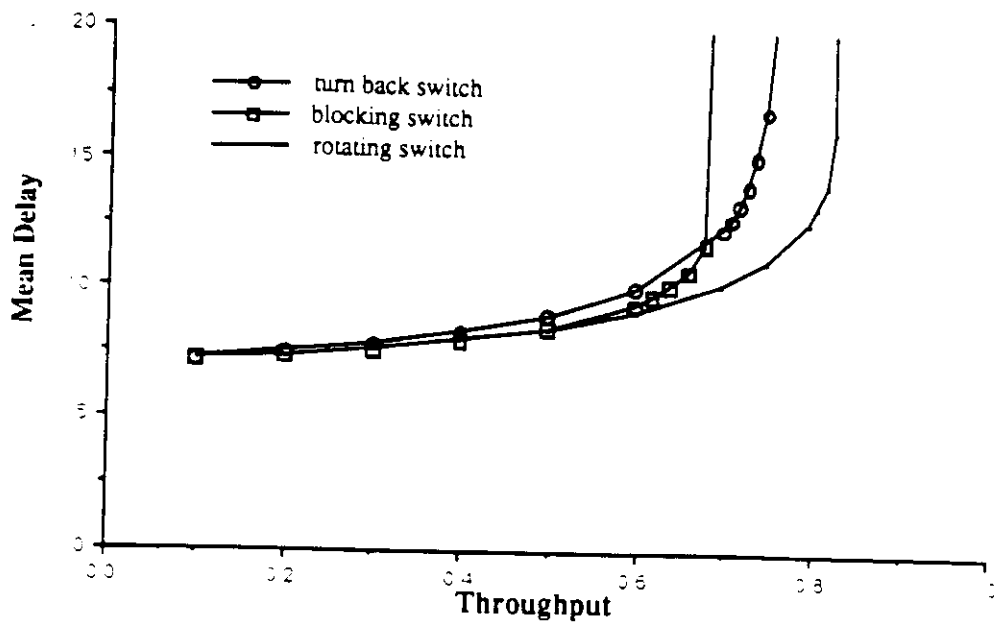


Figure 6.18: Analytical result for a 6 stage rotating switches under uniform traffic pattern

since the blocking switch still has a small probability of persistent blocking, the blocking switch has a slightly higher delay when the offered load lies in the range between 0.4 and 0.5 (the difference is too small to be shown on the figure). Increasing the system load, the rotating switch is much better than the blocking switch. The reason is the removal of persistent blocking. In heavy system load cases, persistent blocking severely degrades system performance.

The rotating switch is better than the turn back switch in every aspect. When the system load is heavy, the rotating switch outperforms the re-submission switch. The maximal throughput the rotating switch can achieve is 0.832. The turn back switch can only achieve 0.76. The difference between these two models is where to re-submit. We expect the turn back switch to have a higher delay due to re-submission at processing elements. A rejected packet has to go through contention and rejecting again in each stage. Surprisingly, the rotating switch



has a higher throughput than the turn back switch. This better throughput performance results from the different Markov chain assumption. When a queue reaches state  $K-1$ , the rotating switch can reach state  $K$  (full buffer) in next cycle. For the turn back switch, the queue can only remain at state  $K-1$  or move down to state  $K-2$  (see the discussion in section 6.1.2.1). The rotating switch can accept only one packet when it is in state  $K-1$ . But the packet in the server might be blocked, and resubmitted to the end of the queue. The queue thus moves to state  $K$ . In the same situation, the turn back switch rejects the packet and re-submits it to the processing element, hence state  $K$  is never reached. Therefore the rotating switch has a higher throughput than the turn back switch.

The rotating switch combines the advantages of the locality principle (blocking switch) and the removal of persistent blocking (reject switch). As a result, the throughput-delay curve is better than the other two switches.

## CHAPTER 7

### Summary and Future Research

#### 7.1 Summary

In this dissertation, several analytical models were proposed to approximate a finite-buffered interconnection network with general traffic patterns and different switching architectures. These models were analyzed using an iterative approach. The blocking switch models were presented in Chapter 2, 3, 4 and 5. The turn back switch model and the rotating switch model were presented in Chapter 6.

A decomposition and iteration model was proposed in Chapter 2 to analyze an interconnection network with both a specific hot spot traffic pattern and a uniform traffic pattern. It was shown that contention for the same output queue degrades the system performance in the unbuffered case. With a finite number of buffers added at the output ports of each switching element, the system performance was improved significantly with a uniform traffic pattern. However, buffering does not improve the system performance to a satisfactory degree when a severe hot spot traffic pattern was presented. This indicates that buffering cannot resolve the hot spot problem. Additional means for controlling the hot spot is needed.

