National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

## NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.



## AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.



Canada

# MULTICAST PACKET SWITCHING

## — Scheduling, Contention Resolution, Modeling, and Architecture

*Xing Chen*

A Thesis

in

The Department

of

Electrical & Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy at
Concordia University
Montréal, Québec, Canada

May 1992

ISBN 0-315-80973-6

# ABSTRACT

Multicast Packet Switching
— Scheduling, Contention Resolution, Modeling, and Architecture

Xing Chen, Ph.D.
Concordia University, 1992

This dissertation offers a discussion of various aspects of multicast packet switching. It emphasizes the issues of call scheduling disciplines, contention resolution algorithms, mathematical analysis and switch architecture. We review three call scheduling disciplines, one-shot, SS call splitting, WS call splitting, and propose a novel scheme — revision scheduling, which would give an indication of the best delay-throughput performance under input port queueing. We discuss the contention resolution algorithms by first introducing some well known schemes then proposing two novel algorithms, a cyclic priority access scheme with its combinational logic implementation and a neural network based algorithm. The former features high speed operation, and is compact, reliable and growable. The latter provides higher throughput and lower delay and demonstrates a potential of optimal scheduling. We present some analytic tools of traffic theory for the multicast switching system. We discuss the mathematical analysis of both the random selection policy and the cyclic-priority input access scheme for one shot discipline. Then, we introduce a general unified mathematical model for both one-shot and WS call splitting input access disciplines, by using the matrix-geometric technique. We summarize some proposed multicast packet switches, then propose a shared buffer memory switch structure with maximum queue and minimum allocation, where a shared buffer pool and a reserved buffer pool are allocated for switching and buffering the packets. We estimate the size of buffer memory required for certain packet loss probability under both balanced and unbalanced traffic pattern. The proposed switch accommodates multicasting and priorities and has the modular growth capability.

# ACKNOWLEDGEMENTS

I would like to express my thanks to my supervisor, Professor Jeremiah F. Hayes, for his invaluable guidance, encouragement and support throughout the course of this work. He gave me the opportunity and introduced me to various aspects of the telecommunication networks. He contributed a great deal of his time, effort and ideas to the work presented in this dissertation. I consider myself fortunate to have been working in association with him.

I would like to thank Canadian International Development Agency (CIDA) for the support in carrying out my studies under the joint doctoral program between Concordia University and Southeast University (China). I would like to thank Professors Shixin Cheng and Yongbin Chen of Southeast University for reading the dissertation and making valuable comments.

I would also like to thank Professor M. K. Mehmet-Ali and Dr. I. Lambadaris for very helpful discussions and suggestions. I would like to acknowledge the examiners for their valuable comments. Many thanks are extended to all my friends for their friendship and good time. I appreciate the financial assistance in part from Centre de Recherche en Informatique de Montréal (CRIM) through a graduate bursary. Finally, I want to thank my wife Guang for her encouragement, patience and care over many weekends and week nights I spent working away at school, and my lovely daughter Wenjia for her enduring my absence and preoccupation during the period of this work.

Dedicated to

My parents *Shuhua, Fugeng,*
My wife *Guang,*
and My daughter *Wenjia*

# TABLE OF CONTENTS

**LIST OF FIGURES** ..... x

**LIST OF TABLES** ..... xiv

**1. INTRODUCTION** ..... 1

1.1. Background ..... 2

1.2. Multicast Packet Switching ..... 3

1.2.1. Addressing of a Multicast Packet ..... 3

1.2.2. Multicast Packet Switch Fundamentals ..... 4

1.2.3. Definition of Throughput ..... 6

1.3. Interesting Issues ..... 7

1.4. Outline ..... 8

References 1 ..... 10

**2. MULTICASTING SCHEDULING** ..... 11

2.1. Introduction ..... 12

2.2. Scheduling and its Implementation Considerations ..... 14

2.2.1. One-shot Scheduling ..... 14

2.2.2. SS Call Splitting ..... 15

2.2.3. WS Call Splitting ..... 18

2.2.4. Revision Scheduling — A Novel Scheme ..... 20

2.3. Performance Comparison by Simulation ..... 22

2.4. Conclusion ..... 24

Appendix 2: Simulation Model ..... 25

References 2 ..... 28

**3. CONTENTION RESOLUTION ALGORITHMS** ..... 29

3.1. Review of Three Algorithms ..... 30

3.2. Cyclic Priority Access Scheme with its Combinational Logic Implementation ..... 34

3.3. Neural Network Based Algorithm ..... 44

3.3.1. Preliminary of Hopfield Model of Neural Networks ..... 46

3.3.2. A Model for One-Shot Scheduling ..... 49

3.3.3. An Extension Model — Windowed service ..... 5[illegible]

3.3.4. A Model for SS Call Splitting ..... 58

3.4. Conclusion ..... 62

References 3 ..... 65

**4. MODELING AND MATHEMATICAL ANALYSIS** ..... 66

4.1. Introduction ..... 67

4.2. Analysis of One-Shot Discipline with Random Selection ..... 69

4.2.1. Mathematical Model ..... 69

4.2.2. Numerical Results ..... 74

4.3. Analysis of One-Shot Discipline with Cyclic Priority ..... 77

4.3.1. Mathematical Model ..... 77

4.3.2. Numerical Results ..... 83

4.4. Review of Two Models for WS Call Splitting ..... 88

4.4.1. Model I by J. F. Hayes et al. ..... 88

4.4.2. Model II by J. Y. Hui et al. ..... 90

4.5. A General Unified Model for Performance Analysis ..... 91

4.5.1. Description of the System ..... 92

4.5.1.A). $Q$ Matrix for One-Shot Discipline ..... 95

4.5.1.B). $Q$ Matrix for WS Call Splitting Discipline ..... 95

4.5.2. Solution via Matrix Geometric Approach ..... 98

4.6. Remarks ..... 100

Appendix 4A: Algorithm for the Formation of the $Q$ Matrix (One-Shot Scheduling) ..... 103

Appendix 4B: Algorithm for the Formation of the $Q$ Matrix (WS Call Splitting) ..... 104

References 4 ..... 105

**5. SWITCH ARCHITECTURE** ..... 106

5.1. Introduction ..... 107

5.2. A Survey of Multicast Packet Switches ..... 107

5.2.1. Banyan-Based Space-Division Switch ..... 107

5.2.2. Knockout Switch ..... 115

5.2.3. Shift Switch ..... 123

5.2.4. Shared Buffer Memory Switch ..... 127

5.3. A Shared Buffer Memory Switch with Maximum Queue and Minimum Allocation ..... 130

5.3.1. Classifications of Shared Memory Switches ..... 131

5.3.2. The Switch Architecture ..... 132

5.3.3. Estimation of Packet Loss Probability ..... 137

5.3.3.1 Balanced Traffic ..... 140

5.3.3.2 Unbalanced Hot-Spot Traffic ..... 143

5.3.4. The Capabilities of Multicasting, Modularity and Priority ..... 144

5.4. Conclusion ..... 148

References 5 ..... 150

**6. CONCLUSIONS AND FUTURE RESEARCH** ..... 153

6.1 Conclusions ..... 154

6.2 Future Research ..... 155

# LIST OF FIGURES

Figure 1.1. ATM packet format ..... 5

Figure 1.2. A multicast switch ..... 6

Figure 1.3. An example of multicast switch ..... 6

Figure 2.1. Concept of one-shot scheduling ..... 15

Figure 2.2. A switch fabric for SS call splitting scheduling ..... 16

Figure 2.3. Multicasting operation ..... 17

Figure 2.4. Concept of WS call splitting scheduling ..... 18

Figure 2.5. Cross-bar structure with WS call splitting scheduling ..... 19

Figure 2.6. Lee's multicast switch ..... 20

Figure 2.7. Comparison of the delay-throughput performance ..... 23

Figure A2.1. A flowchart of the program SIMU ..... 26

Figure 3.1. A re-entry network for contention resolution ..... 31

Figure 3.2. Three-phase algorithm ..... 32

Figure 3.3. Ring reservation-based contention resolution ..... 33

Figure 3.4. A basic decision element ..... 35

Figure 3.5. Cyclic priority assignment ..... 36

Figure 3.6. A 4×4 decision board for WS call splitting discipline ..... 37

Figure 3.7. A basic element with *HFS* ..... 38

Figure 3.8. A 4×4 decision board for SS call splitting discipline ..... 39

Figure 3.9. A basic element with *fHFS* and *bHFS* ..... 40

Figure 3.10. A 4×4 decision board for one-shot scheduling ..... 41

Figure 3.11. A basic element for revision scheduling ..... 42

Figure 3.12. An example of the longest signal routes ..... 43

Figure 3.13. A Hopfield model of neural network ..... 47

Figure 3.14. A comparison of cyclic priority selection and neural network selection ..... 55

Figure 3.15. Input port queueing with windowed service ..... 57

Figure 3.16. Improvement of windowed service (window size $W$=2) ..... 59

Figure 3.17. A two-dimensional structure of Hopfield neural network ..... 60

Figure 3.18. Improvement by neural network for SS call splitting ..... 63

Figure 4.1. Average number of copies of the HOL packet vs. packet arrival rate ..... 68

Figure 4.2. Multistage screening process ..... 70

Figure 4.3. Average packet delay for several switch sizes with constant $\overline{m}$ (2 and 3) ..... 75

Figure 4.4. Average packet delay with one-shot scheduling and random-selection policy ..... 76

Figure 4.5. Average packet delay with one-shot scheduling and cyclic priority policy where each incoming packet generates copies according to Bernoulli trials with parameter $p$ for each output port ..... 85

Figure 4.6. Average throughput with one-shot scheduling and cyclic priority policy where each incoming packet generates copies according to Bernoulli trials with parameter $p$ for each output port ..... 86

Figure 4.7. Average packet delay with one-shot scheduling and cyclic priority policy where each incoming packet generates $NC$ (a constant number of) copies to a set of output ports ..... 87

Figure 4.8. Transition of the states ..... 93

Figure 4.9. Configuration of a block matrix ..... 94

Figure 4.10. Transition of the states within block matrices for one-shot scheduling ..... 96

Figure 4.11. Transition of the states within block matrices for WS call splitting ..... 97

Figure 4.12. Numerical results with binomial copy distribution ..... 101

Figure 4.13. Numerical results with deterministic copy distribution (constant fanout of the packet) ..... 102

Figure 5.1. A 8×8 Batcher-banyan nonblocking routing network ..... 109

Figure 5.2. Copy network of Starlite switch ..... 110

Figure 5.3. Copy network of Turner's broadcast packet switch ..... 111

Figure 5.4. A block diagram of Lee's copy network ..... 112

Figure 5.5. Multicast switch with centralized controller ..... 113

Figure 5.6. A modular structure of a Batcher-banyan multicast switch ..... 114

Figure 5.7. Knockout switch ..... 115

Figure 5.8. A 8:4 concentrator ..... 116

Figure 5.9. Modular growth of knockout switch ..... 117

Figure 5.10. A knockout switch with multicast modules ..... 118

Figure 5.11. Multicast operation with packet duplicator approach ..... 119

Figure 5.12. Multicast operation with fast packet filter approach ..... 120

Figure 5.13. Bus interface with sorter-type concentrator ..... 122

Figure 5.14. An alternative of modular structure of knockout switch ..... 123

Figure 5.15. A shift switch ..... 124

Figure 5.16. Multicast operation in a shift switch ..... 126

Figure 5.17. Kuwahara's shared buffer memory switch ..... 127

Figure 5.18. Output queue chain in shared buffer memory switch ..... 128

Figure 5.19. Multicast operation in shared buffer memory switch ..... 129

Figure 5.20. Concept of a memory-type switch ..... 130

Figure 5.21. A shared buffer memory switch ..... 133

Figure 5.22. Memory allocation in proposed switch ..... 134

Figure 5.23. FIFO address queue ..... 135

Figure 5.24. Memory resource used by an output queue ..... 138

Figure 5.25. Buffer requirement vs. $B_2/N$ ..... 140

Figure 5.26. Packet loss probability vs. buffer size for sharing with maximum queue length ..... 141

Figure 5.27. Packet loss probability vs. buffer size for sharing with maximum queue and minimum allocation ..... 142

Figure 5.28. Packet loss probability vs. buffer size for sharing with maximum queue under unbalanced traffic ..... 144

Figure 5.29. Packet loss probability vs. buffer size for sharing with maximum queue and minimum allocation under unbalanced traffic ..... 145

Figure 5.30. Multicast operation in proposed switch ..... 146

Figure 5.31. Modular structure of a 1024×1024 shared buffer memory switch ..... 147

# LIST OF TABLES

Table A2.1. Options of SIMU ..... 25

Table 3.1. Truth table of a basic element ..... 35

Table 3.2. Truth table for the basic element in Figure 3.7 ..... 38

Table 3.3. Truth table for the basic element in Figure 3.9 ..... 40

Table 3.4. Comparison in terms of the number of gates ..... 44

Table 3.5. Realizable switch size ..... 44

Table 3.6. Saturated throughput of $N \times N$ unicast switch ..... 56

# CHAPTER 1

# INTRODUCTION

1.1. BACKGROUND

1.2. MULTICAST PACKET SWITCHING

1.2.1. Addressing of a Multicast Packet

1.2.2. Multicast Packet Switch Fundamentals

1.2.3. Definition of Throughput

1.3. INTERESTING ISSUES

1.4. OUTLINE

REFERENCES 1

## 1.1. BACKGROUND

The information age had been characterized as a time of exploding demand for communications applications of all kinds. These new demands create both great opportunities and difficult challenges for suppliers of telecommunications services and equipment. Most attention currently centers on Integrated Services Digital Network (ISDN). It is in the environment of changing user needs, intelligent digital communication technologies, and declining costs that many of the world's telecommunications networks have begun to deploy ISDN.

Work on ISDN, or narrowband ISDN, has resulted in a standard for a digital line interface, comprising two 64Kb/s circuit switched channels for voice and bulk data and a 16 Kb/s packet switched channel for data and control information. But, the only integration possible is at the level of physical packaging. It has become clear that these transfer modes can not efficientiy meet the requirements for faster enhanced communications or multi-media communications.

An information transport network of next generation should be able to provide connections in a wide range of bandwidth. Application, such as telemetry require just a few bits per second, while video may require 100Mb/s or more. An information transport network should be able to handle such extremes, as well as everything in between. New technology, most notably fiber optics, has brought the broadband ISDN (BISDN) closer to reality. For example, using single-mode fibers and single-wavelength lasers, data rate equal to 4Gb/s over 100 km without repeaters has been achieved. While fiber optics technology provides the necessary bandwidth for transmission purposes, the creation of a network that can provide high bandwidth services to the users remains a challenge. The bottleneck comes primarily from switching, which will carry all applications in an integrated fashion and offer the desired flexibility to handle the wide diversity of data rate and latency requirement resulting from the integration of services.

The Asynchronous Transfer Mode (ATM) was proposed for BISDN to meet these requirements by combining the high-speed capability of the circuit mode and the flexibility of the packet mode. ATM specifies fixed-size packets comprising 48 bytes of data and 5 bytes of control information each. Many line speeds are also specified, with nominal rates equal to 155.52Mb/s

(required for digitized HDTV), and 622.08Mb/s. The various architectures for such fast packet switches have been reported and were surveyed in [1].

In BISDN, the services will be very diversified. Such diversity of services demands not only point-to-point communications but also multipoint communications. Therefore, ATM switching networks, the essential component in the network, should be able to provide the multicast function. The term multicast would appear to be derived from the term broadcast and means concurrent data transmission to a restricted set of recipients. We shall use the term unicast to indicate the point-to-point connection.

The idea of multicast first appeared in the early 1980s. It was expected to make point-to-multipoint communication less complex, lead to greater concurrency and decrease network load. This would benefit a wide range of applications: resource location in the presence of multiple servers; updates of, and information retrieval from, replicated databases; replicated procedure cells; majority voting systems; distributed parallel computation; routing table dissemination; distributed games; commercial television distribution; voice or video teleconferencing; and clock synchronization. So far, multicast has still not become a standard facility, but is very much alive as a research topic.

## 1.2. MULTICAST PACKET SWITCHING

Multipoint connections could be accomplished by creating multiple copies of the packet at the source node, each destined to one of the desired destinations, and routing the copies independently. Alternatively, multicast routing may be achieved by requiring the switches in the network to have the capability to replicate a packet at several of their output ports, according to information provided for that purpose, thus reaching the multiple destinations from a single packet originating at the source. This mode of operation results in lower traffic throughout the network and gives birth to the ATM multicast packet switch design.

### 1.2.1. Addressing of a Multicast Packet

Addressing is one of the key functionality for multicasting communication. A multicast address is the most general form of an address. Common special cases are unicast and broadcast addresses. Whereas the latter may be realized comparably easily in almost all types of networks, multicast addressing turns out to be quite tricky.

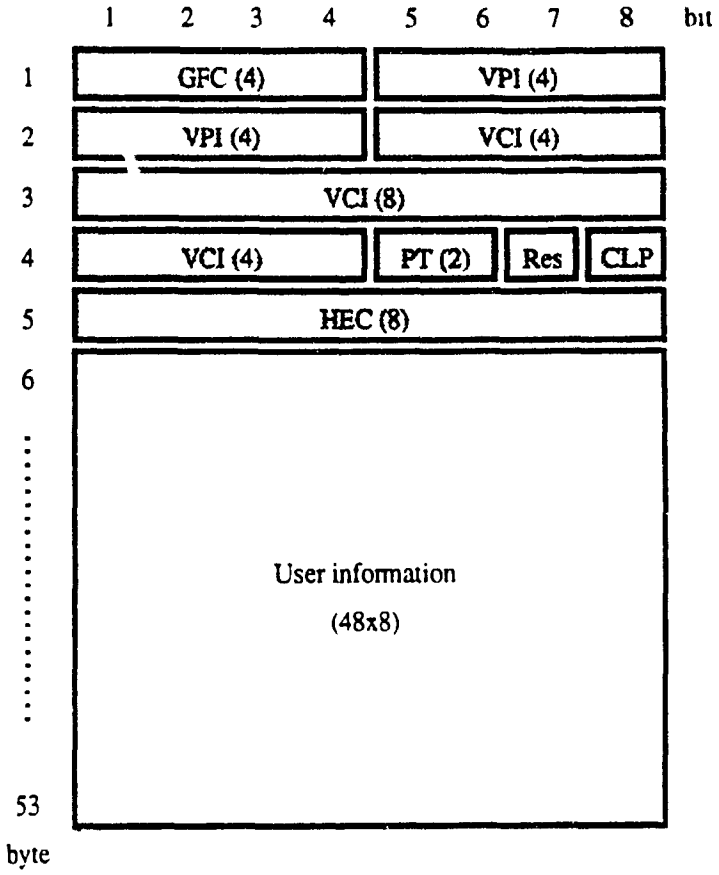
In ATM, all information is conveyed through the network in fixed-size packets. Figure 1.1 shows the ATM packet format, as currently conceived by CCITT 1990 Recommendation I.361 [2][3]. When a packet arrives at the input of the switch node, a destination address relevant to local routing inside the switch is derived, based on its header information (mainly VCI). For a multicast packet, it is impracticable to carry all the addresses of the destinations to which the packet is sent, resulting in too much overhead on the multicast packet. An easier way would be that multicast packet only carries a multicast group address (or a multicast channel number) to indicate the subset of destinations. A simple table look-up is usually used to decode the addresses in the multicasting operation. Multicast group addresses need to be integrated into present addressing standards, and be assigned, maintained and managed.

### 1.2.2. Multicast Packet Switch Fundamentals

The function of a multicast packet switch is to transmit an arriving multicast packet to a set of destinations. Furthermore, it must be flexible enough to support unicast and broadcast connections as special cases. Like unicast switches, the $N \times N$ multicast switch† architecture has $N$ input ports (or input lines, or input links) where the traffic arrives and $N$ output ports (or output lines, or output links) where the traffic leaves as shown in Figure 1.2.

The multicasting operation usually comprises two sequential steps: the arriving packet is replicated and the copies of the packet are routed to a set of output ports (Figure 1.3), while in some cases, these two steps are completed simultaneously. In this dissertation, we address the arriving traffic as the packet, and the leaving traffic as the copy without exception even for uni-

† It is not necessary that the number of input ports equals to the number of output ports. The nonsquare $NI \times NO$ switch, with larger fan-out than fan-in ($NO \geq NI$), would be appropriate in some practical applications.




| | |
|---|---|
| VCI | Virtual Channel Identifier |
| VPI | Virtual Path Identifier |
| CLP | Cell Loss Priority |
| GFC | Generic Flow Control |
| PT | Payload Type |
| HEC | Header Error Control |
| Res | Reserved |

Figure 1.1. ATM packet format

cast connections.

In a multicast packet switch, there are two reasons for buffering. One is that the required total bandwidth exceeds the capacity of the switch, that is the total number of destination requests exceeds the number of output ports. The other is that the required bandwidth for a particular output line exceeds the capacity of the single output line, that is multiple packets request

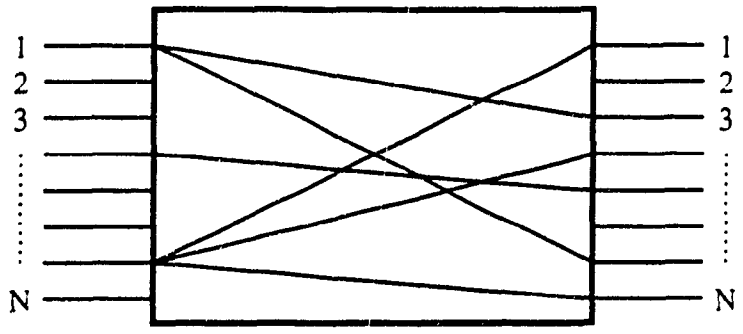



Figure 1.2. A multicast switch

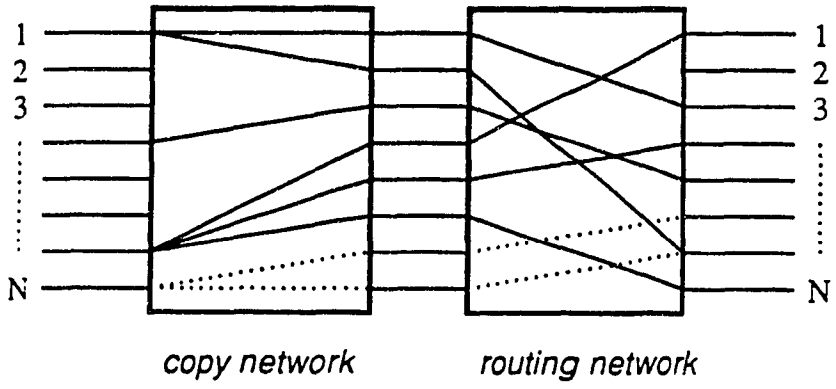



Figure 1.3. An example of multicast switch

the same output port simultaneously. Buffering or queueing provides the means to regulate the traffic and prevent the packet loss.

### 1.2.3. Definition of Throughput

In unicast switching, the throughput is defined as the average number of packets being transmitted through the switch node per input/output per time slot, or as the utility of each output line. The saturated throughput is defined as the throughput when all the input queues are 100% full, or as the throughput when packet delay goes up rapidly.

As to the multicast switch, we have two possible definitions. One is the average number of

copies being transmitted to output ports per output line per time slot, namely copy's throughput Since the generated copies are for different output ports, the saturated copy's throughput is greater than that of the unicast switch. Another definition is the average number of packets being taken from input port queues per input line per time slot, namely packet's throughput, which is much lower than that of the unicast switch because each packet may generate several copies.

In this dissertation, we shall only use the latter, packet's throughput, simply called throughput. Because the most interesting value of throughput is the saturated throughput, rather than the evolution of the throughput with the change of packet arrival rate, we shall evaluate the switch by showing only the delay-throughput performance, from which the saturated throughput can be read explicitly.

## 1.3. INTERESTING ISSUES

Multicasting does not only bring promising services, but also raises new research topics which do not arise in unicast switches. The output port conflicts when multiple packets request the same output port concurrently is present in both unicast and multicast cases. Therefore, queueing is required in order to avoid packet loss, in any type of switch structure. In most cases, the queue is located either at the input or at the output of the switch.

If we assume that the switch fabric runs at the same speed as the input and output lines, only one packet can be accepted by any given output line during a time slot, and other packets addressed to the same output must queue on the input lines. Under this assumption, it has been shown that a unicast switch has a maximum throughput of 58.6% of switch capacity, regardless of the contention resolution mechanism [4][5][6]. However, in the multicast switch, we may consider several different service disciplines in terms of scheduling the transmission of the copies of the packet. Such a topic of call scheduling does not arise in unicast switches. We will see that different call scheduling discipline leads to different delay-throughput performance. Furthermore, the delay-throughput performance can be improved by introducing an optimal contention resolution algorithm. Also, the exact mathematical analysis becomes more difficult because of header-of-line (HOL) destination coupling and lack of a model to describe the copy distribution of the

HOL packet.

In contrast, if the $N \times N$ switch fabric runs $N$ times as fast as the inputs and outputs or has $N$ parallel paths so that any incoming packet is able to pass through the switch within a time slot, then an output queueing is required instead of input queueing. In this context, the existing traffic theory for unicast switching can be applied to multicast switching just by considering multicasting traffic arrival process. Issues such as call scheduling, contention resolution are not present in output port queueing scheme. Therefore, our interest tends to packet loss probability due to the finite buffer resources for output queueing.

As far as the switch architecture is concerned, in point-to-point communication networks, a switching system is able to connect any incoming channel to any outgoing channel. To support multipoint connections, a switching system must be able to connect any incoming channel to any subset of its outgoing channels. Therefore, an extra network is needed, such as copy network in a banyan-based space-division multicast packet switch [7], multicast module and multicast bus in knockout switch [8] and multicasting controller in shared buffer memory switch [9].

## 1.4. OUTLINE

The intent of this dissertation is to offer a discussion of various aspects of multicast packet switching. The next three chapters emphasize the issues related to the multicast packet switch with input port queueing: call scheduling disciplines, contention resolution algorithms, and mathematical analysis. Then, we spend one chapter to discuss the multicast packet switch architecture both with input queueing and output queueing.

Chapter 2 reviews three call scheduling disciplines, one-shot, SS call splitting and WS call splitting, and proposes a novel scheme — revision scheduling. By comparison, revision scheduling is most preferred as far as the delay-throughput performance is concerned. Taking into account complexity of implementation as well as performance, each of the four categories does not seem to have distinctive superiority over the others. There is a trade-off between the performance and switch structure complexity.

Chapter 3 discusses contention resolution algorithms by first introducing some well known schemes. After that, we propose two novel algorithms, a cyclic priority access scheme with its combinational logic implementation and a neural network based algorithm. The combinational logic scheme features simple and high speed operation, in comparison with the other methods such as the three-phase algorithm [4] and the ring-reservation algorithm [6]. It is compact, reliable and growable. The neural network based algorithm provides higher throughput and lower delay and demonstrates a potential optimal scheduling.

Chapter 4 presents some analytic tools of traffic theory for the input port queue of the multicast switching system. In this chapter, we discuss the mathematical analysis of both the random selection policy and the cyclic-priority input access scheme for one-shot discipline. Then, we summarize two analytic models for WS call splitting discipline by Hayes and Hui, respectively. Finally, we introduce a general unified mathematical model for both the one-shot and the WS call splitting input access disciplines, by using matrix-geometric techniques. These results could serve to model the multicast packet switch and to predict the onset of congestion in the system. Although this chapter is devoted to the multicast switch, the analytic model can also be applied to other queueing problems.

In Chapter 5, we first summarize some proposed multicast packet switches, including the banyan-based space-division switch, the knockout switch, the shift switch and the shared buffer memory switch. Then we propose a shared buffer memory switch structure with maximum queue and minimum allocation, where a shared buffer pool and a reserved buffer pool are handled for switching and buffering the packets. We estimate the size of buffer memory required for certain packet loss probability under both balanced and unbalanced traffic pattern. The proposed switch accommodates multicasting and priorities and has the modular growth capability.

Finally, concluding remarks and directions for future research are given in Chapter 6.

## REFERENCES 1

[1] F. A. Tobagi, "Fast packet switch architecture for broadband Integrated Services Digital Networks", Proceedings of the IEEE, Vol.78, No.1, pp.133-167, Jan. 1990.

[2] H. Ohnishi, T. Takahashi, K. Kuroda, T. Okada, "Switching technologies for B-ISDN", NTT Review, Vol.3, No.3, pp.59-70, May 1991.

[3] CCITT, "Thirteen B-ISDN recommendations approved by SG XVIII at Matsuyama", Temporary Documents 1 and 9 (XVIII), Nov. 1990.

[4] J. Y. Hui, E. Arthurs, "A broadband packet switch for integrated transport", IEEE J. Select. Areas Commun., Vol.5, No.8, pp.1264-1273, Oct. 1987.

[5] M. J. Karol, M. G. Hluchyj, S. P. Morgan, "Input versus output queueing on a space-division packet switch", IEEE Trans. on Comm., Vol.35, No.12, pp.1347-1356, Dec. 1987.

[6] B. Bingham, H. Bussey, "Reservation-based contention resolution mechanism for Batcher-Banyan packet switches", Electronics Letters, 23rd, Vol.24, No.13, pp.772-773, June 1988.

[7] T. T. Lee, "Nonblocking copy networks for multicast packet switching", IEEE J. Select. Areas Commun., Vol.6, No.9, pp.1455-1467, Dec. 1988.

[8] K. Y. Eng, M. G. Hluchyj, Y. S. Yeh, "Multicast and broadcast services in a knockout packet switch", Proc. IEEE Infocom'88, pp.29-34, New Orleans, LA., March 1988.

[9] H. Kuwahara, et al., "A shared buffer memory switch for an ATM exchange", Proc. ICC'89, pp.118-122, Boston, June 1989.

# CHAPTER 2

# MULTICASTING SCHEDULING

2.1. INTRODUCTION

2.2. SCHEDULING AND ITS IMPLEMENTATION CONSIDERATIONS

2.2.1. One-shot Scheduling

2.2.2. SS Call Splitting

2.2.3. WS Call Splitting

2.2.4. Revision Scheduling — A Novel Scheme

2.3. PERFORMANCE COMPARISON BY SIMULATION

2.4. CONCLUSION

APPENDIX 2: Simulation Model

REFERENCES 2

## 2.1. INTRODUCTION

We assume that the switch fabric runs at the same speed as the input and output lines, only one packet can be forwarded to any given output line during a time slot, and other packets addressed to the same output must queue at the input ports. Blocking is present in unicast switches and is compounded in multicast switches, i.e. two or more input ports seek to send copies to the same output port in the same time slot. Unlike the unicast switch with saturated throughput 58.6% due to HOL blocking, the multicast switch does not have such a limit, in fact, the throughput of the multicast switch is the function of copy distribution of the packet as well as the traffic arrival rate. Not only that, under the assumption of input port queueing, we may consider the switch performance for several different service disciplines in terms of scheduling the transmission of the copies of the packet in case of output contention at the input ports.

A number of service disciplines were proposed and summarized in [1]. Of these the most interesting are the following three. One-shot scheduling requires all the copies of the same packet to be transmitted in the same time slot. Strict-sense (SS) call splitting and wide-sense (WS) call splitting allow the transmission of the packet to be split over several time slots. SS specifies that each packet can send at most one copy to the destination per time slot; WS does not carry this restriction. In this chapter, a novel scheme, revision scheduling, is proposed to mitigate the HOL blocking effect by sequentially combining the one-shot scheduling and WS call splitting discipline, which may give an indication of the best performance under the assumption of input port queueing.

For simplicity, we may use a matrix to describe the call scheduling problem. Suppose there is a matrix where each row and each column correspond to the input line and output line of the switch, respectively. The elements 1's stand for the copies of the HOL packets. In the transmission request matrix, $I_{ij} = 1$ denotes the HOL packet at input $i$ seeks to send a copy to output $j$. In the case of multicasting switching, each row may contain two or more 1's rather than only one 1 as in the unicast case. Since an output port may take only one packet in a time slot, each column has at most one 1 being selected. Of course, one of them must be selected if there is at least one

in the column. Obviously, there are a number of choices to fulfill this requirement. The criteria for the four considered disciplines are set up with respect to the rows of the matrix:

1) One-shot scheduling: the 1's at the same row must be all selected or all ignored simultaneously.

2) SS call splitting: at most one 1-element can be selected from each row.

3) WS call splitting: no restriction on rows.

4) Revision scheduling: a sequential combination of a one-shot scheduling and a WS call splitting discipline.

Following are some examples for these disciplines.

1 1 0 0 1
0 1 0 0 0
0 0 0 0
0 1 1 0 0
0 0 1 1 0

Transmission request matrix
(1: copy request)

ϕ ϕ 0 0 ϕ
0 1 0 0 0
0 0 0 ϕ 0
0 1 1 0 0
0 0 1 1 0
*One-shot*
(1)

ϕ 1 0 0 1
0 ϕ 0 0 0
0 0 0 ϕ 0
0 1 ϕ 0 0
0 0 1 1 0
*SS call splitting*
(2)

ϕ 1 0 0 ϕ
0 ϕ 0 0 0
0 0 0 ϕ 0
0 1 ϕ 0 0
0 0 1 1 0
*WS call splitting*
(3)

ϕ ϕ 0 0 ϕ
0 1 0 0 0
0 0 0 ϕ 0
0 1 1 0 0
0 0 ϕ 1 0
*Revision scheduling*
(4)

Call scheduling disciplines
(ϕ: accepted request 1: rejected request)

These scheduling schemes have their own advantages and disadvantages, based on delay-throughput performance and implementation consideration. This chapter introduces schematic

structures for each category of scheduling. The complexity of several implementations is addressed. The simulations were carried out for the comparison of performance.

The remainder of the chapter is organized as follows. Section 2.2 describes four call scheduling schemes and their implementation considerations. A performance comparison by simulation for these four schemes is presented in Section 2.3. Section 2.4 contains the conclusion of the chapter.

## 2.2. SCHEDULING AND ITS IMPLEMENTATION CONSIDERATIONS

Previous studies showed that the delay-throughput performance of the unicast switch does not depend on the way of selecting winning packet from the conflicting packets. However, in the case of multicast switching, our results show that a different call scheduling discipline does provide different performance. In particular, we will show that simple revision scheduling achieves the best performance among the all concerned. Here, we are going to define these scheduling disciplines.

### 2.2.1. One-shot Scheduling

In the case of the one-shot scheduling, all copies of the same packet must be switched in the same slot. If at least one copy loses the contention for an output port, the whole packet must wait in a queue and try again in the next slot. This strategy favors the packet with less copies. The packet with more copies will be blocked more often than that with less copies, because of simultaneous output contention.

The one-shot discipline can be implemented by a serial combination of an input buffer memory, a copy network [2] and a routing network, as depicted in Figure 2.1. A centralized controller is used to regulate the input traffic. Copy requests of the multicast packet will be considered before these packets enter the copy network. Any selected packet should pass through the switch within a single slot; accordingly, no buffer is needed between the copy network and routing network. Two implementation examples of this scheme are the cyclic-priority input access scheme and the neural network based scheme.

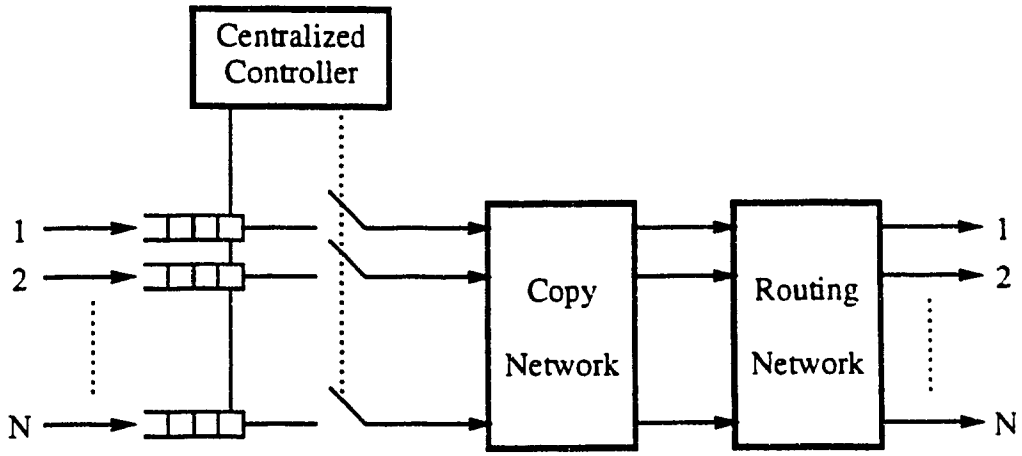



Figure 2.1. Concept of one-shot scheduling

The cyclic-priority input access scheme is a derivative of the token passing ring, and is a simple input access scheme to handle the contention resolution problem. Instead of token ring passing, the cyclic-priority input access scheme can be implemented by a simple combinational logic circuit which will be described in Section 3.2.

The neural network based contention resolution mechanism allows us to select, from all HOL packets in input queues, a particular set of packets as inputs of copy network, which ensures contention-free access to output ports and maximizes packet throughput. The detailed discussion will be seen in Section 3.3.2.

### 2.2.2. SS Call Splitting

A contrasting discipline under consideration is SS call splitting scheduling. In the case of the one-shot discipline, even if only one copy loses the contention, the whole packet (all its copies) is blocked. Obviously, this degrades the utilization of the output lines and suggests that we can transmit copies independently. But, each input can send at most one copy into the switch fabric per time slot, assuming that there is only one line connecting each input port and the switch fabric. The packet with residual copies continues to contend for output ports in the following slots until all of its copies have been transmitted. Obviously, at least $C$ slots are needed to

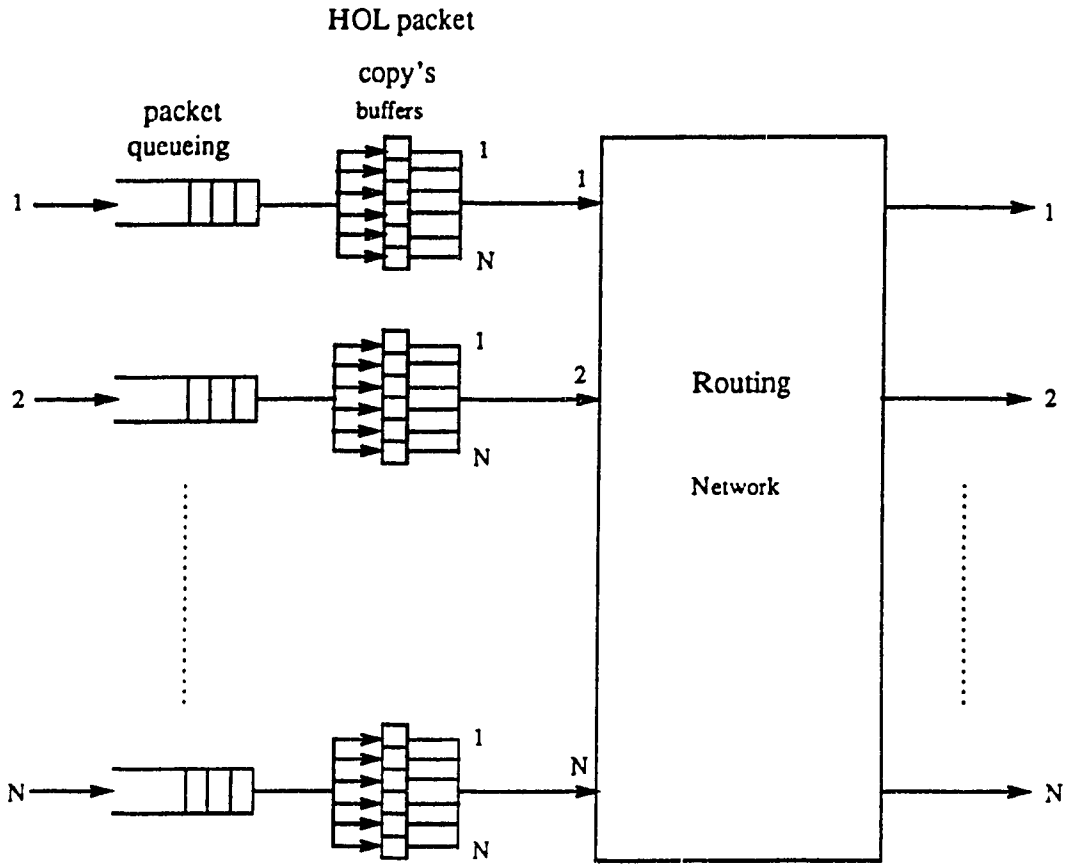



Figure 2.2. A switch fabric for SS call splitting scheduling

transmit a packet if it has $C$ copies.

As mentioned in Chapter 1, a multicast packet only carries a multicast channel number (MCN) to indicate the subset of destinations, rather than carry all individual destinations of the copies resulting in too much overhead on the multicast packet. A simple table look-up is usually used to decode the addresses in multicasting operation. Therefore, it is difficult to relabel the multicast packet with residual number of copies in order to prevent both the reduplication and ignorance of the destinations. One way of implementing this is that the copies of a packet are served in a prescribed order [1] so that the switch only sees one copy of a packet at a time and operates as in unicast fashion. The another implementation is to reproduce the HOL packet

before the splitting transmission so that the switch is able to see all the copies of the HOL packet simultaneously. This scheme is functionally shown in Figure 2.2. The multicasting operation is implemented in each input queue by shifting the HOL packet into a set of parallel buffers corresponding to its multicast destinations, as depicted in Figure 2.3. MCN is taken as the primary key of the multicasting table. Each cell of the table consists of $N$ bits to indicate the corresponding destinations by 1's. The destination addresses of the copies are automatically added to the messages in the individual buffers. A centralized controller to select contention-free copies will be discussed in Section 3.2. No special copy network is needed. When all the buffers are cleared, the next packet will be shifted in.

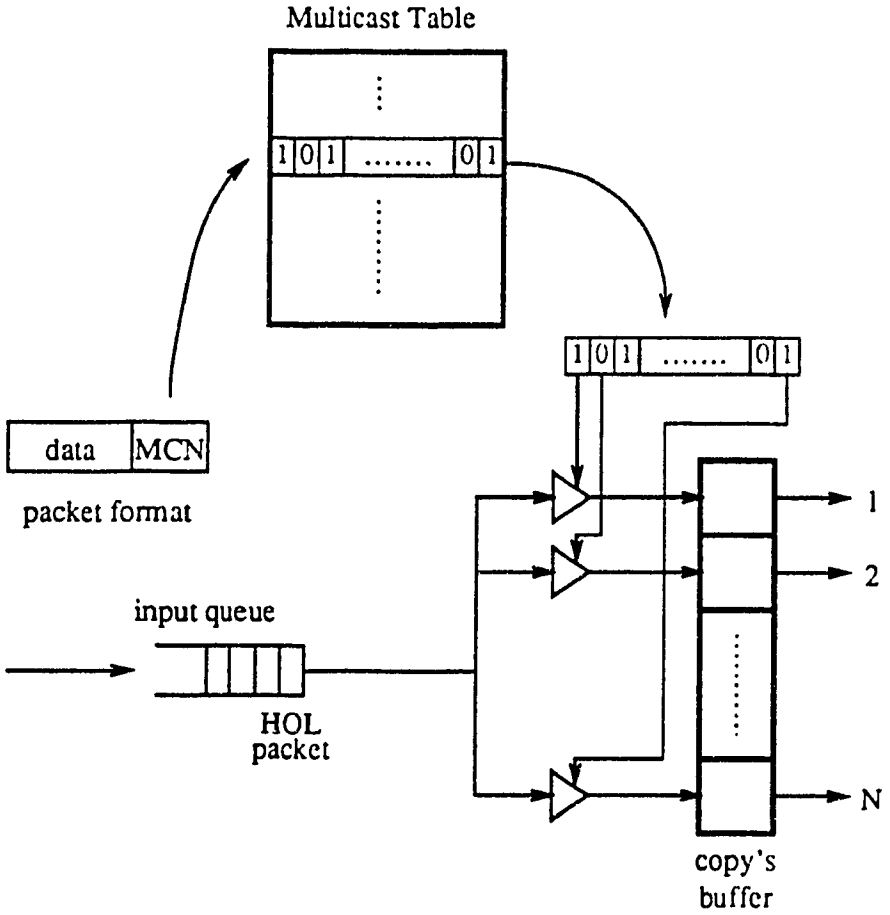



Figure 2.3. Multicasting operation

Again, an optimal selection policy is sought. We will use neural-network-based input access method to deal with this problem in Section 3.3.4.

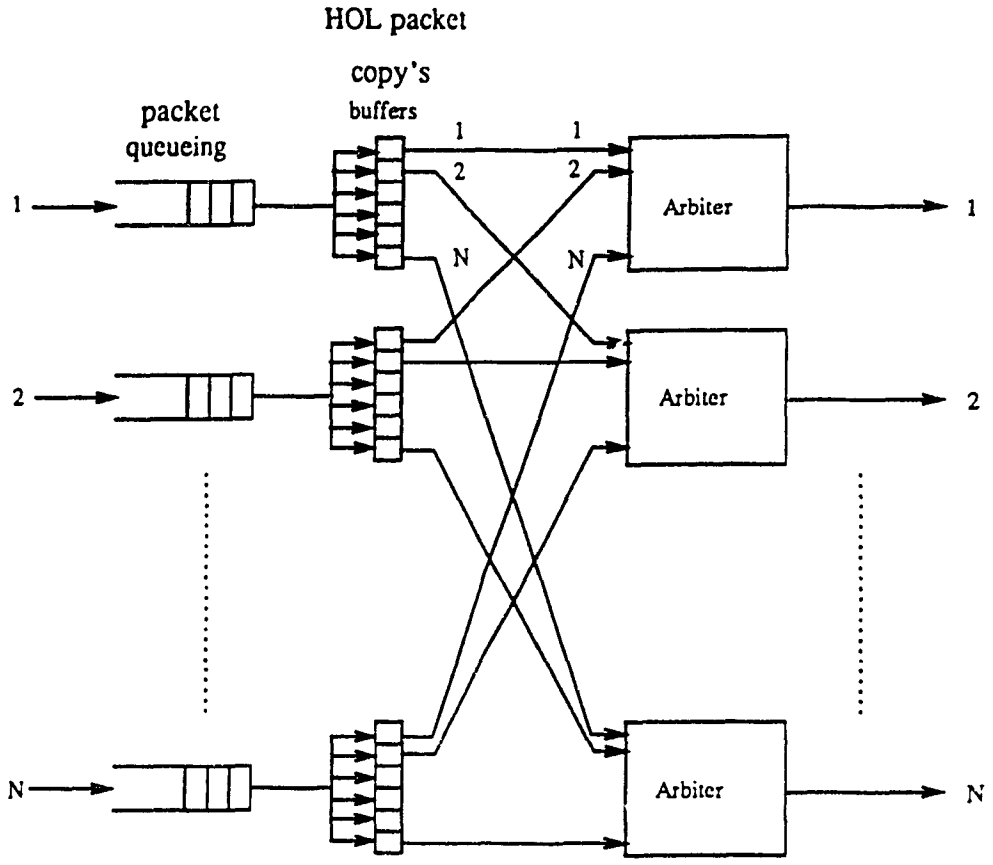



Figure 2.4. Concept of WS call splitting scheduling

### 2.2.3. WS Call Splitting

SS call splitting results in throughput degradation in light traffic case. Although there may be a case that only one active input line and all the output lines are free for that packet, only one copy can be accepted at a time, so that the utility of the output line is not sufficient. Thus, the third discipline we are interested in is WS call splitting. We allow more than one copy from the same packet to gain access to output ports simultaneously as long as those output ports are free.

Full use is made of the output lines since an output line is used as long as there is at least one copy in the HOLs destined for it.

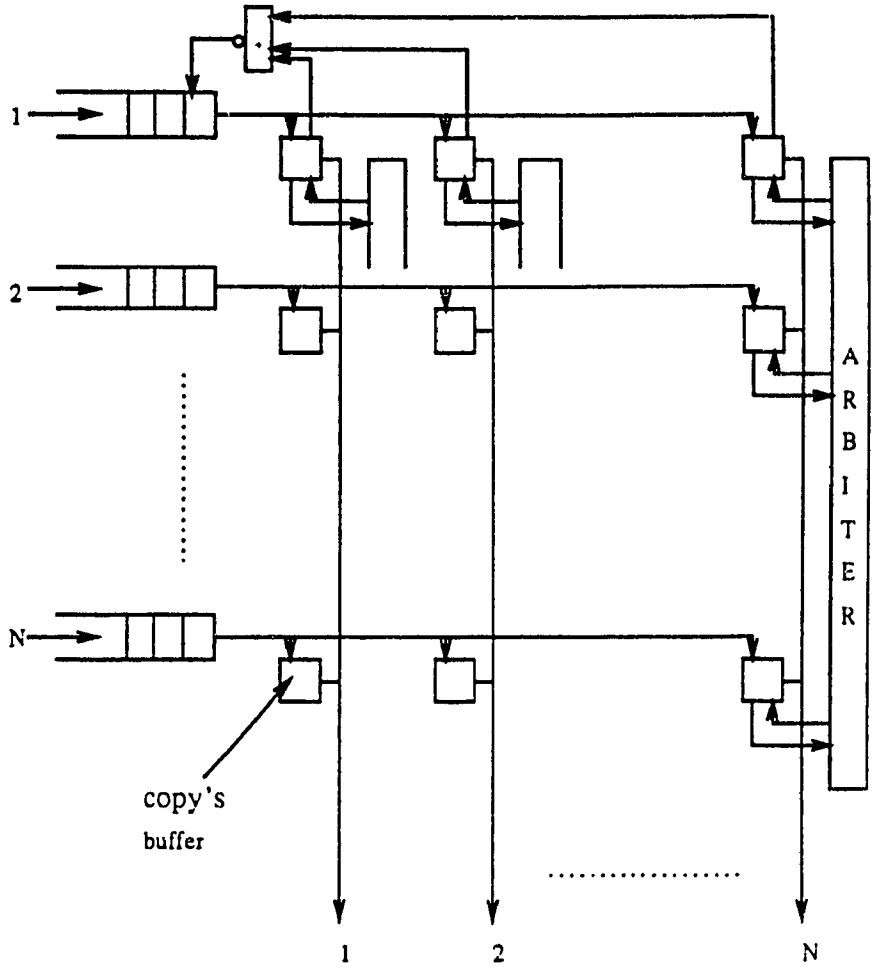



Figure 2.5. Cross-bar structure with WS call splitting scheduling

The concept of this discipline is illustrated in Figure 2.4. *N* arbiters are used in place of a routing network. Each such arbiter sees all copies that are proceeding to the corresponding output and selects one of them randomly (or according to some rule). This scheme can be viewed as a derivative of the crossbar switch with input buffering and centralized output arbiters, as depicted in Figure 2.5. *N* copy's buffers of the HOL packet at an input port are distributed over a row of *N*

crosspoints. When all the buffers in a row are cleared, a new packet is multicast along the bus connecting to $N$ copy's buffers. The transmission of the copies of the packet will be dispersed over several time slots.

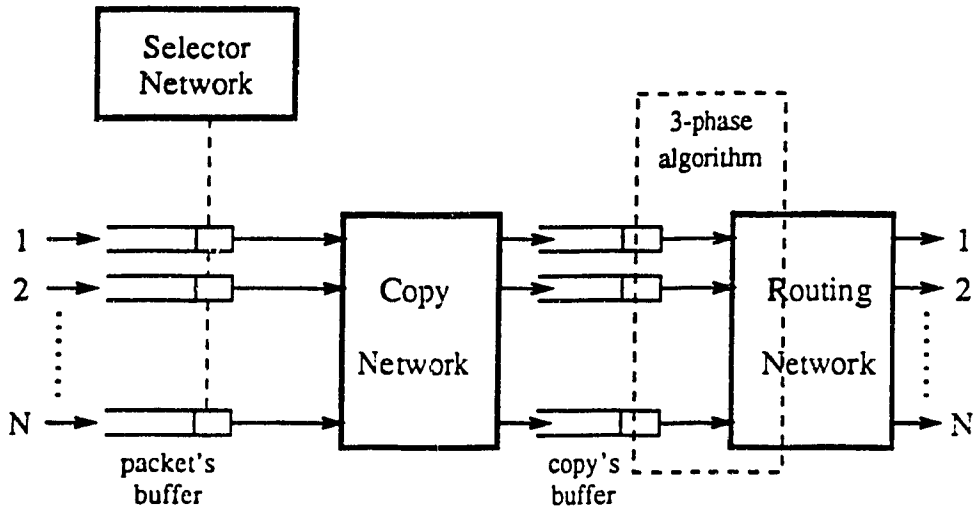



Figure 2.6. Lee's multicast switch

Another implementation was suggested by Lee [3] (see Figure 2.6), where the HOL packets are duplicated, and blocked copies are kept in buffers between copy network and routing network to prevent message loss. The overflows of the copy network are prevented by using selector network and the output contention is resolved by employing a three-phase algorithm [4]. In a sense, this is a true call splitting scheduling, since the copies queueing at the buffers have lost their association with the input packet. Hence, the FIFO principle for incoming packets has been lost as well.

### 2.2.4. Revision Scheduling — A Novel Scheme

As we know, output queueing performs better than input queueing although it may require more complicated designing. It is HOL blocking that causes throughput degradations in input queueing schemes. To mitigate this phenomenon, we try to get as many packets through as

possible so that the packets behind the HOL packets can join the contention in the next slot. For example, in the sense of WS call splitting, the following two choices illustrated below are equivalent. But, (b) is potentially better than (a) because three new packets in (b) will participate in the contention in the following slot, while there will be only one new packet in (a).

| | |
|---|---|
| ϕ 0 1 0 ϕ | ϕ 0 ϕ 0 ϕ |
| 0 1 0 0 0 | 0 ϕ 0 0 0 |
| 0 ϕ 0 0 1 | 0 1 0 0 1 |
| 0 0 0 ϕ 0 | 0 0 0 ϕ 0 |
| 0 1 ϕ 0 0 | 0 1 1 0 0 |
| (*a*) | (*b*) |

The novel service discipline which operates in this fashion is called revision scheduling, the purpose of which is to maximize the number of packets getting through and the number of output lines being busy simultaneously. We first select packets according to one-shot discipline until all remaining packets interfere with the selected packets. Then, we lift the restriction of the one-shot rule, to allow the individual copies of the remaining packets to contend for the remaining output channels. The packets with residual number of copies, called residual packets, would contend for outputs with fresh packets altogether in the next time slot. Obviously, in a specific time slot, revision scheduling is the same as WS call splitting in the sense of utilization of the output lines. Simulation will show us that the overall delay-throughput performance of revision scheduling is better than WS call splitting scheme.

With the cyclic-priority input access scheme, the revision scheduling can be implemented by circulating the token two times around a ring connecting the $N$ input ports. At the token's first visit to the input port, the reservation must be made for the whole packet according to the one-shot rule. At its second visit to the input port, the remaining HOL packets will make reservations for their copies following the WS call splitting discipline. There are two main difficulties in the implementation. One is that some additional control functions are required for identifying the subset destinations of the remaining packets, which might cause complicated control functions. The other is that the speed limit prevents tokens from visiting all input port two times when the switch size increases. To overcome these difficulties, we can use a cross-bar structure and coordinate the arbiters by using combinational logic circuits.