In all these models, we used an iterative approach to solve for the probability of acceptance from which we found the mean delay. The saturated tree was analyzed through the calculation of the mean busy buffer size in the tree. A method for calculating an upper bound for the tree build up time was proposed and com-

pared to other analytical results. With the renewal routing choice assumption, the simulation indicated that the decomposition and iteration model is very accurate for networks of any size. The analytical model was extended to analyze an interconnection network with any general traffic pattern. A transformation method was proposed to transform a given traffic pattern (in terms of accessing probabilities) into a set of routing probabilities. The routing probabilities reflect the steady state behavior of the traffic pattern. Incorporating the transformation method in the decomposition and iteration model, the performance of an interconnection network whose processing elements have the same general output traffic pattern can be analyzed.

A superposition method was proposed in Chapter 3 to analyze an interconnection network whose processing elements have their own output traffic patterns. A weighting factor was proposed to be incorporated in the superposition method to analyze an interconnection network whose processing elements have their own input rates. The Not Uniform Traffic Spot (NUTS) traffic patterns were analyzed as examples. The NUTS traffic pattern is a special pattern which seems uniform outside the network, but forms congested spots along the overlapping paths inside the network. Without the renewal routing choice assumption, the simulation indicated a significant discrepancy between the analytical model and the simulation result. The discrepancy was attributed to the model's failure to capture the persistent blocking effect.

An approximation was proposed to capture the persistent blocking effect in Chapter 4. The steady state probability during which a server is blocked was calculated. We then assumed that a server is inactive with this probability. The simulation showed that this approximation greatly diminished the discrepancy between the simulation and the analytical model. The approximation is very

accurate for both a uniform traffic pattern and a EFOS traffic pattern. However, the approximation does not totally capture the effect of persistent blocking with a severe hot spot traffic pattern. This remains as a future research topic.

An extension of the general traffic pattern model was proposed in Chapter 5 where each processing element has a finite number of buffers to accommodate the rejected packets. Simulation indicated the model was very accurate. Furthermore, a rate adjusted model was proposed to reduce the mean delays while maintaining the throughput. The mean queue length and the mean delays of models with and without the rate adjusted method were compared. It indicated a significant reduction in mean delay and mean queue length were achieved without sacrificing throughput performance. The maximal allowable input rate to satisfy a given loss probability was also determined.

An analytical model for a turn back switch was proposed in Chapter 6. A recurrence equation for the mean delay was proposed for an interconnection network with a uniform traffic pattern. A solution algorithm which combines both the Markov chain analysis and the mean delay calculation was discussed. The simulation showed that the re-submission model was very accurate for networks of different sizes. A detailed evaluation of the feedback traffic was proposed for an interconnection network with a non-uniform traffic pattern. The feedback traffic changes the overall traffic pattern if the original traffic pattern is not uniform. An iterative model which changes the traffic pattern in each iteration according to the feedback information was proposed. A hot spot traffic pattern and a EFOS traffic pattern were analyzed. The simulation showed good agreement for both traffic patterns. The turn back switch model was compared to the blocking switch model. The throughput-delay curve showed that each switching architecture is good in a certain range of offered load. This led us to propose a new

switching architecture, the rotating switch, which combines the advantages of both the turn back switch and the blocking switch. An analytical model for this new switching architecture was proposed, and its performance was compared to both the turn back switch and the blocking switch. The throughput-delay curve indicated that the rotating switch outperforms both the turn back switch and the blocking switch in every range of system load. The rotating switch model was analyzed only for an interconnection network with a uniform traffic pattern. An extended model which iteratively changes the traffic pattern to account for the feedback traffic is to be done in the future.

## 7.2 Future Research

There are some interesting problems for future research.

The first is to compare the three analytical models with different switching architectures for the rate adjusted case. This will show the performance of each model under the range of stable operation.

It is necessary to have a model for slow memory modules. The memory modules in the real world are not necessarily fast enough to take out a packet every cycle. Thus a relaxation on the speed of the memory modules is needed such that each memory module has its own speed.

In real world applications, a processing element might generate a message which contains several packets destined for the same memory module. This can be modelled as bulk arrivals to the processing element.

There may be different priority classes of packets as inputs in the real world. The controlling mechanisms for different switching architectures in order to achieve performance goals is an important issue which should be investigated.

Finally, the modelling and the methodology to control or to prevent hot spots

from degrading the system performance is needed.