## 2.3. PERFORMANCE COMPARISON BY SIMULATION

We have discussed four categories of call scheduling and their implementation considerations. Simulations of the four disciplines were carried out. Once the contention happens, a random selection policy is used to make a choice. The packet delay and throughput were determined as the basic performance measurements of the scheduling schemes. It is assumed that the arrival of packets to the input is described by a Bernoulli process and each packet generates copies according to independent Bernoulli trial on each output port, with success probability $p$. This particular distribution was chosen because it provides a way of generating multiple copies which is easy to implement in a simulation. In any case, our focus is upon the comparison of a number of service disciplines. There is no reason to believe that this distribution and random selection policy would lead to contradictory results with alternative distributions and policies. The simulation results for a 32×32 switch are shown in Figure 2.7, where $\bar{c}$ is the average number of copies generated by a packet. We choose the 32×32 switch here because it is supposed to be a module of the configuration of the large size switch. The conclusion made from the plots are also true for the switch with various sizes. Simulations were run over $10^6$ slots for each value of $\lambda$ and $p$ considered. The mean values are within 1% of the true mean with 95% confidence. A brief introduction of the simulation program can be seen in Appendix 2.

As a reference, we plotted results for a output queueing scheme which is required when the $N \times N$ switch fabric runs $N$ times as fast as the inputs and outputs or has $N$ parallel paths. All arrival packets and their copies will be transferred to a set of queues corresponding to their destinations at the slot immediately after their arrival. There is no queueing required at input port, since there is no more than one packet arriving per slot per input link.

The one-shot discipline bears the most stringent restrictions on scheduling so that the saturated throughput is the lowest among the four. However, it is simple and easy to implement.

The SS call splitting discipline relaxes the one-shot restriction in the heavy traffic case. But, it seems rigid in the light traffic case and results in degradation in the performance. Even if there is only one HOL packet, $C$ time slots are still needed if the packet has $C$ desired destinations.

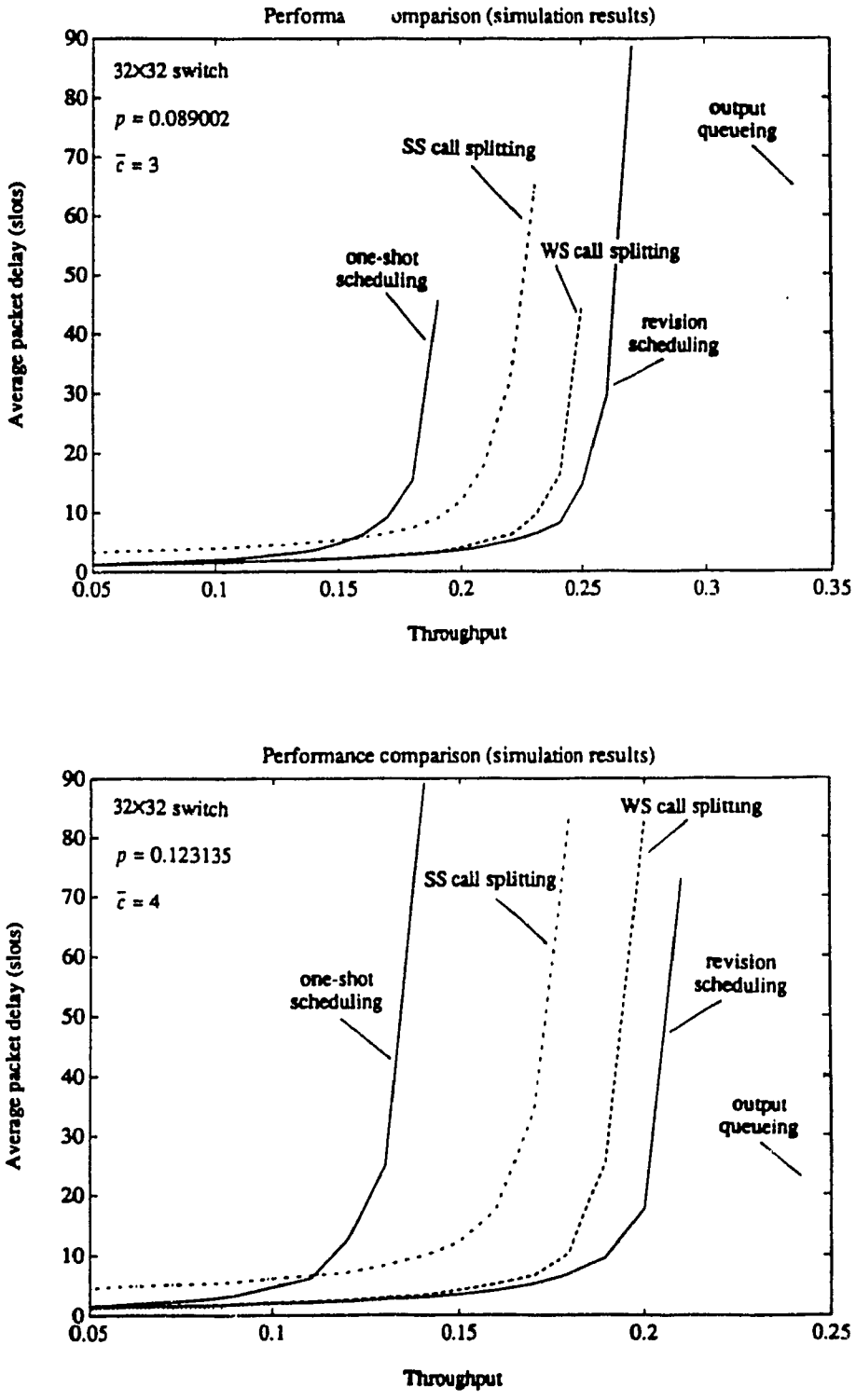



Figure 2.7. Comparison of the delay-throughput performance

The SS call splitting discipline does not require the copy network, however, it does need distributed duplicating operations.

The WS call splitting pursues full use of the output lines, hence achieves higher throughput. It has the same complexity as the crossbar switch does. It is HOL blocking that prevents it from yielding the throughput which the output queueing scheme can provide.

Revision scheduling seeks to mitigate the HOL blocking by trying to get as many packets through as possible while insuring full use of the output lines. This discipline would give an indication of the best performance under the assumption of input port queueing. The switch structure to implement WS call splitting discipline is applicable to the revision scheduling, however, a complicated control logic is predicted.

## 2.4. CONCLUSION

We have classified and discussed four call scheduling disciplines and their implementation considerations. Revision scheduling is preferred as far as the delay-throughput performance is concerned. Taking into account the complexity of implementation as well as the performance, none of the four categories seems to have distinctive superiority over the others. There is a trade-off between the performance and switch structure complexity. Future research should be focused on the switch structures to implement these disciplines efficiently, and to adapt to practical networks.

## APPENDIX 2: Simulation Model

Simulation of the multicast packet switch with various service disciplines and various selection policies for input access scheme under different traffic arrival models and copy distribution models, is implemented through a software package SIMU which was written in C and running at Sun Sparc WorkStation. SIMU allows us to specify particular switching strategy very easily. All the options we can choose are listed in Table A2.1. The block diagram of SIMU is shown in Figure A2.1.

Table A2.1. Options of SIMU

| Description | Option |
|---|---|
| scheduling discipline | one-shot scheduling<br>SS call splitting<br>WS call splitting<br>revision scheduling |
| selection policy | cyclic priority<br>random selection<br>neural network selection |
| packet arrival model | binomial<br>Poisson |
| copy distribution model | binomial<br>deterministic |
| test mode | normal<br>uniform<br>fresh |

In the simulation, time is segmented into fixed size slots, and a slot is the minimum duration of any event. Packet arrivals at each input port are generated according to a binomial or a Poisson distribution with average rate $\lambda$. Then, for each new HOL packet, copies are distributed to output ports as the result of Bernoulli trials with parameter $p$ for each output port, or with a deterministic distribution such that a packet generates and evenly distributes $NC$ (a constant number of) copies to the destinations. The HOL packets contend for output based on one of the four scheduling disciplines, incorporating one of the three selection policies. The blocked packets in current slot are held in buffers for future contention in following slots. In order to keep track of all events during the simulation and to facilitate the manipulation of data, a well defined data structure is adopted. The simulation was made over a specified number of slots for each

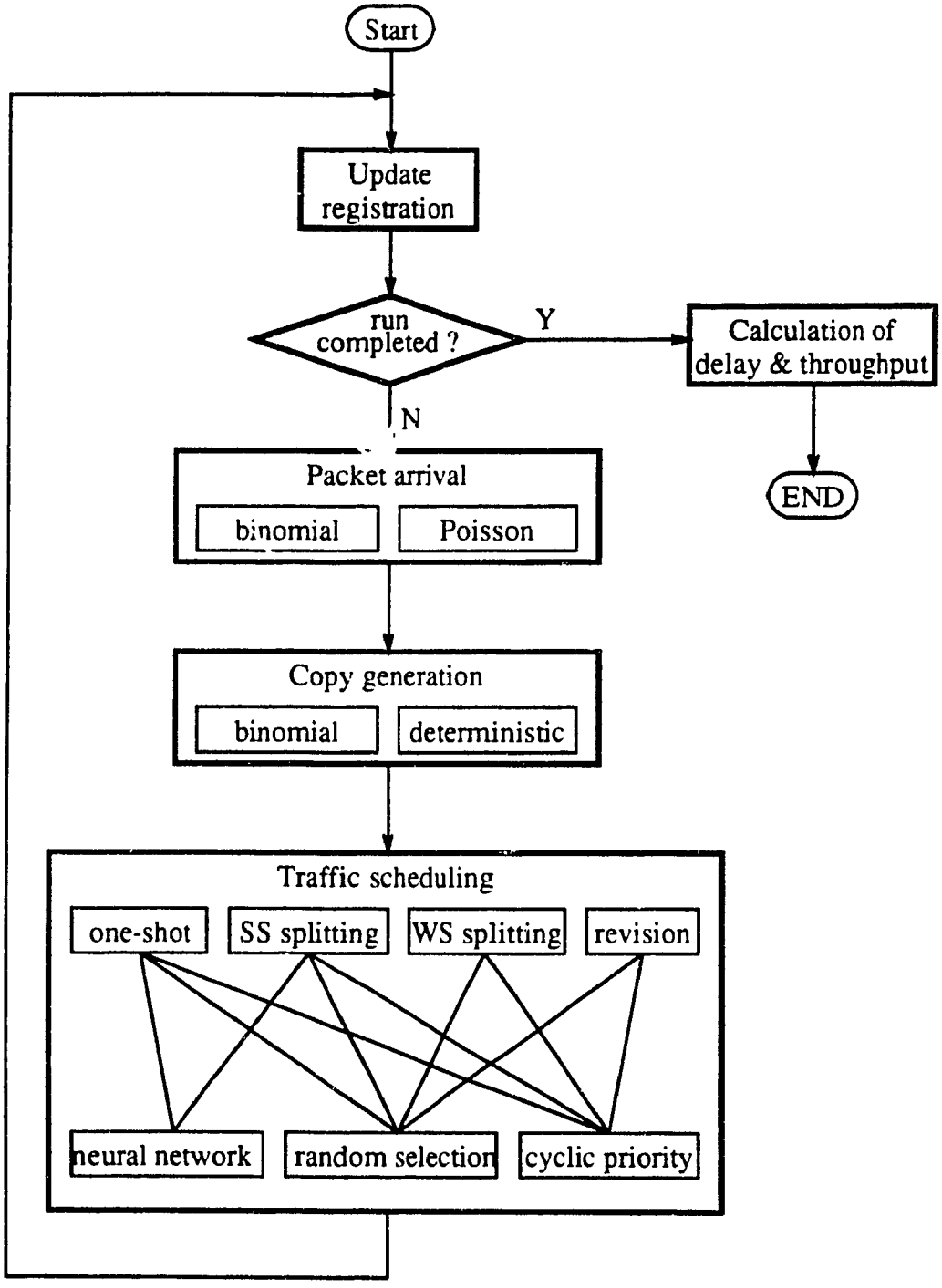



Figure A2.1. A flowchart of the program SIMU

specified configuration. The simulation yields a set of performance measures: system load, throughput, delay and confidence interval corresponding to a set of defined system parameters.

## REFERENCES 2

[1] J. Y. Hui, T. Renner, "Queueing strategies for multicast packet switching", Proc. of Globecom'90, pp.1431-1437, San Diego, CA, Dec. 1990.

[2] T. T. Lee, "Nonblocking copy networks for multicast packet switching", IEEE J. Select. Areas Commun., Vol.6, No.9, pp.1455-1467, Dec. 1988.

[3] T. T. Lee, R. Boorstyn, E. Arthurs, "The architecture of a multicast broadband packet switch", Proc. IEEE Infocom'88, pp.1-8, New Orleans, LA, March 1988.

[4] J. Y. Hui, E. Arthurs, "A broadband packet switch for integrated transport", IEEE J. Select. Areas Commun., Vol.5, No.8, pp.1264-1273, Oct. 1987.

# CHAPTER 3

# CONTENTION RESOLUTION ALGORITHMS

3.1. REVIEW OF THREE ALGORITHMS

3.2. CYCLIC PRIORITY ACCESS SCHEME WITH ITS COMBINATIONAL LOGIC IMPLEMENTATION

3.3. NEURAL NETWORK BASED ALGORITHM

3.3.1. Preliminary of Hopfield Model of Neural Networks

3.3.2. A Model for One-Shot Scheduling

3.3.3. An Extension Model — Windowed service

3.3.4. A Model for SS Call Splitting

3.4. CONCLUSION

REFERENCES 3

We have investigated the relative performance of the multicast scheduling disciplines which are designed to regulate traffic and prevent output contention for input port queueing switches. Either the centralized controller or the centralized arbiter is equipped for the purpose. In this chapter, we will discuss contention resolution algorithms by which these disciplines are implemented. The contention resolution algorithm tells us how to select contention-free packets under a specific scheduling discipline. We first introduce some interesting schemes; thereafter, we will propose two novel algorithms, in Section 3.2 and 3.3 respectively.

## 3.1. REVIEW OF THREE ALGORITHMS

A banyan network is widely used as the routing network of a switch. The banyan network itself is a blocking network. However, the banyan network becomes internally nonblocking if the destinations of input requests are sorted in ascending or descending order. The Batcher's bitonic sorting network can be used to achieve this requirement. Therefore, a combination of Batcher's bitonic sorting network and a banyan routing network is a well known self-routing nonblocking packet switch. The nonblocking property holds if and only if the destinations of all input packets are distinct. However, blocking will occur if there is output contention, i.e. the requests for output ports are not distinct. A good collection of discussions of this issue can be found in [1].

Several contention resolution schemes for point-to-point transmission were proposed [2][3][4] for resolving output conflicts of a Batcher-banyan packet switch, which guarantee that at most one packet addressed to a given output enters the switch at any time.

Huang and Knauer [2] proposed a re-entry network as shown in Figure 3.1. At the output of the sorting network, a packet loses the contention if the packet on the preceding line in the sorted order has the same destination address. The packets which lose the contention are then concentrated by a concentration network, and are fed back to the input of the Batcher network for reentry. The packets which win the contention are not concentrated due to purged packets. Consequently, another concentration network is required in front of the banyan network for skewing all packets to the top lines, filling all holes left behind by purged packets. The drawback of this approach is that the two extra expanded concentration networks require more chip sets and

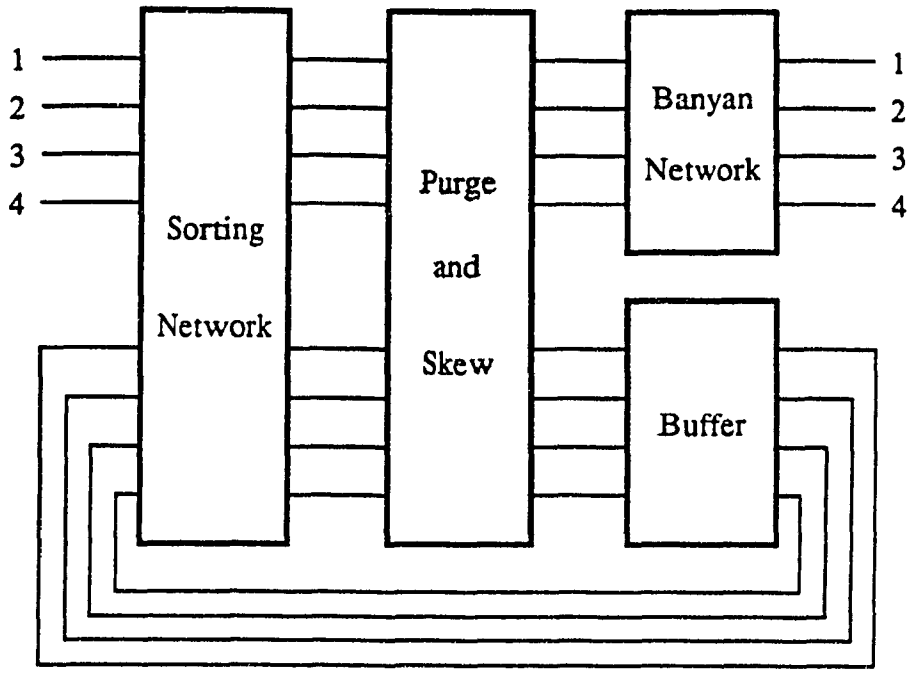
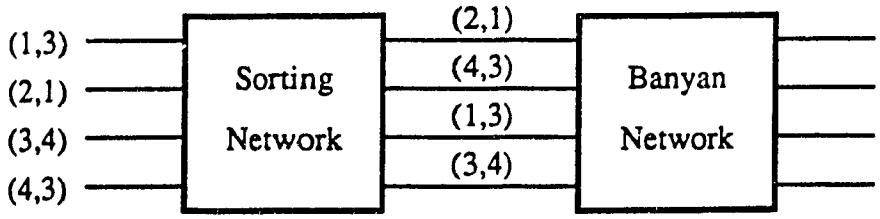



Figure 3.1. A re-entry network for contention resolution

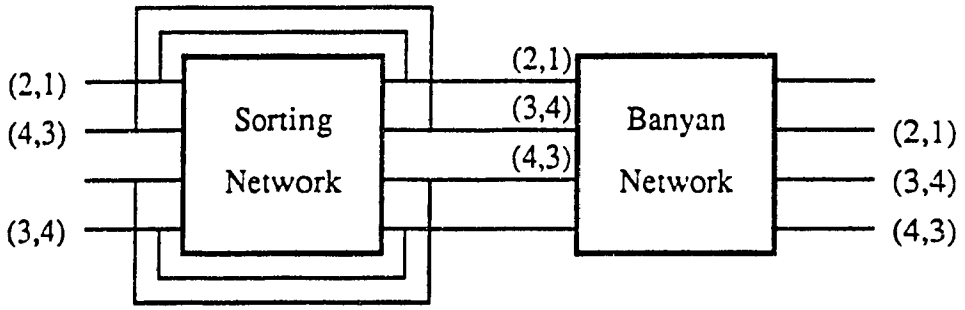
subsystem designs. Furthermore, FIFO operation will be lost.

Hui and Arthurs [3] suggested a 3-phase algorithm to solve contention problem as shown in Figure 3.2. In phase one, each input port sends a request for a port destination through a Batcher sorting network, which sorts the request destinations in ascending order so that we may easily purge all but one request for the same destination, Figure 3.2a. The winning request acknowledges its originating port from the output of the Batcher network, with the acknowledgment routed through a Batcher-banyan switch according to its input port source address in phase two, Figure 3.2b. In phase three, the acknowledged input port then sends the full packet through the same Batcher-banyan self-routing switch without any contention, Figure 3.2c. Unacknowledged ports buffer the blocked packet for reentry in the next cycle. However, this algorithm requires



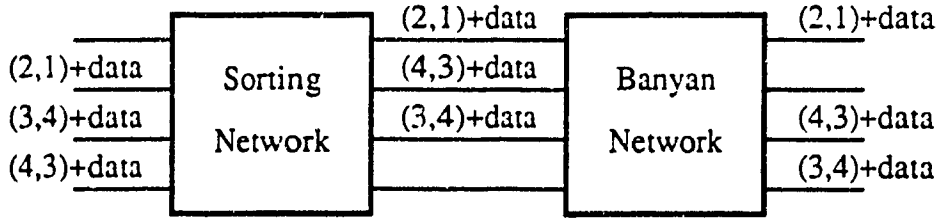

Phase 1: Send and resolve request
* Send source-destination pair through sorting network
* Sort destination in non-decreasing order
* Purge adjacent requests with same destination

(a) Phase one




Phase 2: Acknowledge winning port
* Send ACK with destination to port winning contention
* Route ACK through Batcher-Banyan network

(b) Phase two




Ph se 3: Send packet
* ' knowledged port send packet through Batcher-Banyan network

(c) Phase three

Figure 3.2. Three-phase algorithm

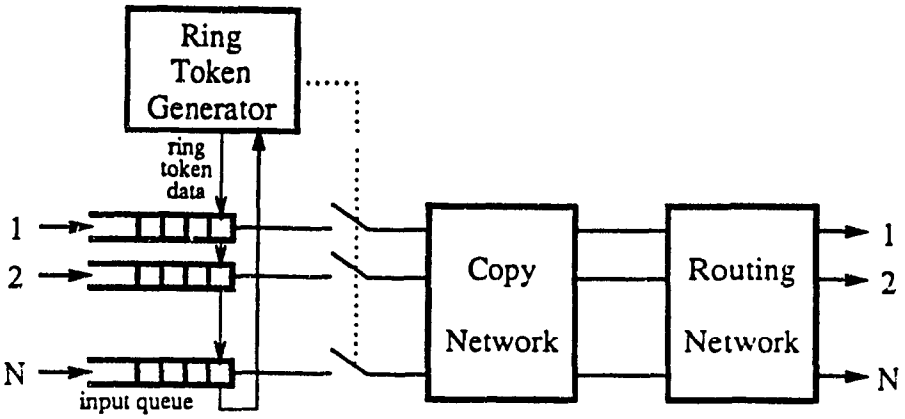



Figure 3.3. Ring reservation-based contention resolution

stringent timing and synchronization.

Bingham and Bussey [4] proposed a reservation-based contention resolution method as shown in Figure 3.3. Unlike the previous two feed back algorithms, the reservation-based contention resolution method is a ring token passing scheme. At the beginning of a time slot, a cleared token is issued by a token generator. The token has an $N$-bit field to indicate the availability of $N$ output ports. The token is circulated around a ring connecting the input ports. When the token comes by, the packet at the head of each input buffer will make the reservation. If the intended output is successfully reserved, the packet is transmitted at the next time slot. The reservation cycle and transmission cycle can be overlapped to minimize the overhead. But, it has a growth problem because the duration of the reservation cycle is proportional to the number of input ports. Generally speaking, this is an intrinsic problem of the centralized control schemes.

Certainly, all three resolution algorithms remain valid for Batcher-banyan based multicast switch in a certain sense. The re-entry policy and the three-phase algorithm have the advantage

of distributed operation, however, both need an extra buffer between the copy network and routing network in addition to input port buffer in the front of copy network [2][5]. Because of that, the FIFO principle for incoming packets has been lost. Copies may be out of sequence upon reaching the destination.

It is straightforward to apply the ring-reservation algorithm to multicast switches. Although simple, the ring reservation method has the drawback of built-in unfairness since the requests at higher numbered ports are deferred more often. To eliminate this drawback, we suggest a cyclic-priority input access scheme, which is the topic of next section.

## 3.2. CYCLIC PRIORITY ACCESS SCHEME WITH ITS COMBINATIONAL LOGIC IMPLEMENTATION

The cyclic priority input access scheme is a derivative of the ring-reservation method. We assign $N$ levels of priority to the $N$ input ports at the beginning of each time slot and rotate the assignment cyclically slot-by-slot. An input port with priority level $i$ in the current slot will have priority level $i-1$ in the following slot. Input port with priority level 1 (the highest priority) in current slot will have priority level $N$ (the lowest priority) in the following slot, and so on. The HOL packets make reservations according to their priority order. But, with a ring token passing scheme we still have a problem that the speed limit prevents tokens from visiting all input ports when the switch size increases.

In the last chapter, we formulated contention resolution problem as a decision problem in which it is required to obtain a decision matrix **O** corresponding to a specific scheduling discipline for a given transmission request matrix **I**. A combinational logic circuit is devised to implement the decision, by incorporating a cyclic priority access policy. This circuit has the advantages of being simple and fair. Without need of stringent timing and synchronization, it is expected to operate fast.

A basic element of a decision board is shown in Figure 3.4 where four gates (one AND, one NAND, one OR and one INV) are connected to make the decision and propagate the vertical for-

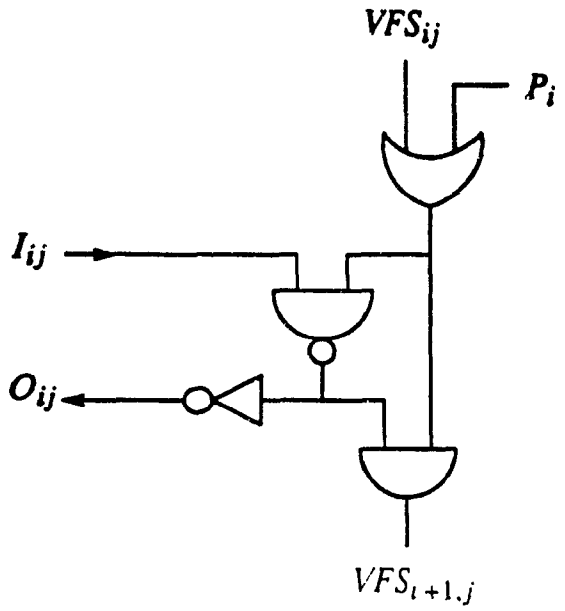
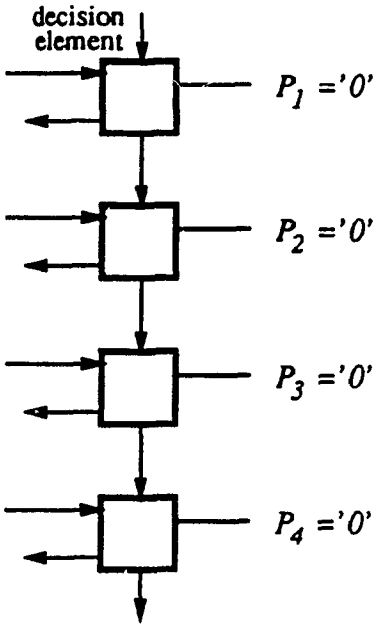



Figure 3.4. A basic decision element

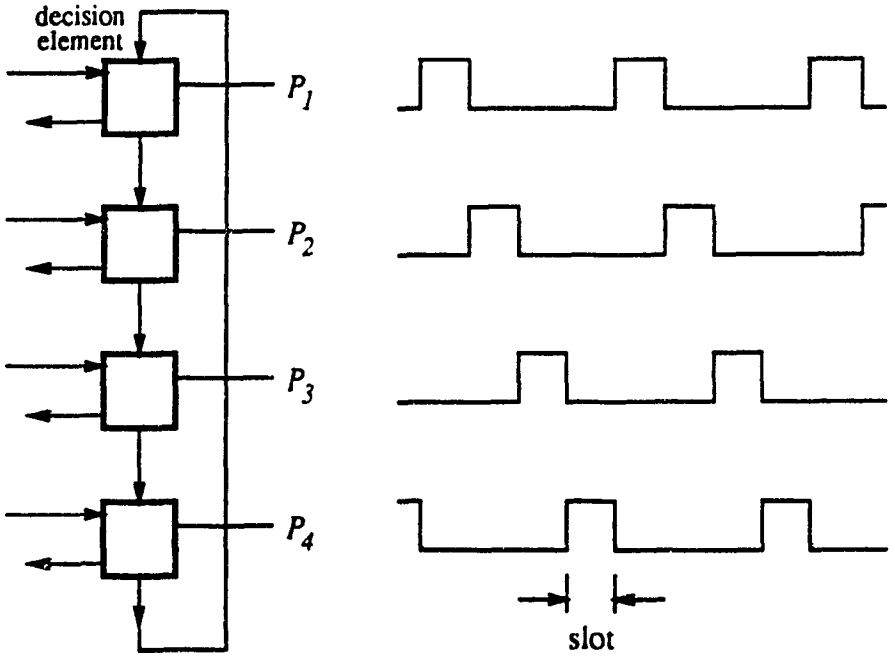
Table 3.1. Truth table of a basic element

| $P_i$ | $VFS_{ij}$ | $I_{ij}$ | $O_{ij}$ | $VFS_{i+1,j}$ |
|---|---|---|---|---|
| 0 | 0 | X | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | X | 0 | 0 | 1 |
| 1 | X | 1 | 1 | 0 |

bidden signal ($VFS$). $I_{ij}$ corresponds to the element of the transmission request matrix; $I_{ij}$ = 1 indicates a message in input $i$ seeking to be sent to output $j$. $O_{ij}$ corresponds to the element of the decision matrix; $O_{ij}$ = 1 indicates the request from input $i$ to output $j$ being approved. Since in each column, only one request can be accepted, a $VFS_{ij}$ is needed to cooperate between adjacent decision elements. $VFS_{ij}$ = 1 signals that output $j$ has not yet been selected, input $i$ is allowed to make a reservation if needed. $VFS_{ij}$ = 0 signals that the output $j$ has been used by either input $i-1$ or previous inputs. $P_i$ is a priority control signal which will be described later. The truth table for the basic element is shown in Table 3.1.




(a)




(b)

Figure 3.5. Cyclic priority assignment

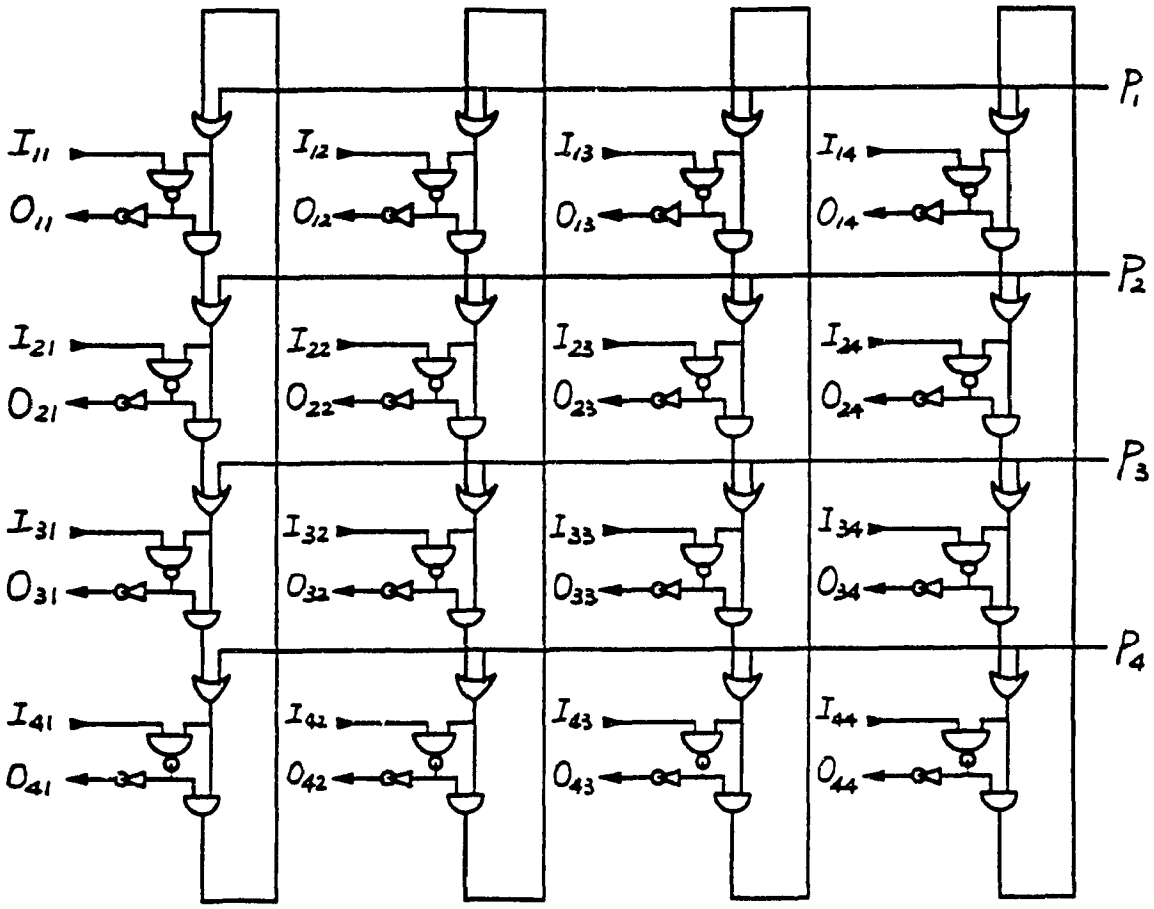



Figure 3.6. A 4×4 decision board for WS call splitting discipline

A column of such decision elements is used by an output to select the packet (Figure 3.5a). Once an input has been selected, all of the following requests are eliminated. For fairness, we connect $VFS_{N+1,j}$ to $VFS_{1,j}$ to form a ring, and choose input $i$ as the first priority port by setting $P_i = 1$, and $P_l = 0$ for all $l \neq i$, in a slot (Figure 3.5b). The priority level descends along $VFS$ flow. Then, we cyclically change the first priority input port from slot to slot. A 4×4 decision board is

shown in Figure 3.6, which can be used as arbiters in Figure 2.5 for the WS call splitting discipline. It is worthwhile to indicate that this approach can also be applied to unicast case in place of ring-reservation algorithm or three-phase algorithm.

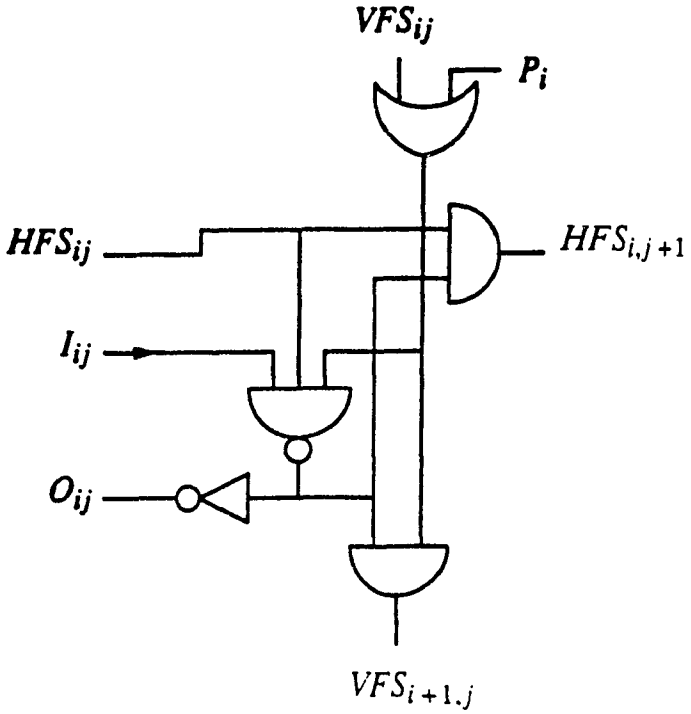



Figure 3.7. A basic element with *HFS*

Table 3.2. Truth table for the basic element in Figure 3.7

| $P_i$ | $VFS_{ij}$ | $HFS_{ij}$ | $I_{ij}$ | $O_{ij}$ | $VFS_{i+1,j}$ | $HFS_{i,j+1}$ |
|---|---|---|---|---|---|---|
| 0 | 0 | X | X | 0 | 0 | $HFS_{ij}$ |
| 0 | X | 0 | X | 0 | $VFS_{ij}$ | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | X | 0 | 0 | 0 | 1 | 0 |
| 1 | X | 1 | 0 | 0 | 1 | 1 |
| 1 | X | 0 | 1 | 0 | 1 | 0 |
| 1 | X | 1 | 1 | 1 | 0 | 0 |

To fulfill the SS call splitting discipline, we should introduce a horizontal forbidden signal (*HFS*), since only one can be selected from each row. One more AND is added to basic element as illustrated in Figure 3.7. The corresponding truth table is shown in Table 3.2. A 4×4 decision

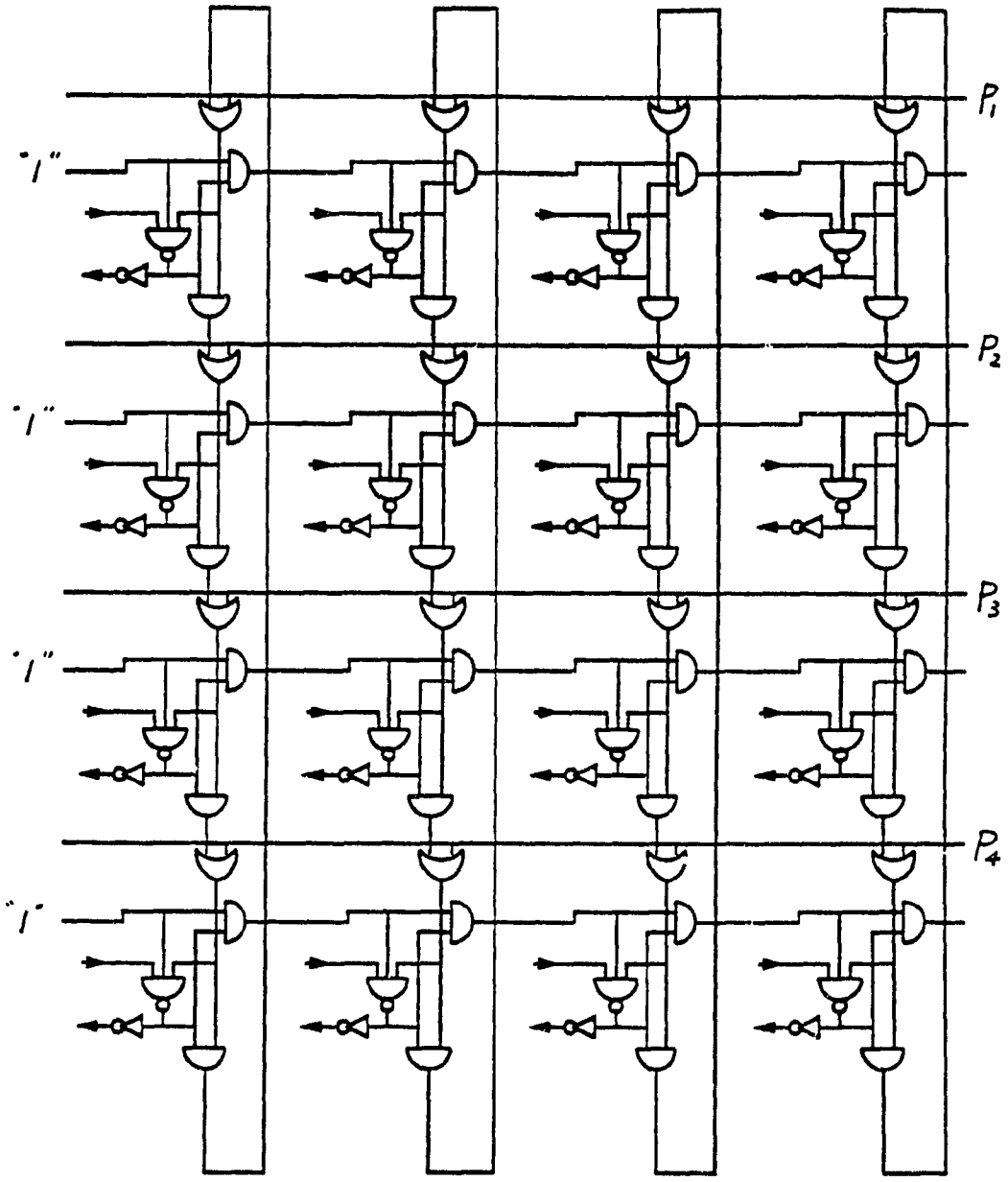



Figure 3.8. A 4×4 decision board for SS call splitting discipline

board for SS call splitting discipline is depicted in Figure 3.8.

The one-shot scheduling can be interpreted as that *HFS* must come into effect on a row both forward and backward if any request in this row is eliminated by *VFS*. A pair of ANDs are used

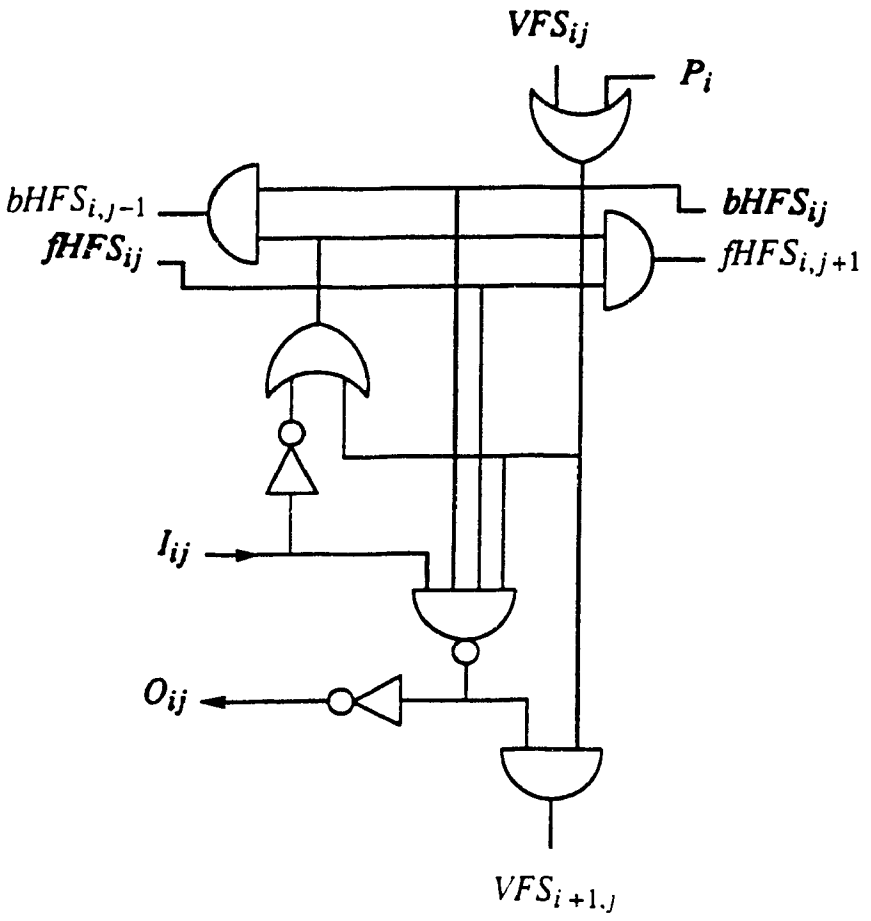



Figure 3.9. A basic element with *fHFS* and *bHFS*

Table 3.3. Truth table for the basic element in Figure 3.9

| $P_i$ | $VFS_{ij}$ | $fHFS_{ij}$ | $bHFS_{ij}$ | $I_{ij}$ | $O_{ij}$ | $VFS_{i+1,j}$ | $fHFS_{i,j+1}$ | $bHFS_{i,j-1}$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | X | X | X | 0 | 0 | 0 | 0 |
| 0 | X | 0 | X | X | 0 | $VFS_{ij}$ | 0 | $bHFS_{ij}$ |
| 0 | X | X | 0 | X | 0 | $VFS_{ij}$ | $fHFS_{ij}$ | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | X | X | X | 0 | 0 | 1 | 1 | 1 |
| 1 | X | X | X | 1 | 1 | 0 | 1 | 1 |

to propagate *fHFS* and *bHFS* respectively as seen in Figure 3.9. The truth table is as in Table 3.3. A fully connected 4×4 decision board for one-shot scheduling is shown in Figure 3.10. where $P_i$'s are priority control signals, and the cyclic-priority scheme is easy to implemented by this

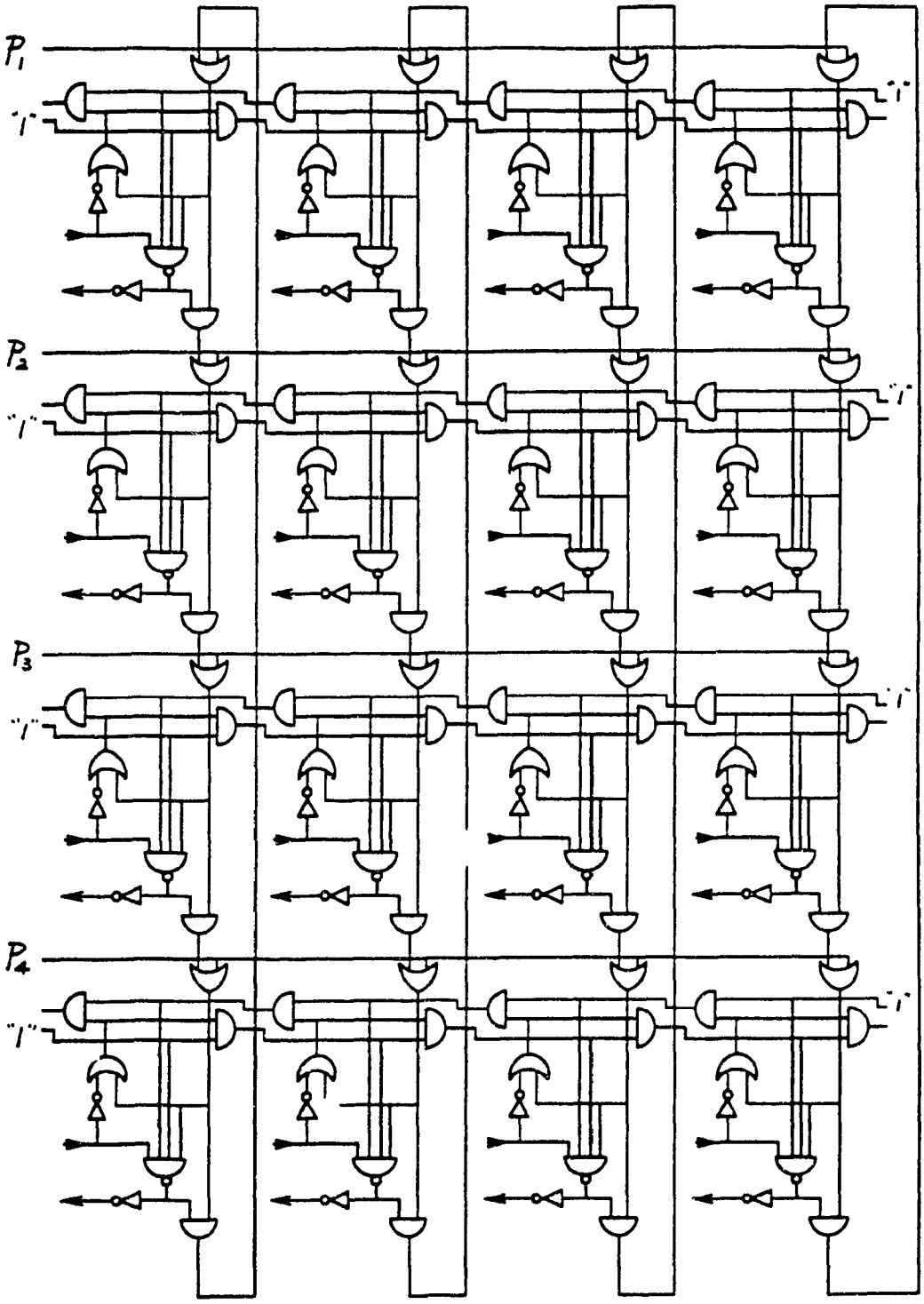



Figure 3.10. A 4×4 decision board for one-shot scheduling

structure.

To implement revision scheduling, we need a simple way to switch the operations between one-shot and call splitting. An ATM slot is divided into two phases. A minor modification as shown in Figure 3.11 enables the circuit to connect or disconnect the *HFS*s by setting $R = 0$ or $R = 1$, alternately in two phases of each time slot.

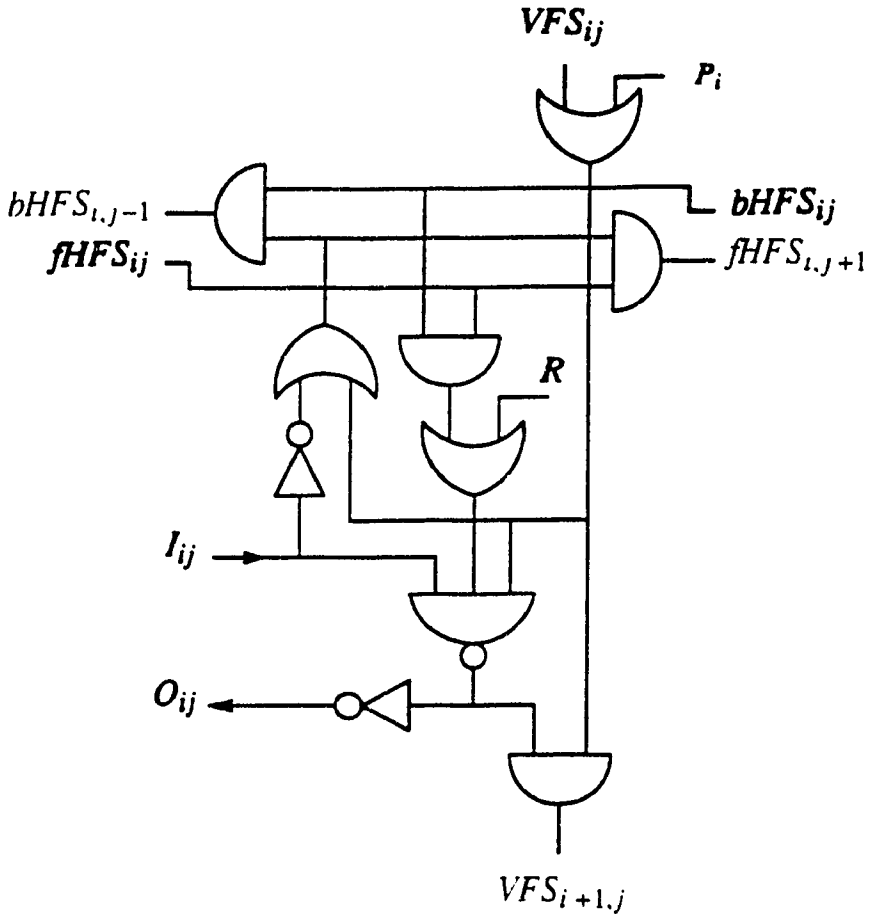



Figure 3.11. A basic element for revision scheduling

These circuits are stable because they are open without any feedback. The decision time is defined as the propagation time of the signal through a proper route. Since the decision period and transmission cycle can be overlapped, the maximum decision time can be a slot time. Assume that the speed of each input line is 150Mbps, each packet contains 53 bytes or 424 bits. The slot time is 2826ns. For simplicity, we assume that various logic gates have the same

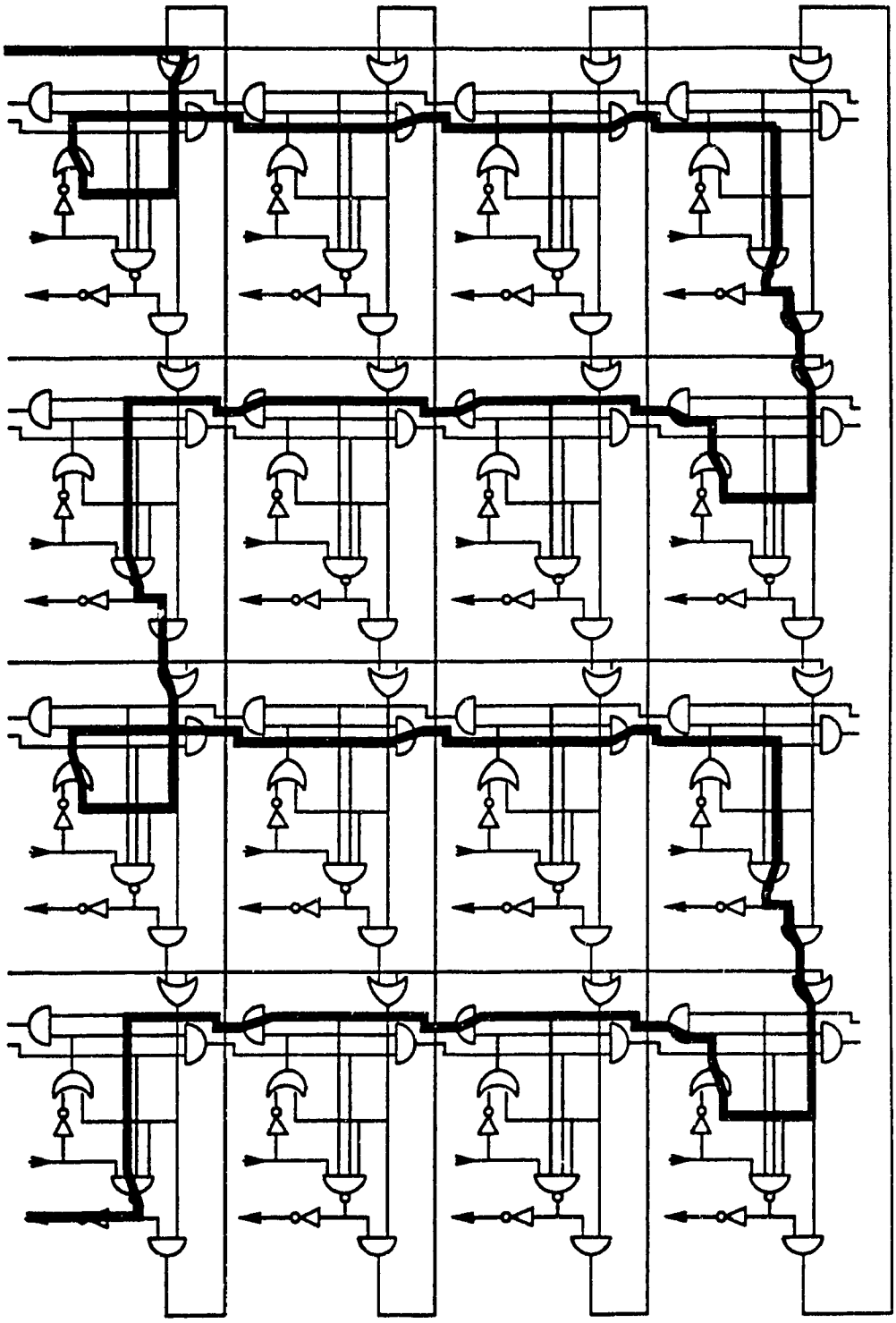


Figure 3.12. An example of the longest signal routes

propagation delay. Figure 3.12 shows us one example of the longest signal routes. The number of gates needed for each decision board and the number of gates along the longest signal route are listed in Table 3.4, where $N$ is the switch size.

Table 3.4. Comparison in terms of the number of gates

| Scheduling | Number of gates in board | Number of gates in the longest route |
|---|---|---|
| WS call splitting | $4N^2$ | $3N$ |
| SS call splitting | $5N^2$ | $4N$ |
| One-shot scheduling | $8N^2$ | $N^2+3N$ |
| Revision scheduling | $10N^2$ | $N^2+8N$ |

With current high speed digital IC technologies, sub-100 ps gate-delay can be achieved [6] with 0.3 μm CMOS, 0.7 μm GaAs DCFL, or 0.5 μm emitter Si ECL. The Table 3.5 shows realizable switch sizes $N$ assuming average gate-delay $t_{gd}$ = 150ps, 500ps and 2ns respectively.

Table 3.5. Realizable switch size

| Scheduling | Realizable switch size | | |
|---|---|---|---|
| | $t_{gd}$ = 150ps | $t_{gd}$ = 500ps | $t_{gd}$ = 2ns |
| WS call splitting | 4096 | 1024 | 256 |
| SS call splitting | 4096 | 1024 | 256 |
| One-shot scheduling | 128 | 64 | 32 |
| Revision scheduling | 128 | 64 | 32 |

We have demonstrated an implementation of cyclic priority selection scheme. We point out that the random selection policy is a mathematical artifice which is used in performance analysis. In real systems, one has to prescribe some kind of order or define some kind of rule, instead of relying on the random action of electronic components. However, through a large number of simulations, we found that the delay-throughput performance for both cyclic priority selection and random selection policies does not show much differences, except that cyclic priority policy attains a slightly smaller confidence interval than the random selection policy.