- [Dias 81] D.M. Dias, J.R. Jump, "Analysis and Simulation of Buffered Delta Networks", *IEEE Transaction on Computers*, Vol.C.30, No.4, April 1981, pp.273-282
- [Dias 89] D.M. Dias, M. Kumar, "Preventing Congestion in Multistage Networks in the Presence of Hotspots", *1989 International Conference on Parallel Processing*, I9-I13
- [Feng 81] T. Feng, "A Survey of Interconnection Networks", *IEEE Computer*, December 1981, pp.12-27
- [Garg 88] U. Garg, Y. Huang, "Decomposing Banyan Networks for Performance Analysis", *IEEE Transaction on Computers*, Vol.C.37, No.3, March 1988, pp.371-376
- [Goke 73] L.R. Goke, G.J. Lipovski, "Banyan Networks for Partitioning Multiprocessor Systems", *The Proceedings of the First Annual Symposium on Computer Architecture*, 1973, pp21-28
- [Gott 83] A. Gottlieb, et al, "The NYU Ultracomputer - Designing an MIMD Shared Memory Parallel Computer", *IEEE Transaction on Computers*, Vol. C.32, No.2, February 1983, pp. 175-189.
- [Ho 89] W.S. Ho, D. L. Eager, "A Novel Strategy for Controlling Hot Spot Congestion", *1989 International Conference on Parallel Processing*, I-14-I-18.
- [Jenq 83] Y. Jenq, "Performance Analysis of a Packet Switch Based on Single-Buffered Banyan Network", *IEEE Journal on Selected Areas in Communications*, Vol. SAC-1, No. 6, December 1983, pp.1014-1021
- [Kamo 80] F. Kamoun, L. Kleinrock, "Analysis of Shared Finite Storage in a Computer Network Node Environment under General Traffic Conditions", *IEEE Transactions on Computers*, Vol. 28, No. 7, July 1980, pp.992-1003
- [Kim 90] H.S. Kim, A. Leon-Garcia, "Performance of Buffered Banyan Networks under Nonuniform Traffic Patterns", *IEEE Transactions on Computers*, Vol. 38 No. 5, May 1990, pp.648-658
- [Klei 79] L. Kleinrock, "Power and Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications" *International Conference on Communications*, June 1979, pp.43.1.1-43.1.10.
- [Krus 83] C.P. Kruskal, M. Snir, "The Performance of Multistage Interconnection Networks for Multiprocessors", *IEEE Transaction on Computers*, Vol. C.32, No. 12, December 1983, pp.1091-1098

- [Krus. 86] C.P. Kruskal, M. Snir, A. Weiss, "The Distribution of Waiting Times in Clocked Multistage Interconnection Networks", *Conference of Parallel Processing*, 1986, pp.12-19
- [Krus 88] C.P. Kruskal, M. Snir and A. Weiss, "The distribution of Waiting Times in Clocked Multistage Interconnection Network", *IEEE Transaction on Computers*, vol. 37, No. 11, November 1988, pp1337-1352
- [Kuma 87] V.P. Kumar, S.M. Reddy, "Augmented Shuffle-Exchange Multistage Interconnection Networks", *IEEE Computer*, June 1987, pp.30-40
- [Kuri 88] L.Kurisasi, T. Lang, "Multistage Networks with Traffic with Real-Time Constraints", *UCLA Computer Science Technical Report CSD-880101*
- [Lang 88] T. Lang, L. Kurisasi, "Nonuniform Traffic Spots (NUTS) in Multistage Interconnection Networks", *UCLA Computer Science Department Technical Report CSD-880001*, January 1988
- [Lawr 75] D.H. Lawrie, "Access and Alignment of Data in an Array Processor", *IEEE Transaction on Computers*, Vol.C-24, Dec. 1975, pp. 1145-1155
- [Lee 86] G. Lee, C.P. Kruskal, D.J. Kuck, "The Effectiveness of Combining in Shared Memory Parallel Computers in the presence of Hot Spots", *Proc. 1986 International Conference Parallel Processing*, Aug. 1986, pp.35-41
- [Lee 89] G. Lee, "A Performance Bound of Multistage Combining Networks", *IEEE Transactions on Computers*, Vol. 38, No. 10, October 1989, pp.1387-1395.
- [Lin 90] T. Lin and A. Tantawi, "Performance Evaluation of Packet-Switched Multistage Interconnection Networks under a Model of Hot-Spot Traffic", *ORSA/TIMS Joint National Meeting*, Philadelphia, PA, October 29-31, 1990.
- [Liu 89] Y. Liu, C. Wang, "Analysis of Prioritized Crossbar Multiprocessor Systems", *Journal of Parallel and Distributed Computing*, July 1989, pp.321-334
- [Mitr 87] D. Mitra, R.A. Cieslak, "Randomized Parallel Communications on an Extension of the Omega Network", *Journal of the Association for Computing Machinery*, Vol. 34, No. 4, October 1987, pp.802-824
- [Pate 81] J.H. Patel, "Processor-Memory Interconnections", *IEEE Transaction on Computers*, Vol. c-30, No. 10, October 1981, pp.306-310