## 3.3. NEURAL NETWORK BASED ALGORITHM

In a unicast switch, delay-throughput performance does not depend on selection policy, hence does not raise questions on scheduling discipline. Unlike the unicast switch, the multicast switch brings a new subject on the way that the copies of the HOL packets are handled. From last chapter, we knew that different scheduling disciplines give different performance. Not only that, here we will show, even if under the same scheduling discipline, one is still able to improve the delay-throughput performance. In other words, we focus on the selection policy within a discipline.

For example, a 4×4 switch has a set of transmission requests as follows

$$\begin{matrix} 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

Input port 1 seeks to send message to output ports 1, 3 and 4; input 2 seeks to send message to output 3; and so on. Under the assumption of one-shot scheduling, there are two possible sets of packets which can be selected to get through the switch without conflict.

$$\begin{matrix} \phi & 0 & \phi & \phi \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \quad (1) \qquad\qquad \begin{matrix} 1 & 0 & 1 & 1 \\ 0 & 0 & \phi & 0 \\ \phi & \phi & 0 & 0 \\ 0 & 0 & 0 & \phi \end{matrix} \quad (2)$$

In the first case, throughput is 0.25 and output utility is 75%; while in the second case, throughput is 0.75 and output utility is 100%. Thus, it seems that we need some means to select a particular set of packets in an effort to maximize the packet throughput.

Similarly, under the assumption of SS call splitting discipline, we can select winning copies in different ways such as

$$\begin{matrix} \phi & 0 & 1 & 1 \\ 0 & 0 & \phi & 0 \\ 1 & \phi & 0 & 0 \\ 0 & 0 & 0 & \phi \end{matrix} \quad (1) \qquad \begin{matrix} 1 & 0 & \phi & 1 \\ 0 & 0 & 1 & 0 \\ 1 & \phi & 0 & 0 \\ 0 & 0 & 0 & \phi \end{matrix} \quad (2) \qquad \begin{matrix} 1 & 0 & 1 & 1 \\ 0 & 0 & \phi & 0 \\ \phi & 1 & 0 & 0 \\ 0 & 0 & 0 & \phi \end{matrix} \quad (3)$$

Each row and each column has at most one 1 being selected, which fulfill the requirement. But as we have seen, in the first case, four messages are transmitted to the output lines; in the second

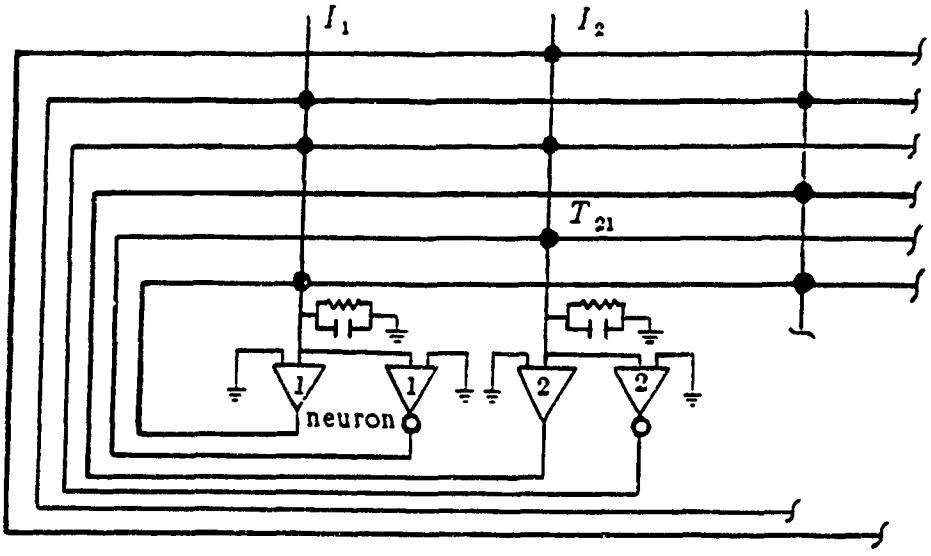
and third cases, only three messages are allowed to get through. This indicates us the potential to enhance the utility of the output lines, thus increase the throughput correspondingly.

In this section, we suggest a neural-network-based contention resolution algorithm for the one-shot and the SS call splitting disciplines. In addition to contention resolution, a further objective would be to maximize the throughput. We formulate our contention resolution problem as an optimization problem. This approach allows us to select, from all HOL requests, a particular set of packets or copies, which ensures contention-free access to output ports and maximizes packet throughput. We have to be clear that WS call splitting discipline is able to make full use of output lines and unable to gain any more through neural network approach. In fact, the revision scheduling is an improved derivation of the WS call splitting discipline.

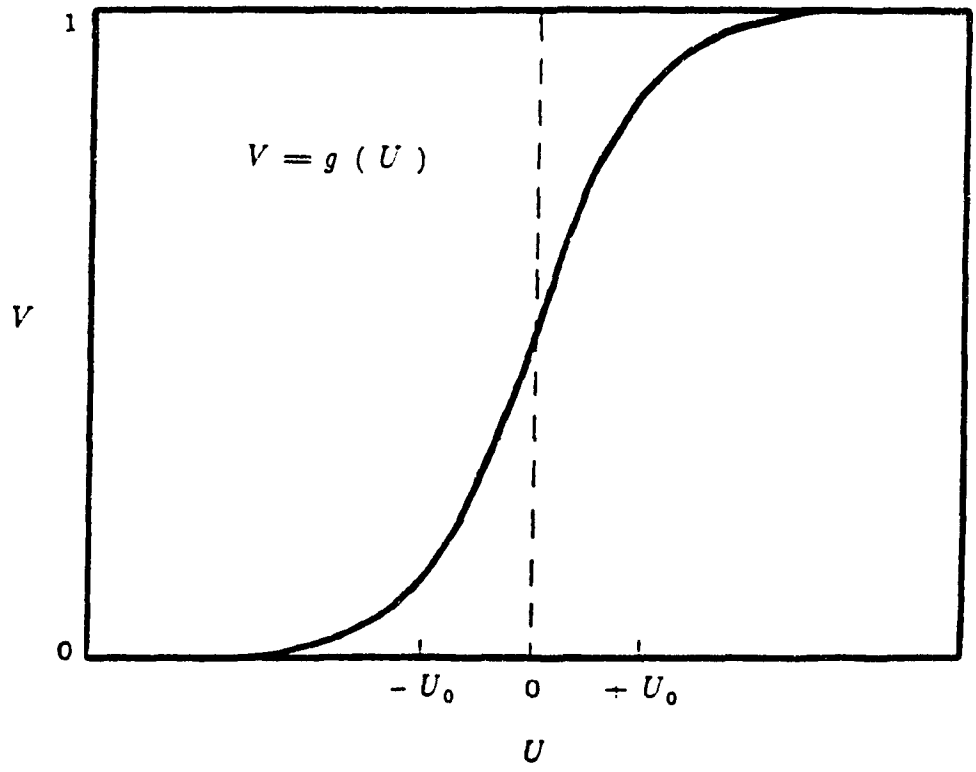
The remainder of this section is organized as follows. Section 3.3.1 describes the general structure of the Hopfield model of neural network. Section 3.3.2 discusses the neural-network-based prototype scheme for the one-shot scheduling. An extension with windowed service is described in Section 3.3.3. Section 3.3.4 discuss a model for the SS call splitting discipline.

### 3.3.1. Preliminary of Hopfield Model of Neural Networks

Neural networks, implementable with electronic components, allow complex problems similar to those occurring in biology to be solved without the need to follow the details of the circuit dynamics. The general structure of the Hopfield model of neural networks [7][8] which can solve optimization problems is shown in Figure 3.13a. These networks have three characteristics: parallel input channels, parallel output channels, and a large degree of interconnectivity between the neural processing elements. The processing elements, or neurons, are modeled as amplifiers in conjunction with feedback circuits comprised of resistors and capacitors organized so as to model the most basic computational features of neurons. The amplifiers have sigmoid (S-shaped) monotonic input-output relations, as shown in Figure 3.13b. The function $V_j=g(U_j)$ which characterizes this input-output relation describes the output voltage of amplifier $V_j$ due to an input voltage $U_j$. The minimum and maximum outputs of the normal amplifier are taken as 0 or 1. A synapse between two neurons is defined by a conductance $T_{ij}$ which connects the output




(a)




(b)

Figure 3.13. A Hopfield model of neural network

of amplifier $j$ to the input of amplifier $i$. Thus, the matrix $T_{ij}$ defines the connectivity among the neurons. In addition, to model the fact that each neuron computes a nonlinear function of a host of inputs under the influence of its own activation level, the circuits include an externally supplied input biasing current $I_i$ for each neuron which provides direct parallel inputs to drive specific neurons, as indicated in Figure 3.13a.

The following set of coupled nonlinear differential equations describes the dynamics of an interacting system of $N$ neurons by showing how the neuronal state variables change with time under synaptic current influences.

$$\frac{dU_i}{dt} = -\frac{U_i}{\tau} + \sum_{j=1}^{N} T_{ij} V_j + I_i \tag{3.1}$$

where

$$V_j = g(U_j)$$

and $\tau$ is the time constant of the neurons. For an "initial-value" problem, in which the input voltages of the neurons are given values $U_i$ at time $t = 0$, these equations of motion provide a full description of the time evolution of the state of the circuit.

A general form of circuit is described by the value of the synapses $T_{ij}$ and input biasing currents $I_i$. The state of the system of neurons is defined by the value of outputs $V_i$. Hopfield has shown that the equations of motion for a network with symmetric connections ($T_{ij} = T_{ji}$) always lead to a convergence to stable states, in which the outputs of all neurons remain constant. The stable states of a network comprised of $N$ neurons are the local minima of the quantity $E$,

$$E = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} T_{ij} V_i V_j - \sum_{i=1}^{N} V_i I_i \tag{3.2}$$

which is interpreted as computational energy, decreases during the change in neural state with time. The state space over which the circuit operates is the interior of the $N$-dimensional hypercube defined by $V_i$=0 or 1. It can be shown that the minima only occurs on the $2^N$ corners of this space.

The networks of neurons with this basic organization can be used to compute solutions to

specific optimization problems by first choosing connectivities ($T_{ij}$) and input biasing currents ($I_i$) which appropriately represent the function to be minimized. The analog system with initial set of input voltages $U_i$ then converges to a stable state which minimizes the $E$ function. Eventually, the solution to the problem is interpreted from the final stable state.

Here is a summary of how to map a problem onto a neural network:

- Choose a representation scheme which allows the outputs of the neurons to be decoded into a solution to the problem.
- Construct an energy function whose minimum value corresponds to "best" solutions to the problem to be mapped.
- Derive connections ($T_{ij}$) and input biasing currents ($I_i$) from the energy function; these should appropriately represent the instance of the specific problem to be solved.
- Set up initial values of the input voltages ($U_i$) which completely determines the stable output voltages ($V_i$) of the neurons in the network.

### 3.3.2. A Model for One-Shot Scheduling

The objective of this model is to choose a combination of input ports such that a maximum number of packets get through without output contention between their copies. At the same time, it may enhance the utility of the output lines. To map this problem onto the Hopfield model, we require a representation scheme which allows the binary output states of the neurons to be decoded into a specific combination of input ports. For example, for a 10-input switch, if the HOL packets from input ports ( 1, 3, 4, 7, 8, 9 ) are able to get through without conflict, the output state of a set of 10 neurons represents this solution by $V$ = ( 1, 0, 1, 1, 0, 0, 1, 1, 1, 0 ). For an $N$-input switch, $N$ neurons are needed to represent this switching problem. An array of $V = ( V_1, V_2, \ldots\ldots, V_N )$ corresponds to the acceptable traffic combination, in which $V_i = 1$ denotes that the packet in input port $i$ can get through in current slot, $V_j = 0$ denotes that the packet in input port $j$ is blocked and may try again in next slot. In total, we have $2^N$ possible traffic combinations.

To enable an $N$-neuron network to compute a solution to the problem, we construct an energy function in which the lowest energy state (the most stable state of the network) corresponds to the contention-free traffic combination and the maximized throughput. Under the assumption of the one-shot discipline, we define two input ports in contention as any one of the copies from one port involving in the contention with one of the copies from another port. We represent contention characteristic of transmission requirement by a symmetric $N \times N$ matrix $D$, in which $D_{ij} = 1$ denotes input ports $i$ and $j$ form a contention pair, and $D_{ij} = 0$ indicates that there is no contention occurring between input $i$ and input $j$.

Our optimization problem can be separated into two requirements. First, the energy function must strongly favor stable states of the contention-free traffic combination. Secondly, from the sets of contention-free solutions, it must favor those representing maximum throughput. An appropriate form for this function can be found by considering that all final normal outputs will be 0 or 1 which corresponds to the local minima of the energy function

$$E = \frac{A}{2} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} V_i V_j D_{ij} + \frac{B}{2} \left( \sum_{i=1}^{N} V_i - n \right)^2 \tag{3.3}$$

where $A$, $B$ and $n$ are positive constants. The first term contains information about the contention feature of transmission requests and is zero if and only if accepted input ports are contention-free with one another. Hence, it strongly favors stable states which are at least valid traffic combinations and fulfill the first requirement for $E$. The second term approaches zero when the number of accepted input ports approaches $n$. By selecting an appropriate $n$, we can fulfill the second requirement.

So far, we have only considered the packet throughput. As we have mentioned and will detail later, the packet with larger number of copies will be more easily involved in contention and be blocked more often than that with less number of copies. On the other hand, in the case of the one-shot scheduling, there may be more than one optimum solution existing, which have the same number of contention-free packets. Obviously, we prefer the optimum solution with maximum number of copies as well. The neural network model defined by the energy function (3.3) is

not able to fulfill such requirement and should be extended as follows

$$E = \frac{A}{2} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} V_i V_j D_{ij} + \frac{B}{2} \left( \sum_{i=1}^{N} V_i - n \right)^2 + \frac{C}{2} \left( \sum_{i=1}^{N} V_i F_i - n' \right)^2 \tag{3.4}$$

Where $F_i$ denotes the number of copies generated from input port i, and $n'$ is the expected upper limit of the number of copies which are switched in a slot. The added term approaches zero when the number of copies generated by accepted input ports approaches $n'$.

Through equations (3.1) and (3.2), the quadratic terms in this energy function define a connection matrix and the linear terms define input biasing currents. From the correspondence between the constructed energy function (3.4) and Hopfield's general energy function (3.2), the parameter values of the neural network are determined. They are

$$T_{ij} = -A D_{ij} ( 1 - \delta_{ij} ) - B - C F_i F_j \tag{3.5}$$

$$I_i = B n + C W_i n' \tag{3.6}$$

With this $E$ function guiding the dynamics of the circuit, the network should give the solution by choosing a final state after starting in some initial unbiased state.

*Simulation of Neural Network*

A network for contention resolution problem using the connection matrix defined in equation (3.5) and the input bias terms of equation (3.6) was simulated on computer. The transmission requests, including the number of copies and transmission destinations requested by a specific input packet, were generated by performing independent Bernoulli trials (each packet generates a copy to each output port with probability $p$). These requests defined a particular set of $D_{ij}$ and hence $T_{ij}$ through equation (3.5). Obviously, symmetry of $D_{ij}$ and hence $T_{ij}$ ensures that the stable states exist. The analog network for this problem generated the equations of motion

$$\frac{dU_i}{dt} = -\frac{U_i}{\tau} - A \sum_{\substack{j=1 \\ j \neq i}}^{N} D_{ij} V_j - B \left( \sum_{j=1}^{N} V_j - n \right) - C F_i \left( \sum_{j=1}^{N} F_j V_j - n' \right) \tag{3.7}$$

where

$$V_i = \frac{1}{2} \left[ 1 + tanh \left( \frac{U_i}{U_o} \right) \right]$$

These equations of motion show the specific contributions made by the $T_{ij}$ and $I_i$ terms. The system parameters, $A, B, C$ and so on, were chosen through intensive simulations to maximize the throughput and minimize the convergence time. After a large number of simulation trials, the parameters were found as follows

$$A = 160,\ B = 10,\ C = 2,\ n = n' = N,$$

$$U_o = 0.02, \text{ and } \tau = 1$$

Here, $\tau$ is the time constant of the neurons and was set to 1 without any loss of generality. We may observe that with $n = n' = N$, the quantity $E$ will almost never be zero since the throughput will almost never be one. Intuitively, it is better to choose $n$ and $n'$ corresponding to the best solution from slot to slot but which is just what should come out with the evolution of the network. For the sake of simplicity, we choose $n = n' = N$ which is the upper limit of these quantities.

The neural network was simulated by examining and updating neurons simultaneously at intervals which are much smaller than $\tau$. Let $U_i(t+1)$ and $U_i(t)$ be the input voltages at time $t+1$ and $t$ respectively. Then the updating rule used in the simulation was

$$U_i(t+1) = U_i(t) + \left[ \frac{dU_i(t)}{dt} \right] \delta t \tag{3.8}$$

where $\delta t$ is step size for the updates to occur and was chosen to be $10^{-4}$, smaller value of $\delta t$ does not improve the results but increase the simulation run time. At each update, the new values of output voltages were compared to the previous ones. If no two consecutive values differed by more than a threshold of $10^{-6}$, then the system was assumed to have reached a stable state and the simulation was stopped.

Since we have no a priori knowledge of which traffic combination is best, we want to pick the initial values of the neural input voltages $U_i$ without bias in favor of any particular input port, by simply having exactly the same initial values. On the other hand, since there may exist more than one optimum solutions and all these potential optimal solutions are at equal distance from the initial unbiased state, the system has no way to choose one of them given an unbiased start,

and thus can not converge to one of them at all. Therefore, we added some noise $\delta U_{io}$ to the initial values

$$U_{io} = U_o + \delta U_{io} \tag{3.9}$$

to break the symmetry and lead the system to one of the good solutions. $\delta U_{io}$ were each randomly chosen uniformly in the interval

$$-0.1\, U_o \le \delta U_{io} \le 0.1\, U_o$$

This will not be a problem in real neural network, since the noise of the circuit is enough to break the symmetry.

When the solution was interpreted from the final state of the neural network, the blocked packets remain in the HOL and the free lines left by winning packets were occupied by new packets generated by the same method mentioned above.

In order to explain the method, we take an 8×8 switch as an example. A set of probable transmission requests can be described by an 8×8 matrix **I**

$$\mathbf{I} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Where $I_{ij} = 1$ denotes input $i$ has a copy to be sent to output $j$. As we can see, columns 3, 6, and 7 have more than one 1's indicating contentions at output ports. From this matrix, we can derive a symmetric contention matrix $D$,

$$D = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Given the random biased initial input $U_{i0}$, the motion of the neural network begins with the initial state of $V_{i0}$

$V_{10} = 0.485099$
$V_{20} = 0.496633$
$V_{30} = 0.460834$
$V_{40} = 0.506906$
$V_{50} = 0.546588$
$V_{60} = 0.492121$
$V_{70} = 0.496206$
$V_{80} = 0.543037$

Then, as time passes, the neural network converges to the final state

$V_1 = 0.000000$
$V_2 = 1.000000$
$V_3 = 1.000000$
$V_4 = 1.000000$
$V_5 = 1.000000$
$V_6 = 0.000000$
$V_7 = 1.000000$
$V_8 = 0.000000$

The final decision, interpreted from the state, is that five packets at input ports 2, 3, 4, 5, and 7 are able to get through without contention with each other. The other three packets have to be buffered in queue in the current slot.

A large number of simulations were made for switch sizes of 8×8 and 16×16. Figure 3.14 shows the average delay as a function of packet arrival rate for both cyclic priority selection and neural network based optimal selection strategies. For the cyclic priority case simulations were run over $10^6$ slots for each value of $\lambda$ and $p$ considered. The mean values are within 1.0% of the true mean with 95% confidence. For the neural network contention resolution algorithm, runs were made over $10^4$ slots for each value of $\lambda$ and $p$ resulting in averages which are within 1.5% of the true mean with 95% confidence. By comparison, neural network approach provides better performance. This is not surprising since it adapts to the particular set of copies in the HOLs in order to minimize the weighted cost criteria so as to minimize delay. In this sense, it is optimum.

The most interesting result shown by the curves is the performance of the cyclic priority scheme in comparison with neural network scheme. For light loading, below the knee of the delay curves, performance was almost the same. The cyclic priority scheme saturation occurs at

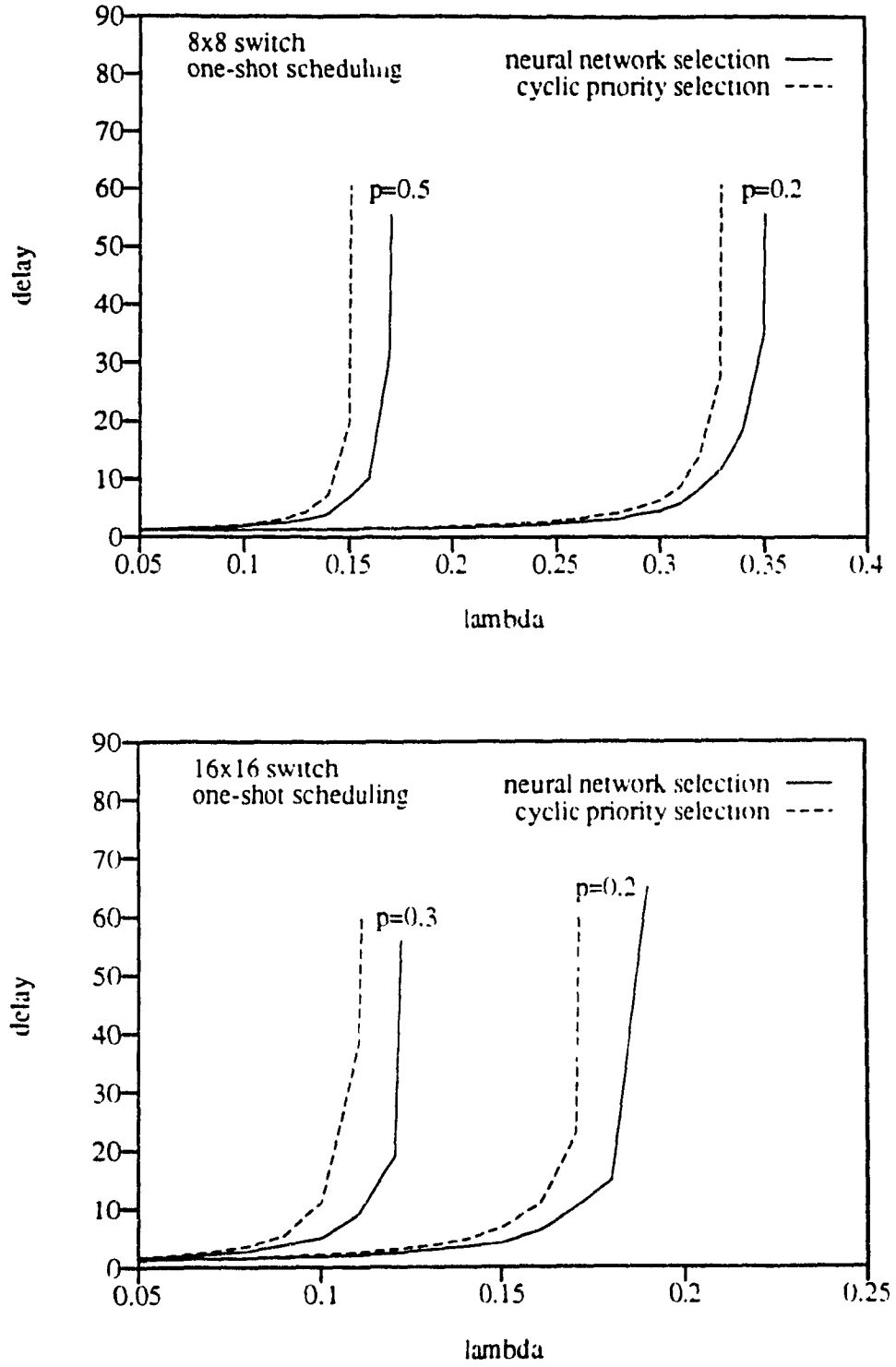



Figure 3.14. A comparison of cyclic priority selection and neural network selection

approximately 10% less traffic. For the cases that we observed, there did not seem to be a trend relating to switch size and the number of copies generated.

We can also use this method to handle contention resolution problems for an $N \times N$ unicast switch. This type of switch has been studied extensively [9]. With saturated input queues, random selection of destination ports, first-in first-out service discipline and $N \rightarrow \infty$, the maximum throughput is equal to $(2 - \sqrt{2}) = 0.5858$, regardless of the contention resolution mechanism [3][4][9]. For small values of $N$, results have been obtained from Markov chain [10]. Our simulation results were verified by these analytical results as presented in Table 3.6, which gives credibility to our simulation model.

Table 3.6. Saturated throughput of $N \times N$ unicast switch

| $N$ | Analytical Results | Neural Network Simulation Results |
|---|---|---|
| 2 | 0.7500 | 0.7493 |
| 4 | 0.6553 | 0.6598 |
| 8 | 0.6184 | 0.6103 |
| 16 | 0.6000 | 0.5980 |
| 32 | 0.5900 | 0.5946 |

### 3.3.3. An Extension Model — Windowed service

In preceding discussion, we assumed the buffers are served FIFO, so only the HOL packets are considered for access to the switch outputs. Intuitively, it seems that the throughput can be increased by relaxing the strict HOL packet contending policy of the input queues. In the multicast case, each input still sends at most one packet (all its copies) into the switch fabric per time slot, but not necessarily the first packet in its queue, and no more than one copy is allowed to pass through the switch fabric to each output in a time slot. We suppose there is a window of size $W$, the first $W$ packets dropping in this window in each input queue contend for access to the switch outputs simultaneously. A window size of $W = 1$ corresponds to input queueing with FIFO buffers.

A straightforward extension of the neural-network-based algorithm can also work here. For an $N$-input switch, $WN$ neurons are needed instead of $N$. An array of $V = (V_1, V_2, \ldots\ldots, V_N, \ldots\ldots, V_{WN})$ corresponds to the acceptable traffic combination. But, at most one packet could be chosen from $W$ packets in the window of each input queue, which follows that only $(W+1)^N$ probable traffic combinations exist instead of $2^{WN}$ under 100% loading. However, the $WN$-neuron network operates in the $WN$-dimensional hypercube defined by $V_i = 0$, or 1. The local minima of $E$ function may occur on the $2^{WN}$ corners of this space. The simplest way to restrict the motion of a neural network is to introduce inherent contention features between the packets in the same input queue regardless of their real behaviors. In other words, we take the input access contention of the packets in the same input port as their output contention.

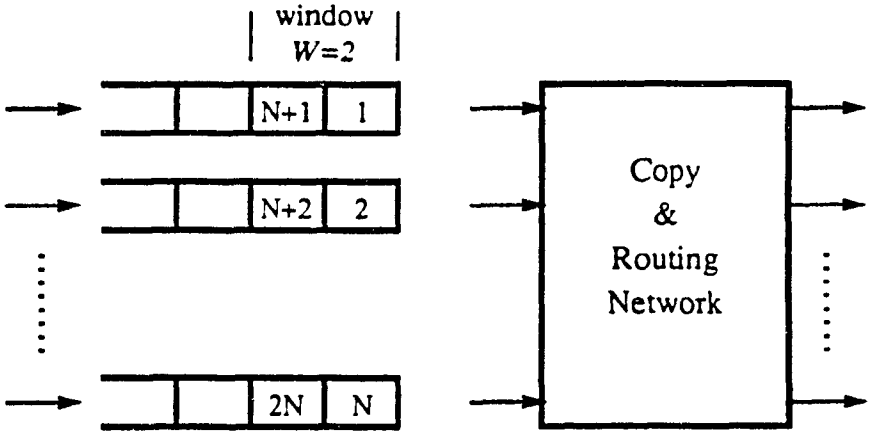



Figure 3.15. Input port queueing with windowed service

An example when $W = 2$ is shown in Figure 3.15. First two packets in each queue, hence totally $2N$ packets contend for access. Because only one could be chosen from each input queue, we must define two packets of the same port as an inherent contention pair. Thus, the $2N \times 2N$ contention matrix $D$ has the form of

$$D = \left[\begin{array}{cccc|cccc} 0 & & & & 1 & & & \\ & 0 & & & & 1 & & \\ & & \ddots & & & & \ddots & \\ & & & 0 & & & & 1 \\ \hline 1 & & & & 0 & & & \\ & 1 & & & & 0 & & \\ & & \ddots & & & & \ddots & \\ & & & 1 & & & & 0 \end{array}\right]$$

The unassigned elements here are decided by their real transmission requests as discussed in last section. Following the same steps, we construct the $E$ function

$$E = \frac{A}{2} \sum_{i=1}^{2N} \sum_{\substack{j=1 \\ j \neq i}}^{2N} V_i V_j D_{ij} + \frac{B}{2} \left( \sum_{i=1}^{2N} V_i - n \right)^2 + \frac{C}{2} \left( \sum_{i=1}^{2N} V_i F_i - n' \right)^2 \tag{3.10}$$

which reflects our contention resolution requirement. The equations of motion, representing the time evolution of this problem, are given by

$$\frac{dU_i}{dt} = -\frac{U_i}{\tau} - A \sum_{\substack{j=1 \\ j \neq i}}^{2N} D_{ij} V_j - B \left( \sum_{j=1}^{2N} V_j - n \right) - CF_i \left( \sum_{j=1}^{2N} F_j V_j - n' \right) \tag{3.11}$$

The simulations of this network were made with the same parameters as in the last section, especially, $n$ and $n'$ were still equal to number of input ports. The throughput performance for some switches with windowed queueing is shown in Figure 3.16. Compared with the FIFO case, we found the throughput is around 10% higher, varying with the switch sizes. But, the management of windowed queue service is a little more complicated than that of FIFO queue.

### 3.3.4. A Model for SS Call Splitting

The optimization problem for the SS call splitting is similar to the Traveling Salesm . Problem solved using neural networks [7]. A similar work for unicast switch can be found in [11].

Unlike the model we use for the one-shot discipline, the model of neural network for the SS call splitting discipline is a two-dimensional structure composed of $N^2$ neurons and each neuron is identified by a set of double indices $i$ and $j$ indicating its row and column respectively (Figure 3.17 is an example where $N$ = 4). The input-output relationship of neurons is given as

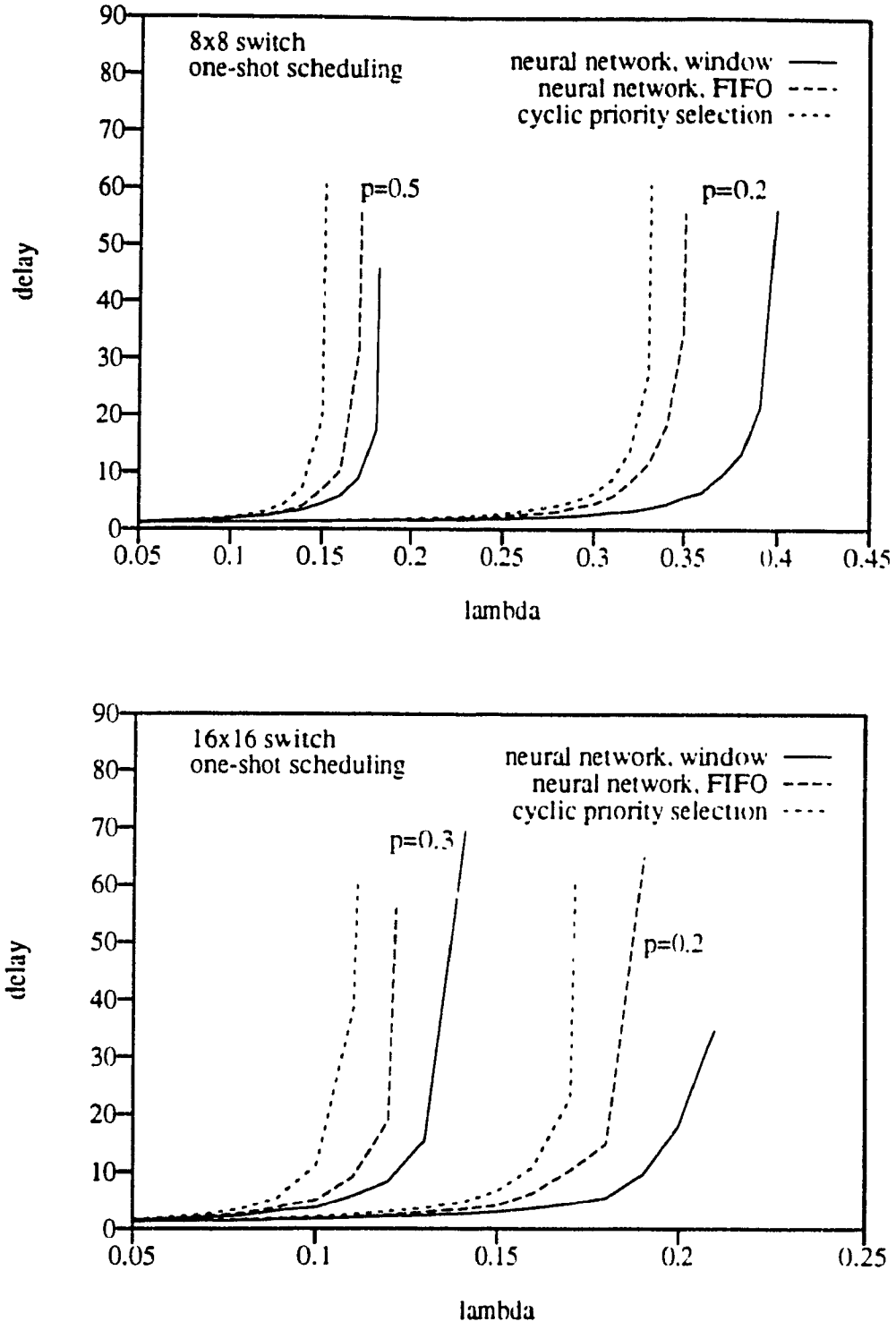



Figure 3.16. Improvement of windowed service (window size $W$=2)

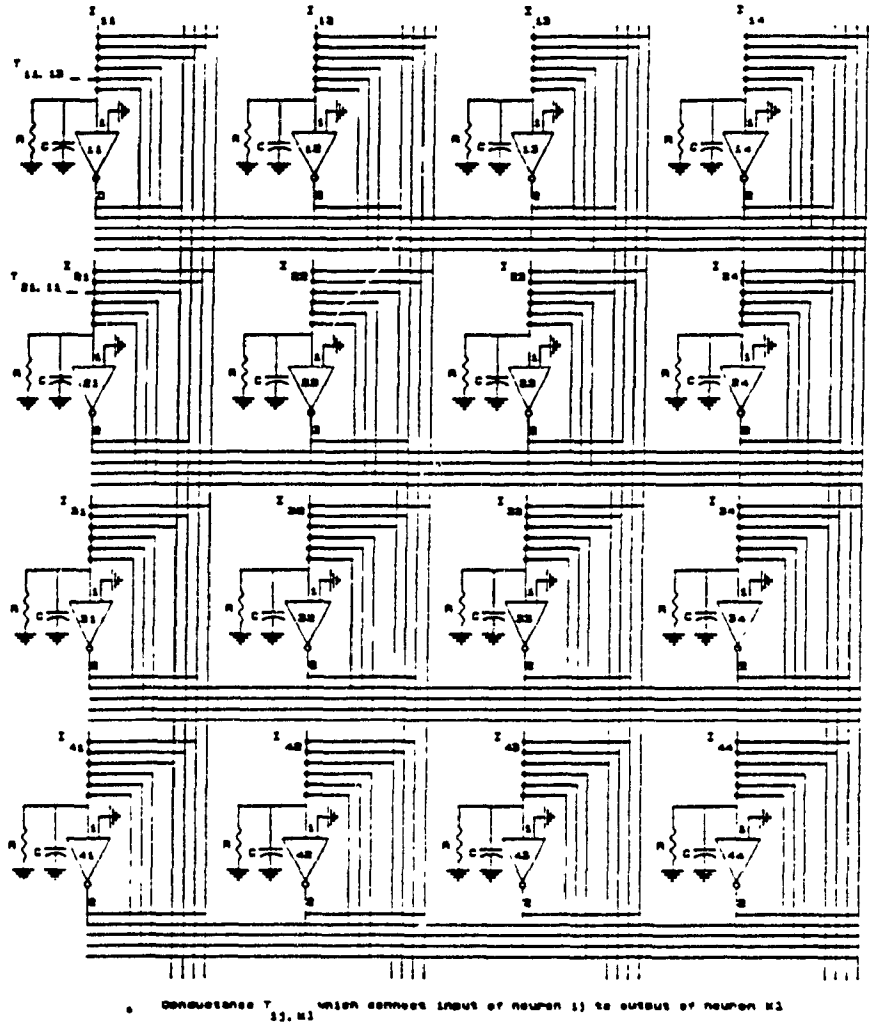


Figure 3.17. A two-dimensional structure of Hopfield neural network

$V_{ij} = g(U_{ij})$. The row subscript has the interpretation of the input address, and the column subscript the output address. There is a feedback path among pairs of neurons, designated as $T_{ij,kl}$ and referred to as connection matrix. Further, there is an external input biasing current $I_{ij}$ supplied to each neuron. The differential equation describing dynamics of neuron is given by

$$\frac{dU_{ij}}{dt} = -\frac{U_{ij}}{\tau} + \sum_{k=1}^{N}\sum_{l=1}^{N} T_{ij,kl}\, V_{kl} + I_{ij} \tag{3.12}$$

The corresponding energy function is defined as

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} T_{ij,kl}\, V_{ij}\, V_{kl} - \sum_{i=1}^{N}\sum_{j=1}^{N} V_{ij}\, I_{ij} \tag{3.13}$$

The local minima of this energy function occurs only on the $2^{N^2}$ corners of the hypercube defined by $V_{ij} = 0$, or 1.

The matrix $V = \{V_{ij}\}$ corresponds to the acceptable traffic combination, where $V_{ij} = 1$ denotes that a copy can be transmitted from input $i$ to output $j$. As explained before, any row or column of matrix $V$ can only has one 1 in final stable state. The neural network has to maximize the total number of copies chosen per slot under this switching constraints. Thus, the energy function of copy reservation discipline is given by

$$E = \frac{A}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{\substack{l=1\\ l\neq j}}^{N} V_{ij}V_{il} + \frac{B}{2}\sum_{j=1}^{N}\sum_{i=1}^{N}\sum_{\substack{k=1\\ k\neq i}}^{N} V_{ij}V_{kj} + \frac{C}{2}\left(\sum_{i=1}^{N}\sum_{j=1}^{N} V_{ij} - n\right)^2 \tag{3.14}$$

The first and second terms are minimized when a solution has at most a single nonzero element per row and column, with the requirement that at most single copy per input and output is chosen. Thus any solution satisfying this requirement is a feasible solution and meets the physical constraints of the switch fabric. The third term is minimized when the total weight of the solution (number of nonzero elements in the solution) is maximized. It was found that the solution always contains single nonzero element in every row or column even if the corresponding input row or column is all zeros. However large number of simulation showed that the neural network never (or with negligibly small probability) placed a nonzero in an output row or column at the expense of another input row or column with nonzero elements in it. Therefore, the correct output could be determined by comparing the output with input, and disregarding the nonzero output elements which were zero at the input. Thus, the elements of the connection matrix and external input biases are given by

$$T_{ij,kl} = -A\,\delta_{ik}\,(1-\delta_{jl}) - B\,\delta_{jl}\,(1-\delta_{ik}) - C \tag{3.15}$$

$$I_i = C\,n \tag{3.16}$$

Substituting equations (3.15) and (3.16) into (3.12) results in following differential equation describing the dynamics of the neurons

$$\frac{dU_{ij}}{dt} = -\frac{U_{ij}}{\tau} - A \sum_{\substack{l=1 \\ l \neq j}}^{N} V_{il} - B \sum_{\substack{k=1 \\ k \neq i}}^{N} V_{kj} - C \left( \sum_{k=1}^{N} \sum_{l=1}^{N} V_{kl} - n \right) \tag{3.17}$$

The simulation procedure follows the same steps as we discussed in Section 3.3.2 except that the initial input voltages $U_{ij}(0)$ are given the values of $+U_0$ or $-U_0$ depending on whether the corresponding elements in the input request matrix have values 1 or 0 respectively. After a number of trials runs, the following parameter values were found to give better results.

$$A = 80, \quad B = 80, \quad C = 30, \quad \text{and } n = N$$

The simulation of cyclic priority selection strategy was also made in order to evaluate the improvement of the optimum selection using neural networks. Both results are plotted together in Figure 3.18, which showed neural-network-based input access scheme always gives better performance than cyclic priority selection scheme.

## 3.4. CONCLUSION

Output contention resolution is an important issue in switch design. The multicast request complicates the output contention resolution algorithm and gives it a new meaning. In this chapter, we have discussed several implementation schemes and have shown the potential of improvement of delay-throughput performance by introducing optimal algorithm.

We have proposed a novel scheme to resolve the output contention by using combinational logic circuits and incorporating cyclic priority access concept. It features simple and high speed operation, in comparison with the other methods such as ring-reservation algorithm and three-phase algorithm. It is compact, reliable and growable.

We have also proposed a neural-network-based mechanism to resolve the output contention for a multicast packet switch. We formulate our contention resolution problem as an optimization problem. The use of an analog neural network ensures not only output contention-free transmission, but also higher packet throughput, and low delay.

It needs to be mentioned that the combinational logic scheme and the neural network scheme have similar lattice structure where logical decision units or neurons are fully connected.

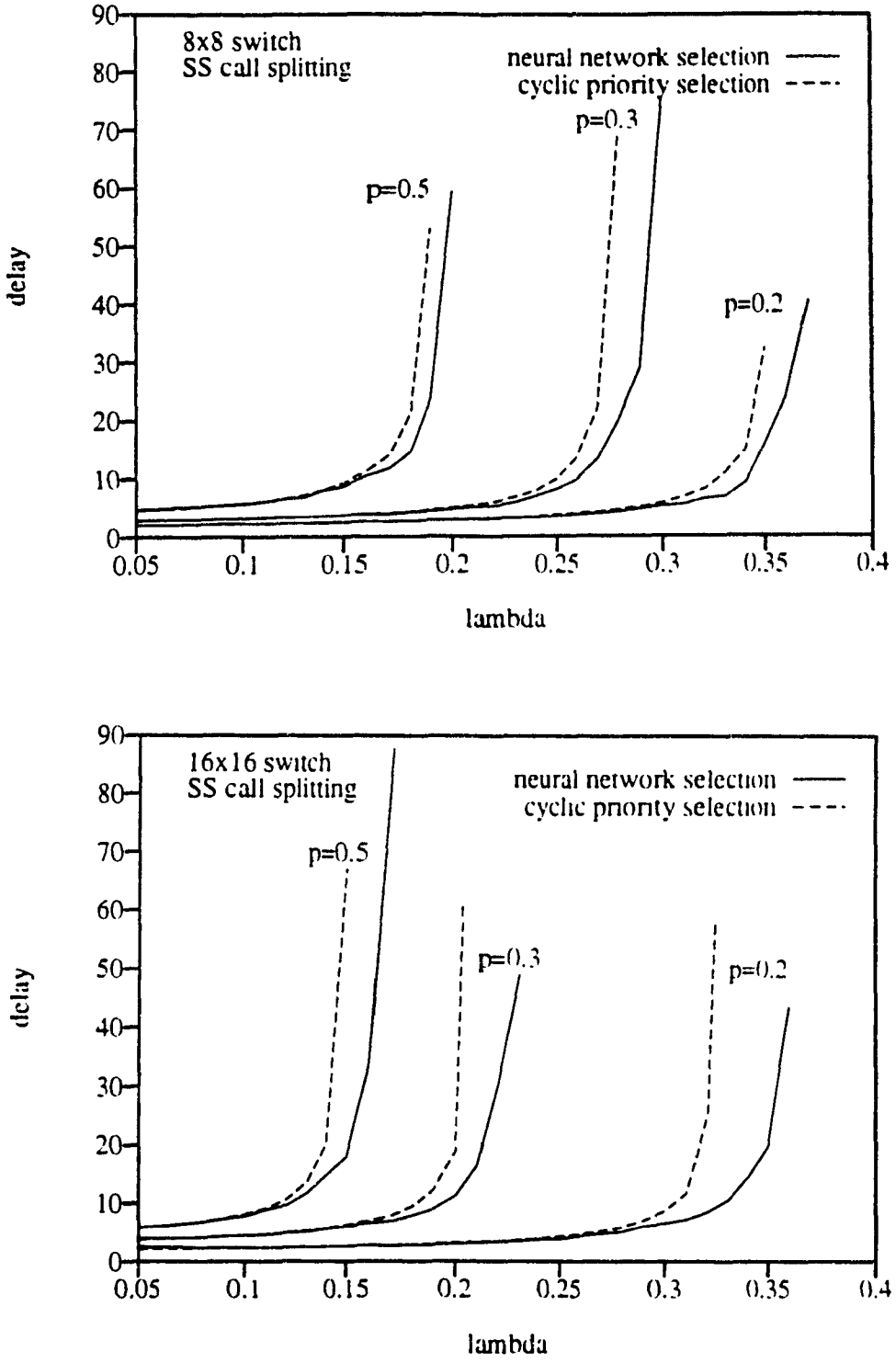



Figure 3.18. Improvement by neural network for SS call splitting

Generally speaking, the logic circuit is more easily implemented with VLSI technology than the analog (Hopfield model) neural network. On the other hand, neural network scheme provides better performance. Furthermore, it demonstrates a potential of optimal scheduling.

Actually, cyclic priority input access scheme is somewhat inefficient due to its determinacy. Although the neural-network-based scheme provides a certain improvement, it seems that the delay-throughput performance advantage is not so great as to overrule the simplicity of implementation of cyclic priority scheme.

The highly-interconnected networks of nonlinear analog neurons can rapidly solve difficult optimization problems. To map our contention resolution problem onto the general Hopfield model, we construct an energy function, in which the lowest energy state corresponds to the contention-free traffic combination and maximized throughput. With given initial states, connectivities between neurons and input biasing currents, related to a specific set of transmission requests, the analog system converges to the final stable state. From this final state, the solution to the problem is easily interpreted. The application of neural network in switching development is not confined with the contention resolution. It can be applied to routing and other optimal decision problems.

## REFERENCES 3

[1] J. Y. Hui, *Switching and Traffic Theory for Integrated Broadband Networks*, Kluwer Academic Publishers, Boston, 1990.

[2] A. Huang, S. Knauer, "Starlite: a wideband digital switch", Proc. of Globecom'84, pp.121-125, Atlanta, GA., Dec. 1984.

[3] J. Y. Hui, E. Arthurs, "A broadband packet switch for integrated transport", IEEE J. Select. Areas Comm., Vol.5, No.8, pp.1264-1273, Oct. 1987.

[4] B. Bingham, H. Bussey, "Reservation-based contention resolution mechanism for Batcher-banyan packet switches", Electronics Letters, 23rd, Vol.24, No.13, pp.772-773, June 1988.

[5] T. T. Lee, R. Boorstyn, E. Arthurs, "The architecture of a multicast broadband packet switch", Proc. IEEE Infocom'88, pp.1-8, 1988.

[6] M. Rocchi, *High-Speed Digital IC Technologies*, Artech House, Boston, 1990.

[7] J. J. Hopfield, D. W. Tank, "'Neural' computation of decisions in optimization problem", Biological Cybern., Vol.52, pp.141-152, 1985.

[8] T. Khanna, *Foundations of Neural Networks*, Addison-Wesley Publishing Company, 1990.

[9] M. G. Hluchyj, M. J. Karol, "Queueing in high performance packet switching", IEEE J. Select. Areas Comm., Vol.6, No.9, pp.1587-1597, Dec. 1988.

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

[11] M. K. Mehmet-Ali, H. T. Nguyen, "A neural network implementation of an input access scheme in a high-speed packet switch", Proc. of Globecom'89, pp.1192-1196, 1989.

# CHAPTER 4

## MODELING AND MATHEMATICAL ANALYSIS


4.1. INTRODUCTION

4.2. ANALYSIS OF ONE-SHOT DISCIPLINE WITH RANDOM SELECTION

4.2.1. Mathematical Model

4.2.2. Numerical Results

4.3. ANALYSIS OF ONE-SHOT DISCIPLINE WITH CYCLIC PRIORITY

4.3.1. Mathematical Model

4.3.2. Numerical Results

4.4. REVIEW OF TWO MODELS FOR WS CALL SPLITTING

4.4.1. Model I by J. F. Hayes et al.

4.4.2. Model II by J. Y. Hui et al.

4.5. A GENERAL UNIFIED MODEL FOR PERFORMANCE ANALYSIS

4.5.1. Description of the System

4.5.1.A). $Q$ Matrix for One-Shot Discipline

4.5.1.B). $Q$ Matrix for WS Call Splitting Discipline

4.5.2. Solution via Matrix Geometric Approach

4.6. REMARKS

APPENDIX 4A: Algorithm for the Formation of the $Q$ Matrix (One-Shot Scheduling)

APPENDIX 4B: Algorithm for the Formation of the $Q$ Matrix (WS Call Splitting)

REFERENCES 4

## 4.1. INTRODUCTION

In the previous two chapters, we have discussed various service disciplines and various selection policies concerning the input access scheme of the multicast packet switch. Computer simulations were carried out to compare the delay-throughput performance of the switch associated with various service disciplines and various selection policies. In this chapter, we focus on mathematical models suitable for performance analysis. Simulation results will be presented for verification purposes, in turn, mathematical analysis may provide more statistical measures which are hardly observed through simulations. This work can also be taken as an introductory traffic theory for the multicast packet switch. Finally, although this chapter is devoted to the multicast switch, the analytic model can also be applied to other queueing problems.

In what follows, input port queueing along with an input access scheme is assumed in order to resolve the output request conflict. The performance analysis of the SS call splitting discipline can be carried out by considering each copy of the multicast packet as an unicast packet, ignoring the batch arrival nature of these copies. We shall not discuss the exact analysis for this discipline. Furthermore, the revision scheduling is a combination of the one-shot scheduling and the WS call splitting disciplines which makes an analysis too difficult to carry out. Our interests will be focused on modeling and performance analysis of the one-shot scheduling and the WS call splitting (or pure work conservation) disciplines.

Before we discuss the details of the mathematical analysis of the switch, it would be of interest to the reader to be aware of the difficulties in carrying out the exact analysis. Hluchyj and Karol [1] reported that for a large size unicast switch, the saturated throughput is approximately 0.586. But, if we drop interfering packets at the end of each time slot, rather than storing them in the input queues, the saturated throughput increases up to 0.632. The reason for this phenomenon is that the blocked packets are not uniformly distributed over the set of output ports and tend to cluster on blocked output ports. Therefore, blocked packet will suffer another blocking with higher probability than the fresh packet. It seems impossible to distinguish between blocked packet and fresh packet by using different destination distribution model for the time

being. This residual effect is also present in multicast switching.

Another intrinsic problem in the analysis of multicast switch is the characterization of the distribution of the number of copies of the HOL packets. Among the HOL packets, the blocked packet has more copies than the newly incoming packet because the packet with less copies gets through in the one-shot case more easily. However, in the case of WS call splitting scheme, the blocked packet has less copies than the newly incoming packet. The difference varies with the packet arrival rate as shown in Figure 4.1, which was observed via simulations. This effect makes exact analysis very difficult.

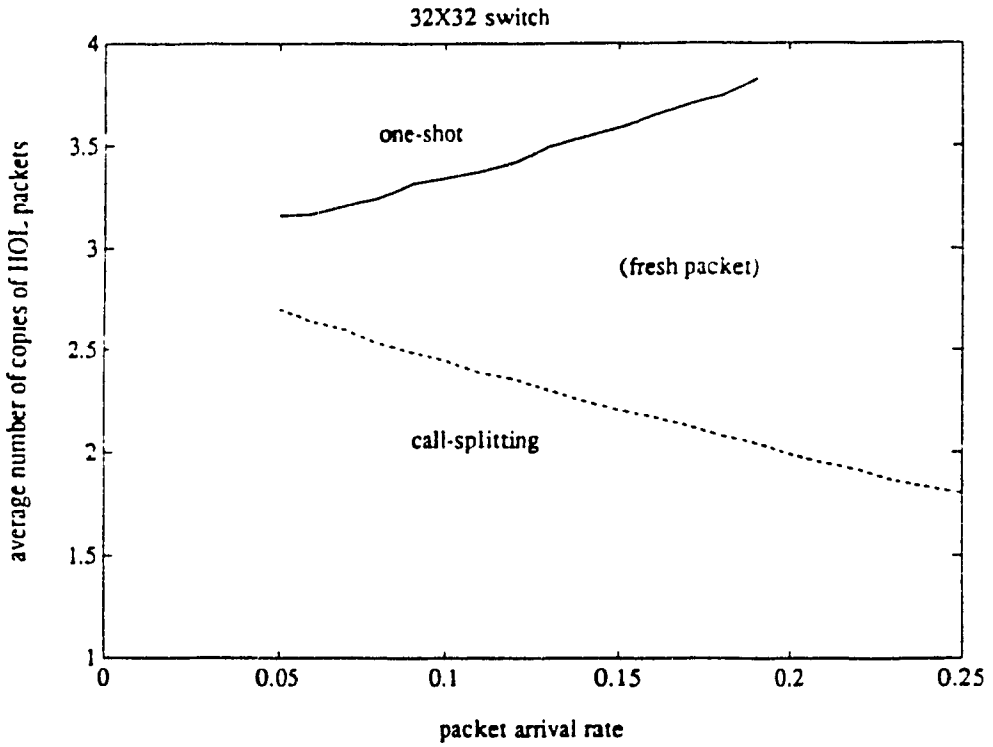



Figure 4.1. Average number of copies of the HOL packet vs. packet arrival rate

Despite these difficulties, the performance analysis for the multicasting disciplines has received much attention. Several approximations were introduced into the mathematical model to

describe the copy distribution model for both one-shot and WS call splitting cases, while the effect of HOL packets' cluster on blocked output ports was neglected for the sake of a tractable mathematical analysis. In our study, the copy distribution for one-shot scheduling is modified according to the encountered total life distribution. In the analysis for the WS call splitting discipline, Hayes [2] employs residual distribution and Hui [3] uses the independent service assumption by setting the probability that each HOL destination is served, as a function of copy's arrival rate only.

Given the above considerations, we were motivated to use matrix-geometric techniques to obtain more accurate results. In addition, a unified mathematical model was devised for the analysis of both the one-shot and the WS call splitting input access disciplines.

In our study, we assume input port queueing with infinite buffer size. Packet arrivals are modeled as Bernoulli process with average arrival rate of $\lambda$ per slot per input port. The packet delay is considered as a basic performance measure of the switch.

The remainder of the chapter is organized as follows. Section 4.2 gives the mathematical analysis of random selection policy for the one-shot discipline. Section 4.3 discusses the analysis of cyclic priority input access scheme for the one-shot discipline. For completeness, Section 4.4 summarizes two analytic models for the WS call splitting discipline reported by Hayes and Hui, respectively. In Section 4.5, we introduce a unified model based on matrix-geometric techniques for both the one-shot scheduling and the WS call splitting disciplines. Our conclusions are summarized in Section 4.6.

## 4.2. ANALYSIS OF ONE-SHOT DISCIPLINE WITH RANDOM SELECTION

It is the purpose of this section to examine the performance of the one-shot access scheme for a multicast switch. A random-selection policy is used to resolve the output conflict.