- [Pate 88] N.M. Patel, P.G. Harrison, "On Hot-Spot Contention in Multistage Interconnection Networks", *ACM SIGMETRICS*, May 1988, pp. 114-123.
- [Perr 86] H.G. Perros, T. Altioik, "Approximate Analysis of Open Networks of Queues with Blocking : Tandem Configurations", *IEEE Transaction on Software Engineering*, Vol. SE-12, No. 3, March 1986, pp.450-461
- [Pfis 85] G.F. Pfister, V. A. Norton, "Hot-Spot Contention and Combining in Multistage Interconnection Networks", *IEEE Transaction on Computers*, Vol. c.34, No. 10, October 1985, pp.943-948
- [Pomb 88] A. Pombortsis, C. Halatsis, "Performance of Crossbar Interconnection Networks in Presence of 'Hot Spot'", *Electronics Letters*, Vol.24, No. 3, 4th February 1988, pp.182-184
- [Ragh 84] C.S. Raghavendra, A. Varma, "INDRA : A Class of Interconnection Networks with Redundant Paths", *1984 Real Time Systems Symp.*, Computer Society Press, Silver Spring, Md., 1984, pp.153-164
- [Ragh 87] .S. Raghavendra, A. Varma, "Reliability and Fault-Tolerance in Multistage Interconnection Networks", *IBM Research Report*, RC12528, 1987
- [Reed 87a] D.A. Reed, D.C. Grunwald, "The Performance of Multicomputer Multistage Interconnection Networks", *IEEE Computer*, June 1987, pp.63-73
- [Reed 87b] D.A. Reed, R.M. Fujimoto, *Multicomputer Networks : Message-Based Paralle Processing*, The MIT Press, 1987
- [Redd 84] S.M. Reddy, V.P.Kumar, "On Fault-Tolerant Multistage Interconnection Networks", *Proceedings of the Internatinal CONference on Parallel Processing*, August 1984, pp. 155-164
- [Sieg 79] H.J. Siegel, "Interconnection Networks for SIMD Machines", *IEEE Computer*, Vol. 12, No. 6, June 1979, pp.57-66
- [Szym 89] T. Szymanski, S. Shaikh, "Markov Chain Analysis of Packet-Switched Banyans with Arbitrary Switch Sizes, Queue Sizes, Link Multiplicities and Speedups", *1989 IEEE INFOCOM*, pp.960-971
- [Tami 88] Y. Tamir, G.L. Frazier, "High-Performance Multi-Queue Buffers for VLSI Communication Switches", *Proceedings of the 15th Annual International Symposium on Computer Architecture*, Honolulu, May 1988

- [Tami 89] Y. Tamir, G.L. Frazier, "Support for High-Priority Traffic in VLSI Communication Switches" To be submitted
- [Tant 88] A.N. Tantawi, "Performance Modeling and Analysis of a Hierarchically Interconnected Multiprocessor", *IBM Research Report*, RC 13335, March 1988
- [Turn 86] J.S. Turner, "Design of an Integrated Services Packet Network". *IEEE J. Select. Areas of Communication*, vol. SAC-4, no. 5, Nov. 1986, pp.1373-1380
- [Tzeng 89] N. Tzeng, "Design of a Novel Combining Structure for Shared-Memory Multiprocessors", *1989 International Conference on Parallel Processing*, 11-18
- [Varm 85] A. Varma, C.S. Raghavendra, "Realizations of Permutations on generalized Indra Networks", *Proceedings of the International Conference on Parallel Processing*, August, 1985, pp.328-333
- [Varm 88] A. Varma, B.D. Rathi, "A Fault-Tolerant Routing Scheme for Unique-Path Multistage Interconnection Networks", *IBM Research Report*, RC13441, 1988
- [Will 90] D.L. Willick, D.L. Eager, "An Analytical Model of Multistage Interconnection Networks", *ACM SIGMETRICS 1990*, pp. 192-199
- [Yen 82] D. W. Yen, J.H. Patel, E.S. Davidson, "Memory Interference in Synchronous Multiprocessor Systems", *IEEE Transaction on Computers*, Vol. c-31, No. 11, November 1982, pp.1116-1121
- [Yoon 90] H. Yoon, K.Y. Lee, M.T. Liu, "Performance Analysis of Multi-buffered Packet-Switching Networks in Multiprocessor Systems". *IEEE Transaction on Computers*, Vol. 39, No. 3, March 1990, pp.319-327