### 4.2.1. Mathematical Model

Leaving aside the details of the implementation, the one-shot discipline with a random-selection policy could be achieved in the following manner: the centralized controller polls the

HOL packets of all input queues in a random order, and issues a license to the packet in case it is contention-free with the previously licensed packets. Then all the tagged packets (including all their copies if applicable) are transmitted to the destinations during one time slot. Here, the output contention between two multicast packets is defined as at least one copy of one packet seeking the same destination as any one copy of the another packet does.

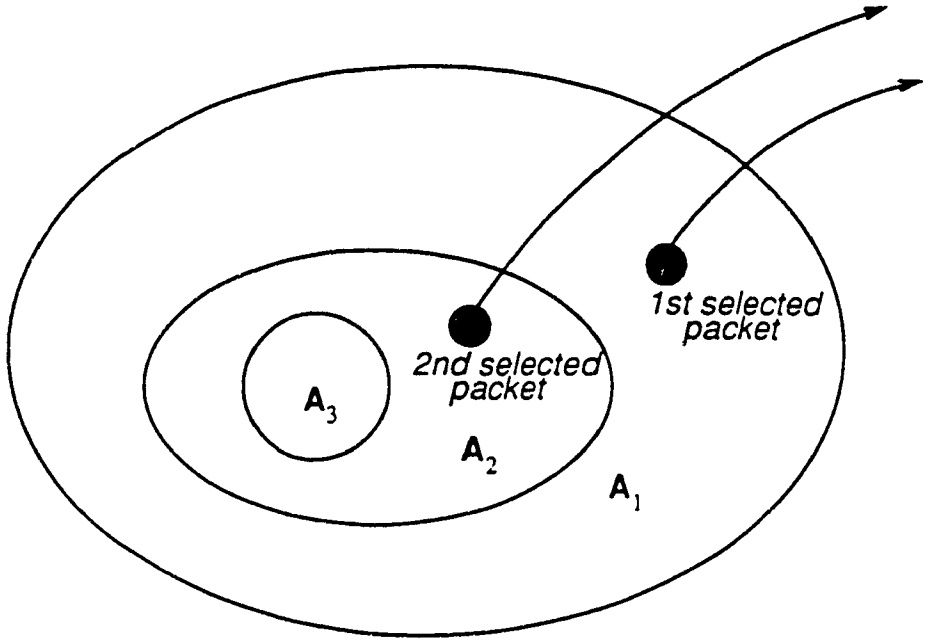



Figure 4.2. Multistage screening process

In carrying out the analysis, we may describe the process of the conflict resolution as follows (Figure 4.2). A single packet is selected randomly from all active input ports which constitute the entire space $A_1$. Within the remainder, there is a subset $A_2$ in which no one conflicts with the first selected packet. Then, the second packet can be picked out from the subset $A_2$ at random. Similarly, within the remainder of the subset $A_2$, there is a subset $A_3$ in which no one conflicts with the second selected packet as well as the first. This process operates as if multistage screening until the descendant subset is empty.

Let us assume that in an $N \times N$ self-routing, non-blocking multicast switch, an input port queue is active with probability $\rho$. Thus, the probability of $n$ ports being busy is given by

$$B_n = \binom{N}{n} \rho^n (1-\rho)^{N-n} \qquad n=0,1,2,.....,N \tag{4.1}$$

First of all, we randomly select an non-empty port for which no blocking is assumed in the current time slot. Then, we determine a subset $A_2$ for which no member port has contention with the first selected packet. The second port is selected from this subset. Let us assume that the first selected packet, which can be any one of the packets in the busy ports, has $l$ copies with probability $R_l$, and that the second considered packet has $m$ copies with probability $R_m$. As long as none of the $m$ copies, generated by the packet under consideration, intend to get access to any one of these $l$ reserved output ports, the packet falls into the subset $A_2$, with probability

$$\frac{\binom{N-l}{m}}{\binom{N}{m}} = \prod_{j=1}^{m} \left[ 1 - \frac{l}{N-j+1} \right]$$

Moreover, the sum of $l$ and $m$ should not exceed $N$. Averaging over $l$ and $m$, we have the probability that an arbitrarily chosen second packet does not contend with the first.

$$S_1 = \sum_{l=1}^{N} \left\{ R_l \sum_{m=1}^{N-l} R_m \prod_{j=1}^{m} \left[ 1 - \frac{l}{N-j+1} \right] \right\} \tag{4.2}$$

Given that $j$ packets remain after first selection, the probability that $i$ of them do not conflict with the first selected packet is given by

$$Pr(i \mid j) = \binom{j}{i} S_1^i (1-S_1)^{j-i} \qquad i \le j \tag{4.3}$$

Since $n$ ports are busy with probability $B_n$ given by equation (4.1), the probability of $j$ remainder packets after first selection is $B_{j+1}$. Removing the condition in equation (4.3) yields

$$\begin{aligned} U_{2i} &= Pr(i \text{ } nonconflicting \text{ } packets \text{ } on \text{ } step \text{ } 2) \\ &= B_0 \delta(i) + \sum_{j=i}^{N-1} \binom{j}{i} S_1^i (1-S_1)^{j-i} B_{j+1} \end{aligned} \tag{4.4}$$

$$i=0,1,2,.....N-1$$

where

$$\delta(i) = \begin{cases} 1 & i=0 \\ 0 & otherwise \end{cases}$$

We define $U_{kl}$ = $Pr$ ($i$ nonconflicting packets on step $k$). By letting $U_{1l}=B_l$, equation (4.4) becomes

$$U_{2i} = U_{10}\delta(i) + \sum_{j=i}^{N-1} \binom{j}{i} S_1^i (1-S_1)^{j-i} U_{1,j+1} \qquad (4.5)$$

$$i=0,1,2,.....,N-1$$

Equation (4.5) is the probability distribution of the subset $\mathbf{A}_2$ which does not conflict with the first selected packet. Clearly, any one in the subset $\mathbf{A}_2$ can be chosen as the second packet winning the contention. The complement of this subset will be discarded from consideration. This procedure continues similarly until the subset is empty. Let $n$ be the number of output ports reserved by the first selected packet with probability $R_n$, then the packets in subset $\mathbf{A}_2$ only have the copies destined for the remaining $N-n$ output ports. Thus, the contention-free subset $\mathbf{A}_3$ for the third selection contains $i$ packets with probability

$$U_{3i} = U_{20}\delta(i) + \sum_{j=i}^{N-2} \binom{j}{i} S_2^i (1-S_2)^{j-i} U_{2,j+1} \qquad (4.6)$$

$$i=0,1,2,.....,N-2$$

where $S_2$ is the probability of no contention between any two packets in subset $\mathbf{A}_2$, which can be easily obtained by inserting $N-n$ in place of $N$ in equation (4.2), and summing over the random variable $n$,

$$S_2 = \sum_{n=1}^{N} \left\{ p_{1n} \sum_{l=1}^{N-n} \left[ R_l \sum_{m=1}^{N-n-l} R_m \prod_{j=1}^{m} \left[ 1-\frac{l}{N-n-j+1} \right] \right] \right\} \qquad (4.7)$$

where $p_{1n} = R_n$ ($n$ = 1, 2, ... , $N$). In similar manner, the subset $\mathbf{A}_k$ contains $i$ packets with probability

$$U_{ki} = U_{k-1,0}\delta(i) + \sum_{j=i}^{N-k+1} \binom{j}{i} S_{k-1}^i (1-S_{k-1})^{j-i} U_{k-1,j+1} \qquad (4.8)$$

$$i=0,1,2,.....,N-k+1$$

and

$$S_k = \sum_{n=1}^{N} \left\{ p_{k-1,n} \sum_{l=1}^{N-n} \left[ R_l \sum_{m=1}^{N-n-l} R_m \prod_{j=1}^{m} \left[ 1-\frac{l}{N-n-j+1} \right] \right] \right\} \qquad (4.9)$$

where $p_{k-1,n}$ is the probability that $n$ output ports have been reserved by first $k-1$ contention-free

packets and is obtained by

$$p_{kn} = \sum_{j=1}^{n-1} p_{k-1,n-j} R_j \tag{4.10}$$

Let $W_k$ denote the probability that $k$ and only $k$ packets may be selected to gain access to the switch without contention in a time slot, which can be interpreted as the probability that the space sets $A_1, A_2, \cdots, A_k$ are non-empty and $A_{k+1}$ is empty. Thus,

$$W_k = U_{k+1,0} \prod_{l=1}^{k} ( 1 - U_{l0} ) \qquad 1 \le k \le N \tag{4.11}$$

where $U_{N+1,0} = 1$. The throughput per input under a certain load $\rho$ is given by

$$T = \frac{1}{N} \sum_{k=1}^{N} k W_k = \frac{1}{N} \sum_{k=1}^{N} k U_{k+1,0} \prod_{l=1}^{k} ( 1 - U_{l0} ) \tag{4.12}$$

The average probability of a packet to get through at a specific busy port is the ratio of the average number of packets getting through, $NT$, and the average number of busy input ports, $N\rho$, i.e.,

$$T_b = \frac{T}{\rho} \tag{4.13}$$

In order to calculate the average packet delay, we have to determine the average and the mean square service time which is the period when a packet stays at HOL and contends for output ports slot by slot. It is assumed that these trials are independent of each other and can be described as a Bernoulli trial model. Thus, the average and the mean square service time are given by

$$\overline{M} = \sum_{n=1}^{\infty} n T_b [ 1 - T_b ]^{n-1} = \frac{1}{T_b} \tag{4.14}$$

and

$$\overline{M^2} = \sum_{n=1}^{\infty} n^2 T_b [ 1 - T_b ]^{n-1} = \frac{2 - T_b}{[ T_b ]^2} \tag{4.15}$$

These may be substituted into the following formula [1] to find the average delay $\overline{D}$.

$$\overline{D} = \overline{M} + \frac{\lambda (\overline{M^2} - \overline{M})}{2 ( 1 - \lambda \overline{M} )} \tag{4.16}$$

Here, $\lambda$ is Bernoulli arrival rate of the packets and is given by $\rho / \overline{M}$.

### 4.2.2. Numerical Results

We assume that each incoming packet generates copies independently with identical distributions. This distribution can be described as Bernoulli trials on each of the output ports in which the probability of generating a copy on each trial is $p$. In practice, however, at least one copy is always generated by every packet. So, a modified Binomial distribution is used to represent the distribution of the number of copies generated by each packet, i.e.

$$Q_m = \frac{\binom{N}{m} p^m (1-p)^{N-m}}{1-(1-p)^N} \qquad m=1,2,\cdots,N \tag{4.17}$$

Under the one-shot scheduling, the packet with less copies will more easily win the contention than that with more copies; accordingly, the blocked packet has more copies than new coming packet on the average. Thus, a blocked packet will suffer another blocking with higher probability than the initial blocking probability of a new packet. We attempt to model this effect by replacing the distribution in equation (4.17) by its encountered total life distribution [4][5]. Thus, the density associated with the HOL packets is given in terms of the density of typical packets by

$$R_m = \frac{m\, Q_m}{\bar{m}} \tag{4.18}$$

Here, $\bar{m}$ is the common average number of copies generated by a typical packet. Under the assumption of the modified Binomial distribution, we have

$$\bar{m} = \frac{N p}{1-(1-p)^N} \tag{4.19}$$

Substituting (4.17) and (4.19) into (4.18), we find

$$Q_m = \frac{m \binom{N}{m} p^m (1-p)^{N-m}}{N p} \qquad m=1,2,\cdots,N \tag{4.20}$$

Figure 4.3 shows the analysis results of average packet delay versus average packet arrival rate. It is interesting to note that when $N \to \infty$ and keeping a constant $\bar{m}$ the average number of copies generated by a packet, the curve of delay will converge to an asymptote. This is not surprising since it approaches unicast mode when $\bar{m} \ll N$, and saturated throughput approaches $0.586/\bar{m}$, where 0.586 is the saturated throughput of the unicast switch under the same

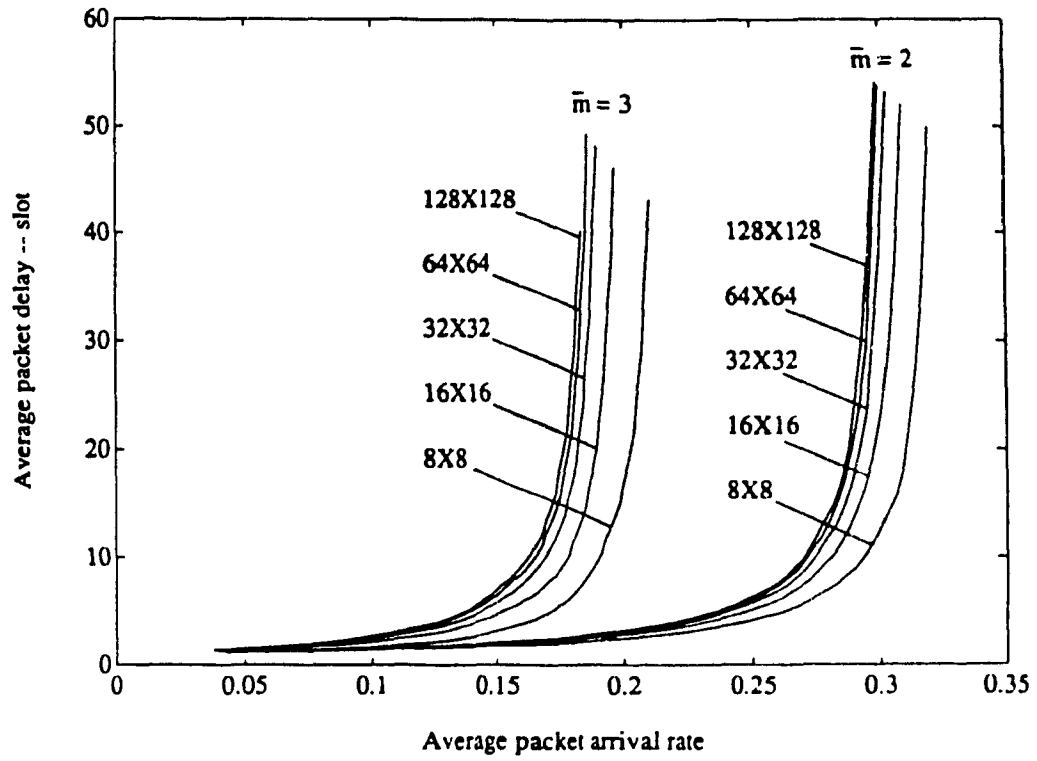
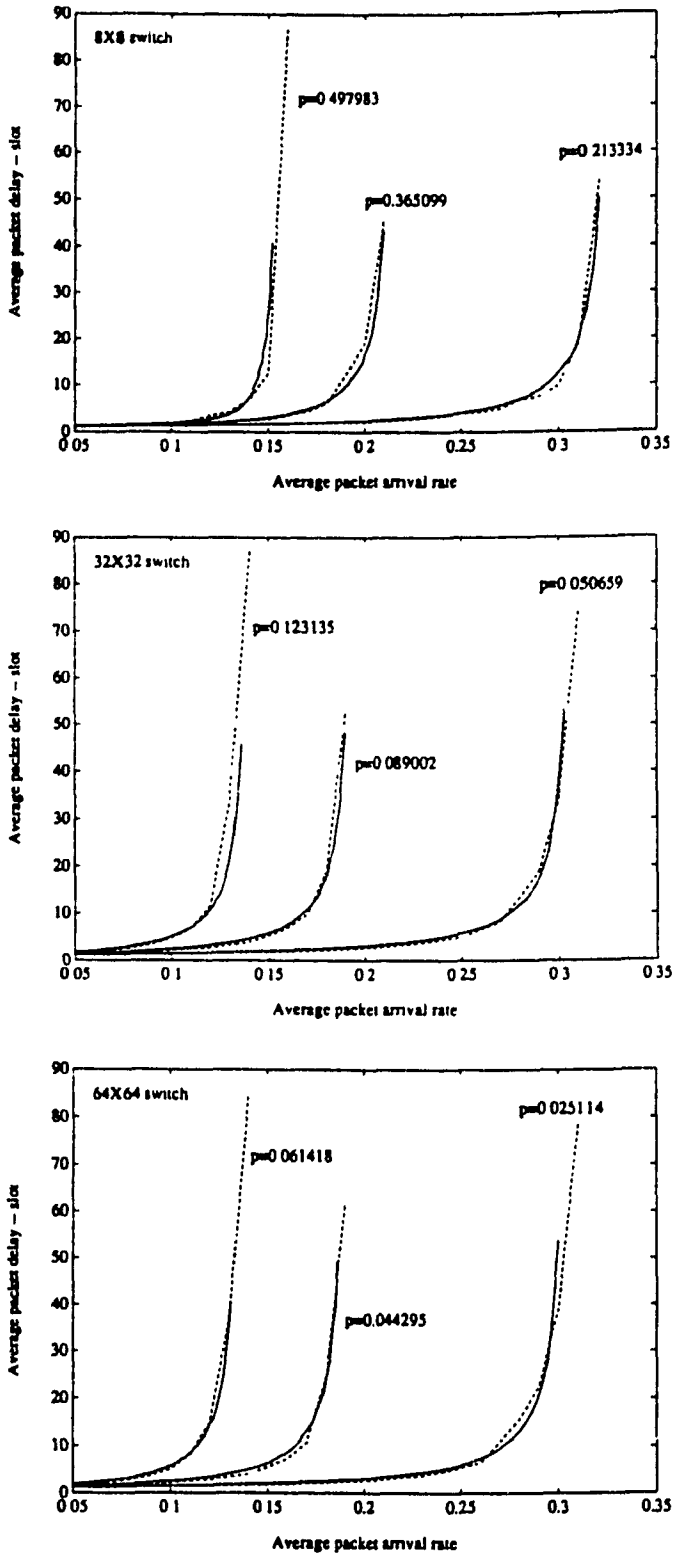



Figure 4.3. Average packet delay for several switch sizes with constant $\bar{m}$ (2 and 3)

assumptions.

In order to test the accuracy of the analysis, the results of the analysis were compared to the simulation results for the case of Geom/G/1 queueing system with infinite buffers. The simulation model was described in Chapter 2.

The results of the analysis and the simulation are shown in Figure 4.4 for various switch configurations. The average packet delay is shown as a function of the average packet arrival rate $\lambda$, with different values of $p$ the probability of a copy being generated at an output port. The comparison of analysis and simulation shows that the correspondence is quite satisfactory.




solid line ——— analysis results
dashed line - - - - simulation results

Figure 4.4. Average packet delay with one-shot scheduling and random-selection policy

## 4.3. ANALYSIS OF ONE-SHOT DISCIPLINE WITH CYCLIC PRIORITY

Unlike the random selection scheme, in this section, we assume that the HOL packets make the reservations of the output ports in the order of their priority levels. We assign $N$ levels of priority to the $N$ input ports in the beginning of each time slot and rotate the assignment cyclically slot by slot. The input port with priority level 1 can get its HOL packet or all of the copies through the switch with probability 1 if its queue is non-empty. As for second priority port, we will check whether all of the copies can get through without contention with the copies from the first priority port; if so, this packet can get through, otherwise it has to wait for next slot. Then we check each port according to the priority order up to the port with priority level $N$.

### 4.3.1. Mathematical Model

The notation used in the following analysis is defined as follows:

$S_i$ event that the packet in the input port with priority level $i$ wins the contention

$F_i$ event that the input port with priority level $i$ is blocked

$Pr(F_i) = 1 - Pr(S_i)$

$C_i$ random variable representing the number of copies generated by the packet at the head of input port with priority level $i$

$R_m$ probability distribution of the number of copies generated by a packet

$Pr( C_i = m ) = R_m \qquad m=1,2,.....,N$

$K_i$ random variable representing the number of output ports reserved by packets from the input ports with the priority level $j=1,2,.....,i$.

$Pr( K_i = k_i ) = P^i_{k_i} \qquad i=1,2,.....,N;\ k_i=0,1,.....,N$

First, we calculate the probability of getting through by conditioning on the number of copies of the packets and previous reservations. In the input port with priority level 1, we have

$$Pr(S_1) = 1 \tag{4.21a}$$

$$Pr(F_1) = 0 \tag{4.21b}$$

As for input port with priority level 2, $K_1$ output ports have been reserved by the input port with priority level 1. Because the generated copies are for different output ports, as long as any one of $C_2$ copies generated by the packet does not intend to get access to any one of those $K_1$ reserved output ports, the packet is able to get through. Moreover, the sum of $K_1$ and $C_2$ should not exceed $N$. We can now write down the conditional probability

$$Pr(S_2 | K_1=k_1, C_2=m, K_1+C_2 \le N) = \frac{\binom{N-k_1}{m}}{\binom{N}{m}}$$

$$= \frac{(N-k_1)(N-k_1-1)\cdots(N-k_1-m+1)}{N(N-1)\cdots(N-m-1)}$$

$$= \prod_{j=1}^{m}\left(1-\frac{k_1}{N-j+1}\right) \tag{4.22}$$

Generally, in the input port with priority level $i$ ($i$=2, 3....., $N$), we have

$$Pr(S_i | K_{i-1}=k_{i-1}, C_i=m, K_{i-1}+C_i \le N) = \prod_{j=1}^{m}\left(1-\frac{k_{i-1}}{N-j+1}\right) \tag{4.23}$$

Averaging over $C_i$ under the restriction of $K_{i-1} + C_i \le N$ yields

$$Pr(S_i | K_{i-1}=k_{i-1}) = \sum_{m=1}^{N-k_{i-1}} R_m \, Pr(S_i | K_{i-1}=k_{i-1}, C_i=m, K_{i-1}+C_i \le N)$$

$$= \sum_{m=1}^{N-k_{i-1}}\left\{R_m \prod_{j=1}^{m}\left(1-\frac{k_{i-1}}{N-j+1}\right)\right\} \quad i>1 \tag{4.24}$$

To remove the condition we find $P_{k_i}^{i} = Pr(K_i=k_i)$. Assume $\rho_i$ is the non-empty probability for the specific port with priority level $i$ in current time slot. All the copies from the port with priority level 1, as long as the port is non-empty, will get served immediately. Therefore,

$$P_{k_1}^{1} = R_{k_1}\,\rho_1 \qquad k_1 = 1,2,.....,N \tag{4.25}$$

In case that the port is empty, $k_1 = 0$ and

$$P_0^1 = 1 - \rho_1 \tag{4.26}$$

We now compute the conditional probability of the number of reservations due to port with priority level 2, given the reservation due to port with the highest priority.

$$Pr(K_2 = k_2 | K_1 = k_1) = \delta_{k_2,k_1} [\, \rho_2 \, Pr(F_2 | K_1 = k_1) + (1 - \rho_2) \,] + \rho_2 \, R_{k_2 - k_1} \, Pr(S_2 | K_1 = k_1) \tag{4.27}$$

where

$$\delta_{i,j} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

The first term of (4.27) corresponds to the case $k_2 = k_1$ which could happen when second reservation fails due to its contending with the first reserved packet or simply when the considered port is empty. The second term describes the fact that a packet with $k_2 - k_1$ copies gets through. Removing the conditioning on $K_1$, we write

$$\begin{aligned} P_{k_2}^2 &= \sum_{k_1=0}^{k_2} P_{k_1}^1 \, Pr(K_2 = k_2 | K_1 = k_1) \\ &= \sum_{k_1=0}^{k_2} P_{k_1}^1 \left\{ \delta_{k_2,k_1} [\rho_2 Pr(F_2 | K_1 = k_1) + (1 - \rho_2)] + \rho_2 R_{k_2 - k_1} Pr(S_2 | K_1 = k_1) \right\} \\ &= P_{k_2}^1 [\rho_2 Pr(F_2 | K_1 = k_2) + (1 - \rho_2)] + \sum_{k_1=0}^{k_2} P_{k_1}^1 \, \rho_2 \, R_{k_2 - k_1} Pr(S_2 | K_1 = k_1) \end{aligned} \tag{4.28}$$

Generally, we can show that

$$Pr(K_i = k_i | K_{i-1} = k_{i-1}) = \delta_{k_i,k_{i-1}} [\, \rho_i \, Pr(F_i | K_{i-1} = k_{i-1}) + (1 - \rho_i) \,] + \rho_i \, R_{k_i - k_{i-1}} \, Pr(S_i | K_{i-1} = k_{i-1}) \tag{4.29}$$

$$\begin{aligned} P_{k_i}^i &= \sum_{k_{i-1}=0}^{k_i} P_{k_{i-1}}^{i-1} \, Pr(K_i = k_i | K_{i-1} = k_{i-1}) \\ &= P_{k_i}^{i-1} [\, \rho_i \, Pr(F_i | K_{i-1} = k_i) + (1 - \rho_i) \,] \\ &\quad + \sum_{k_{i-1}=0}^{k_i} P_{k_{i-1}}^{i-1} \, \rho_i \, R_{k_i - k_{i-1}} \, Pr(S_i | K_{i-1} = k_{i-1}) \\ &= P_{k_i}^{i-1} \left\{ 1 - \rho_i \sum_{m=1}^{N-k_i} \left[ R_m \prod_{j=1}^{m} \left[ 1 - \frac{k_i}{N - j + 1} \right] \right] \right\} \\ &\quad + \sum_{k_{i-1}=0}^{k_i} \left\{ P_{k_{i-1}}^{i-1} \, \rho_i \, R_{k_i - k_{i-1}} \sum_{m=1}^{N-k_{i-1}} \left[ R_m \prod_{j=1}^{m} \left[ 1 - \frac{k_{i-1}}{N - j + 1} \right] \right] \right\} \end{aligned} \tag{4.30}$$

By removing the conditioning in equation (4.24) over $K_{i-1}$, we can obtain the probability of successful transmission for port with priority level $i$ ($2 \le i \le N$), namely,

$$Pr(S_i) = \sum_{k_{i-1}=0}^{N} P_{k_{i-1}}^{i-1} \, Pr(S_i \mid K_{i-1} = k_{i-1})$$
$$= \sum_{k_{i-1}=0}^{N} P_{k_{i-1}}^{i-1} \left\{ \sum_{m=1}^{N-k_{i-1}} \left[ R_m \prod_{j=1}^{m} \left[ 1 - \frac{k_{i-1}}{N-j+1} \right] \right] \right\} \tag{4.31}$$

When $i = 1$

$$Pr(S_1) = 1 \tag{4.32}$$

We now proceed to derive the probability of the number of slots required to switch a multicast packet. First, we calculate the probability by conditioning on the priority when packet arrives to the head-of-line. It is assumed that if not all the copies of the packet can get through in a slot, there is a retry for all copies in the next slot. We assume independence from trial to trial. If a new incoming packet receives the service with priority level $i$, the probability that only one slot is needed to get it through is $Pr(S_i)$, the probability that two slots are taken to transmit it is $Pr(S_{i-1})\,[1-Pr(S_i)]$, and so on. Thus, the probability that $n$ slots are required to switch a packet starting with service priority level $i$ is given by

$$Pr(n \mid starting\ at\ i,\ n \le i) = Pr(S_{i-n+1}) \prod_{j=i-n+2}^{i} [\,1 - Pr(S_j)\,] \tag{4.33}$$

with the mean value

$$\overline{M}_i = \sum_{n=1}^{i} n\, Pr(n \mid starting\ at\ i,\ n \le i) \tag{4.34}$$

Here, we define

$$Pr(starting\ at\ i) = Pr(port\ has\ priority\ level\ i \mid newly\ contending\ packet)$$

Note that if the port happens to have the highest priority, the packet will undoubtedly get through. So, at most $i$ slots are needed for transmission at any port. We now turn our attention to find the probability that the port to which a newly generated packet arrives has priority level $i$. Without loss of generality, we focus on a specific input port. The new HOL packet always follows after successful transmission in the previous slot or after a new arrival to an empty queue.

From the law of total probability, we have

Pr(newly contending packet | port has priority level $i$)

= Pr(in previous slot, port was busy, more than one packet existed in queue, and a packet got through | port has priority level $i$)

+ Pr(in previous slot, port was busy, only one packet existed in queue, a packet got through, and one packet arrived the port | port has priority level $i$)

+ Pr(in previous slot, port was empty, and one packet arrived the port | port has priority level $i$)

As previously, we assume that the probability of the port with priority level $i$ to be empty is $1-\rho_i$. We need to find a reasonable approximation for the probabilities of one message and of more than one message in this queue. To overcome this problem, we may assume that the proportion for these two probabilities is the same as that for the M/M/1 queue. Since all of the probabilities add to one, this gives $\rho_i(1-\rho_i)$ and $\rho_i^2$ for the probabilities of one message and more than one messages, respectively. Thus, for $i$=1, 2, ....., $N$-1

$$Pr(\textit{newly contending packet} \mid \textit{port has priority level } i) = \rho_{i+1}^2 Pr(S_{i+1}) + \lambda(1-\rho_{i+1})[1+\rho_{i+1} Pr(S_{i+1})] \quad (4.35a)$$

Since $Pr(S_1)=1$, then

$$Pr(\textit{newly contending packet} \mid \textit{port has priority level } N) = \lambda + \rho_1^2(1-\lambda) \quad (4.35b)$$

Where $\lambda$ is Poisson arrival rate of the packets per input per slot time. Because each port is served with cyclic priority, then

$$Pr(\textit{port has priority level } i) = \frac{1}{N} \quad (4.36)$$

By using Bayes theorem, we obtain

$$Pr(\textit{starting at } i) = Pr(\textit{port has priority level } i \mid \textit{newly contending packet}) = \frac{Pr(\textit{newly contending packet} \mid \textit{port has priority level } i)/N}{\sum_{j=1}^{N} Pr(\textit{newly contending packet} \mid \textit{port has priority level } j)/N}$$

Thus, for $i$=1, 2, ..... $N$-1

$$Pr(starting\ at\ i) = \frac{\rho_{i+1}^2\ Pr(S_{i+1}) + \lambda\,(1-\rho_{i+1})\,[1+\rho_{i+1}\ Pr(S_{i+1})]}{\sum_{j=1}^{N}[\rho_{j+1}^2 Pr(S_{j+1})+\lambda(1-\rho_{j+1})(1+\rho_{j+1}Pr(S_{j+1}))]} \tag{4.37a}$$

and for $i = N$

$$Pr(starting\ at\ N) = \frac{\lambda + \rho_1^2\,(1-\lambda)}{\sum_{j=1}^{N}[\rho_{j+1}^2 Pr(S_{j+1})+\lambda(1-\rho_{j+1})(1+\rho_{j+1}Pr(S_{j+1}))]} \tag{4.37b}$$

By averaging equation (4.34) over $i$, we obtain the average service time

$$\overline{M} = \sum_{i=1}^{N}\left[Pr(starting\ at\ i)\sum_{n=1}^{i} n\ Pr(n \mid starting\ at\ i,\ n\le i)\right] \tag{4.38}$$

The quantities of $\rho_i$ depend on the current priority and $\lambda$. We may write an equation as

Pr(empty | port has priority level $i$)

= Pr(no arrival during a slot) Pr(empty at beginning of previous slot)

+ Pr(no arrival during a slot) Pr(only one at beginning of previous slot)

× Pr (get through in previous slot)

Similarly to the above discussion, an approximation of the input queue by an M/M/1 model is assumed. Therefore,

$$1-\rho_i = (1-\lambda)(1-\rho_{i+1}) + (1-\lambda)\rho_{i+1}(1-\rho_{i+1})\,Pr(S_{i+1}) \tag{4.39}$$

By manipulating this equation, we can show

$$\rho_i' = \lambda + (1-\lambda)[1-Pr(S_{i+1})]\rho_{i+1}' + (1-\lambda)\,Pr(S_{i+1})\,\rho_{i+1}'^2 \tag{4.40}$$

In the case of the port with priority level $N$, since the previous priority was level 1 and $Pr(S_1)$=1, we have

$$\rho_N' = 1-\lambda+(1-\lambda)\rho_1'^2 \tag{4.41}$$

The average value then is given by

$$\rho' = \frac{1}{N}\sum_{i=1}^{N}\rho_i' \tag{4.42}$$

The load, $\rho$, can also be defined as the average number of arrivals during the time a message is being transmitted or

$$\rho = \lambda \overline{M} \tag{4.43}$$

The modified values of $\rho_i$ are found from normalization

$$\rho_i = \rho_i' \frac{\rho}{\rho'} \tag{4.44}$$

In order to calculate the average packet delay, we have to have the mean square service time which can be obtained by substituting $n^2$ for $n$ in equation (4.38), i.e.

$$\overline{M}^2 = \sum_{i=1}^{N} \left\{ Pr(starting\ at\ i) \sum_{n=1}^{i} [\, n^2\, Pr(S_{i-n+1}) \prod_{j=i-n+2}^{i} (1 - Pr(S_j)) \,] \right\} \tag{4.45}$$

The average delay can be obtained by [1]

$$\overline{D} = \overline{M} + \frac{\lambda (\overline{M^2} - \overline{M})}{2(1-\rho)} \tag{4.46}$$

Another quantity of interest is the throughput, which is the average number of packets getting through per input port per slot. Thus,

$$Throughput = \frac{1}{N} \sum_{i=1}^{N} Pr(S_i)\, \rho_i \tag{4.47}$$

### 4.3.2. Numerical Results

We solve our problems by using an iterative procedure based on the foregoing analysis as follows. Assuming that a steady-state solution exists, we suggest a simple procedure to find $Pr(S_i)$. We initialize $\rho_i$ with the values $0 < \rho_i < 1$ and find $P_{k,}^{i}$ from equations (4.25), (4.26) and (4.30). Substituting $P_{k,}^{i}$ into equation (4.31) yields the approximation of $Pr(S_i)$; which then can be used to update $\rho_i$ through equations (4.40), (4.41), (4.42), (4.43) and (4.44). We repeat this process and the new values of $Pr(S_i)$ are compared to the previous ones. The steady-state values of $Pr(S_i)$ are assumed to be obtained if no two consecutive values differ by more than a threshold of $10^{-6}$. It was observed that smaller values of the threshold do not improve the results but increase the iterating time.

In order to test the accuracy of the model, the results of the analysis were compared to the simulation of the system for the case of Geom/G/1 queueing model with infinite buffers. Two copy distribution models were used in our verification.

We first assume that each incoming packet generates copies independently with identical distributions. This distribution can be described as Bernoulli trials on each of the output ports in which the probability of generating a copy on each tria. is $p$. The same modification as in Section 4.2.3 is made, so that

$$R_m = \frac{m \binom{N}{m} p^m (1-p)^{N-m}}{N p} \qquad m=1,2,.....,N \tag{4.48}$$

This approximation was applied to the forgoing analysis to calculate the throughput and the delay.

The results from our analysis and from simulations are shown in Figure 4.5 and Figure 4.6. The average packet delay is shown as a function of the packet arrival rate, and the throughput as a function of the probability that the input port is busy, with different values of $p$ the probability of a copy being generated at an output port. The comparison of analysis and simulation shows that the correspondence is quite satisfactory; however, the match is better for copy distributions with large mean.

It is clear that the packet with larger average number of copies will suffer blocking more often. The approximation described in equation (4.48) is a proper model in this regard and gives the results close to the simulations especially for larger values of $p$. When $p$ is small, the average number of copies generated by a packet is small and approaches 1 while the switching approaches the unicast case. Therefore, the approximation is somewhat overdone by the encountered total life distribution in this case. Figures 4.5 and 4.6 show the better correspondence with larger values of $p$.

Another model applied to verify our analysis is to simply presume that a packet generates and evenly distributes $NC$ (a constant number of) copies to the destinations so that we do not need the total life approximation any more. Accordingly,

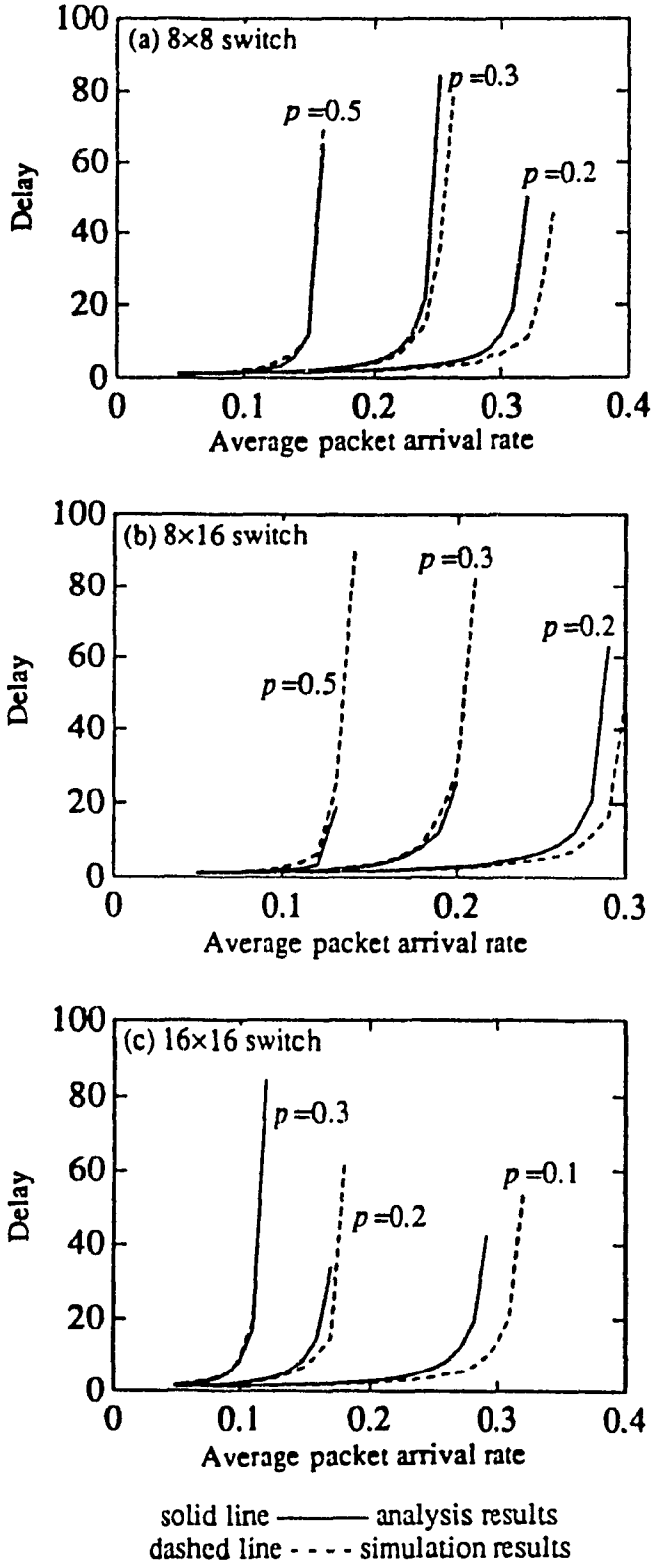



Figure 4.5. Average packet delay with one-shot scheduling and cyclic priority policy where each incoming packet generates copies according to Bernoulli trials with parameter $p$ for each output port

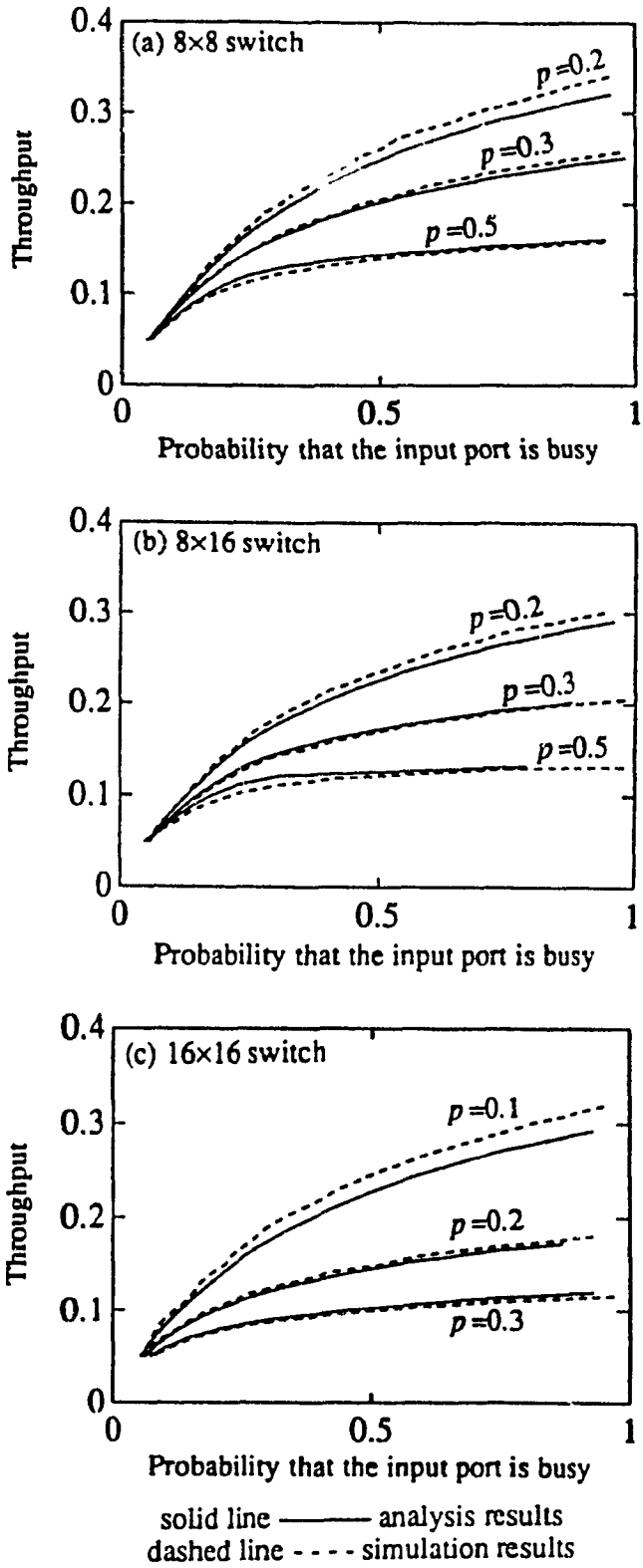



Figure 4.6. Average throughput with one-shot scheduling and cyclic priority policy where each incoming packet generates copies according to Bernoulli trials with parameter $p$ for each output port

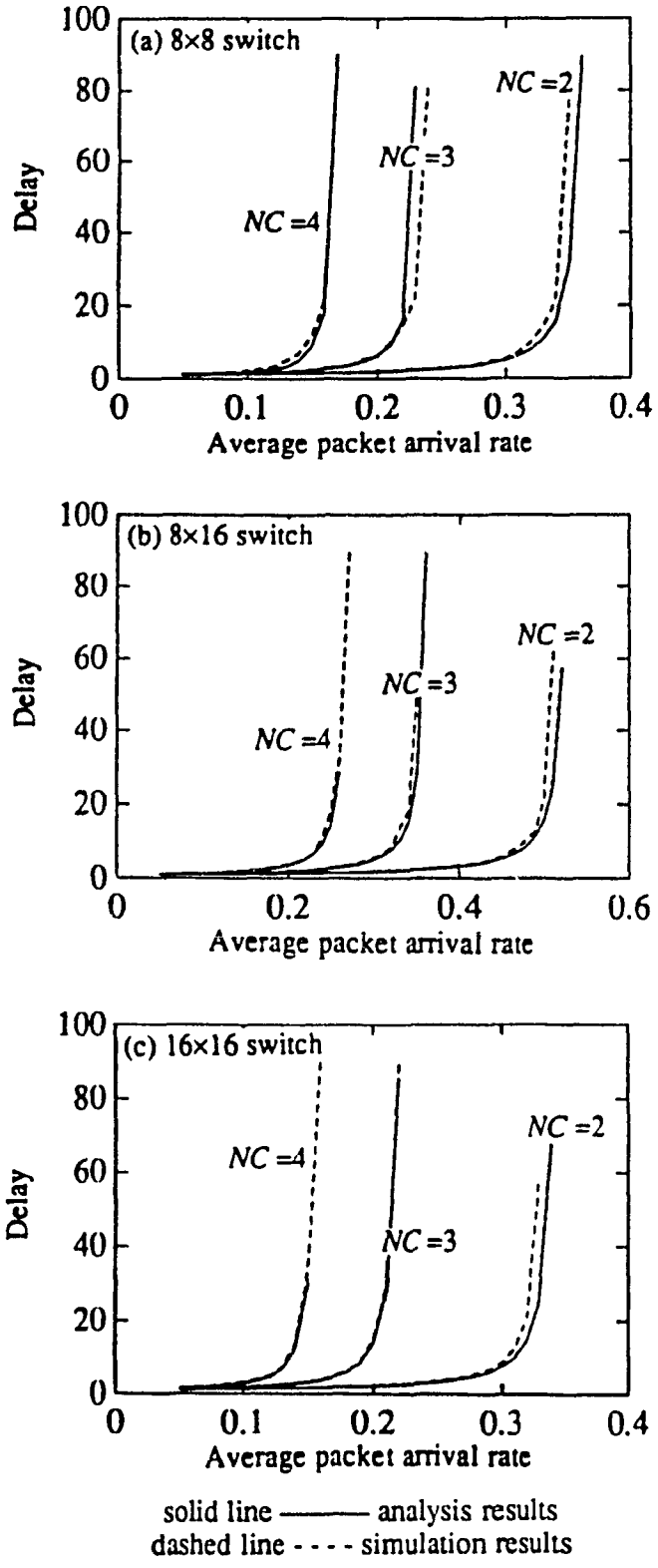



Figure 4.7. Average packet delay with one-shot scheduling and cyclic priority policy where each incoming packet generates *NC* (a constant number of) copies to a set of output ports

$$R_m = \begin{cases} 1 & m = NC \\ 0 & otherwise \end{cases} \tag{4.49}$$

As expected, the algorithm is in good agreement with the simulation results even in small value of $NC$, as shown in Figure 4.7.

## 4.4. REVIEW OF TWO MODELS FOR WS CALL SPLITTING

The performance analysis of the WS call splitting discipline has been studied intensively by Hayes [2] and Hui [3], respectively. For the sake of completeness, we would like to include a brief review of these two contributions in the dissertation. We shall follow the notation of [2] and [3] for the convenience of reference.

### 4.4.1. Model I by J. F. Hayes et al. [2]

We assume that the number of copies generated by each incoming packet have the probability distribution $P_i$. The probability generating function for the number of copies generated by an incoming packet is given by

$$X(z) = \sum_{i=1}^{\infty} z^i P_i \tag{4.50}$$

In this study, the contending traffic is characterized by means of residual distributions. Given that an input port is active the probability distribution of $R$ residual number of copies encountered by a tagged packet is given by

$$Pr[R=j] = r_j = \frac{1}{\bar{X}} \sum_{i=j}^{\infty} P_i \tag{4.51}$$

with the corresponding probability generating function

$$R(z \mid active) = \frac{z\,[1 - X(z)]}{\bar{X}\,(1 - z)} \tag{4.52}$$

where $\bar{X}$ is the average number of copies generated by an incoming packet, $\bar{X} = X'(1)$. It is assumed that each of the input ports is active with probability $\rho$; consequently the probability generating function of the number of interfering copies generated by a single packet as encountered by a tagged packet is given by

$$R(z) = \rho \frac{z\,[1 - X(z)]}{\bar{X}\,(1 - z)} + 1 - \rho \tag{4.53}$$

Neglecting the coupling in the contention process for output ports, input ports are assumed to operate independently to keep the complexity of the analysis manageable. The P.G.F. for the total number of conflicting copies is then given by

$$R_T(z) = \sum_{l=0}^{N(N-1)} Q_l\, z^l = [R(z)]^{N-1}$$

$$= [\,\rho \frac{z\,[1 - X(z)]}{\bar{X}\,(1 - z)} + 1 - \rho\,]^{N-1} \tag{4.54}$$

where the $Q_l$ is the probability of a total of $l$ residual copies from all sources, $Q_l = Pr[R_T = l]$. It was also shown that the joint probability generating function is given by

$$T(z_1, z_2, \cdots, z_N) = R_T(\sum_{j=1}^{N} \frac{z_j}{N}) \tag{4.55}$$

The analysis of the packet delay focuses on a particular input port, where a packet generates $K$ copies. Each of these copies will contend, at the output ports with copies generated from the other $N - 1$ input ports. Assume that a particular set of $L$ copies from the designated input port are chosen. Since the traffic is symmetric, this set can be chosen to be the first $L$ copies. The probability that the copies from the particular input port have won contentions at least at first $L$ outputs of the $K$ chosen by the packet, is found to be

$$Pr(M_L \mid K) = \int_0^1 dz_1 \int_0^1 dz_2 \cdots \int_0^1 dz_L\, T(z_1, z_2, \cdots, z_L, 1, \cdots, 1) \qquad L \le K \tag{4.56}$$

The probability of any particular set of exact $L$ copies getting through is given by

$$Pr(O_L \mid K) = \sum_{i=0}^{K-L} \binom{K-L}{i} (-1)^i\, Pr(M_{L+i} \mid K) \qquad L \le K \tag{4.57}$$

The probability of any of the possible sets of $L$ of $K$ copies getting through is given by

$$Pr(L \mid K) = \binom{K}{L} Pr(O_L \mid K) \qquad L \le K \tag{4.58}$$

The probability of successful transmission of all copies in $n$ slots can . calculated by

$$Q(1 \mid K) = Pr(K \mid K) \tag{4.59a}$$

$$Q(n \mid K) = \sum_{l=0}^{K} Pr(l \mid K)\, Q(n-1 \mid K - l) \tag{4.59b}$$

Then the packet service distribution is found by averaging over $K$.

$$Q(n) = \sum_{k=0}^{K} Pr(K = k)\, Q(n \mid K) \tag{4.60}$$

Now, the mean and the mean square service time, then the average packet delay can be calculated in succession.

### 4.4.2. Model II by J. Y. Hui et al. [3]

In this model, it is assumed that each HOL destination is served independently, with identical probability $q$, across the inputs as well as from slot to slot. A packet arriving at an input queue is destined for a random number $F = f$ of outputs with probability $r_f$. The throughput $\lambda_{output}$ is measured in terms of copies per output. We have $\lambda_{output} = E[F]\,\lambda$. Let $N_j$ be the random number of HOL destinations for the output $j$. The value of $N_j$ in the next time slot is

$$N_j' = N_j - \varepsilon(N_j) + A_j \tag{4.61}$$

where $\varepsilon(x) = 1$ if $x > 0$ and 0 otherwise, and $A_j$ is the total number of destinations for $j$ in all fresh HOL packets arriving in the current time slot.

Considering $q$ averaged over all HOL destinations, at a particular slot time, there are $\sum_{j=1}^{j=N} N_j$ destinations at the HOL, out of which, $\sum_{j=1}^{j=N} \varepsilon(N_j)$ will be served. Over a slot time, the probability of having a randomly picked HOL destination served is

$$\bar{q} = \frac{\sum_{j=1}^{N} \varepsilon(N_j)}{\sum_{j=1}^{N} N_j} \tag{4.62}$$

In [3], it was shown that when $N \to \infty$, $q$ is a function of $\lambda_{output}$ only.

$$\bar{q} = \frac{2\,(1 - \lambda_{output})}{2 - \lambda_{output}} \tag{4.63}$$

The formula is slightly pessimistic compared to simulation. Lacking a better model for the dynamics of $q$ as a function of the past history of HOL services, the independence assumption is induced by setting $q = \bar{q}$ for all the time. For a fresh HOL packet, the probability that a destination is served in $U = u$ time slots is

$$P_u = q\,(1-q)^{u-1} \qquad u \geq 1 \tag{4.64}$$

Consequently,

$$P_u(U > u) = \sum_{k=u+1}^{\infty} q\,(1-q)^{k-1} = (1-q)^u \tag{4.65}$$

Let $U_l$ be the random number of time slots required before the $l$-th destination of the packet is served, $1 \leq l \leq F$. The packet is served after $X$ time slots, where

$$X = \max_{l=1\ to\ F} (U_l) \tag{4.66}$$

Consider now $F = f$. Since the $U_l$ are assumed independent for all $1 \leq l \leq f$, then

$$P_X(X \leq x \mid F=f) = \prod_{l=1}^{f} P_{U_l}(U_l \leq x) = [\,1-(1-q)^x\,]^f \tag{4.67}$$

Consequently,

$$\begin{aligned} P_X(X=x \mid F=f) &= P_X(X \leq x \mid F=f) - P_X(X \leq x-1 \mid F=f) \\ &= [\,1-(1-q)^x\,]^f - [\,1-(1-q)^{x-1}\,]^f \end{aligned} \tag{4.68}$$

The probabilities for the service time for a HOL packet is given by the following unconditioning.

$$P_X(X=x) = \sum_{f=1}^{N} r_f\, P_X(X=x \mid F=f) \tag{4.69}$$

The mean service time is given by

$$\begin{aligned} E[X] &= \sum_{x=1}^{\infty} x \sum_{f=1}^{N} r_f\, [\,1-(1-q)^x\,]^f - [\,1-(1-q)^{x-1}\,]^f \\ &= \sum_{f=1}^{N} r_f \sum_{k=1}^{f} \binom{f}{k} \frac{(-1)^{k+1}}{1-(1-q)^k} \end{aligned} \tag{4.70}$$

For a steady-state, $\lambda\, E[X] = 1$. The remaining steps of the delay calculation are fairly straightforward and would not be included here.

## 4.5. A GENERAL UNIFIED MODEL FOR PERFORMANCE ANALYSIS

We may have noted that several approximations are used in above-mentioned models, and some of the approximations are even lack of proper justification. Therefore, we try to avoid these approximation assumptions by using matrix-geometric approach. Another objective of this effort is to set up a unified analytic model for both the one-shot and the WS call splitting disciplines.

The recently developed matrix-geometric approach [6][7] for the analysis of Markov chains with infinite state space have provided an efficient technique for the performance evaluation of complex queueing systems. The key to the application of this approach is a block-partitioned structure of the transition probability matrix characterizing many queueing systems emerging in a variety of practical cases. A typical form of such a transition probability matrix is

$$Q = \begin{bmatrix} B_0 & B_1 & B_2 & 0 & 0 & . \\ A_0 & A_1 & A_2 & 0 & 0 & . \\ 0 & A_0 & A_1 & A_2 & 0 & . \\ 0 & 0 & A_0 & A_1 & A_2 & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \end{bmatrix} \tag{4.71}$$

By proper definition and partition of the overall system's state space, our problem can be shown to have a structure where matrix-geometric technique can be used to compute queue length distribution and other statistical parameters of the system.

### 4.5.1. Description of the System

We consider a queueing system where a cyclic priority input access scheme is assumed. The HOL packets in the queues make the reservations according to their priority level. We assign $N$ levels of priority to the $N$ input ports in the beginning of each time slot and rotate the assignment, ($1 \rightarrow 2 \rightarrow 3 \rightarrow \ldots \rightarrow N \rightarrow 1 \rightarrow 2 \rightarrow \ldots$), cyclically slot-by-slot. The reader may observe that the assignment rotation is reversed with respect to the one in Section 4.3.

Assuming balanced traffic and a symmetric system, we focus our interest on one tagged queue without any loss of generality. The queue is modeled as a discrete time Markov chain. The state of the system at the beginning of each time slot is defined by ($J$, $I$, $K$, $M$) where

$J$ — number of messages in the queue

$I$ — priority level of the queue in current slot

$K$ — number of output channels being reserved by higher priority input ports

$M$ — number of copies of the HOL packet

We now derive the state transition probability from $(J, I, K, M)$ to $(J', I', K', M')$. The transition probability matrix $Q$ is block partitioned and consists of square block matrices $B_0$, $B_1$, $A_0$, $A_1$ and $A_2$.

$$Q = \begin{bmatrix} B_0 & B_1 & 0 & 0 & 0 & . \\ A_0 & A_1 & A_2 & 0 & 0 & . \\ 0 & A_0 & A_1 & A_2 & 0 & . \\ 0 & 0 & A_0 & A_1 & A_2 & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \end{bmatrix} \tag{4.72}$$

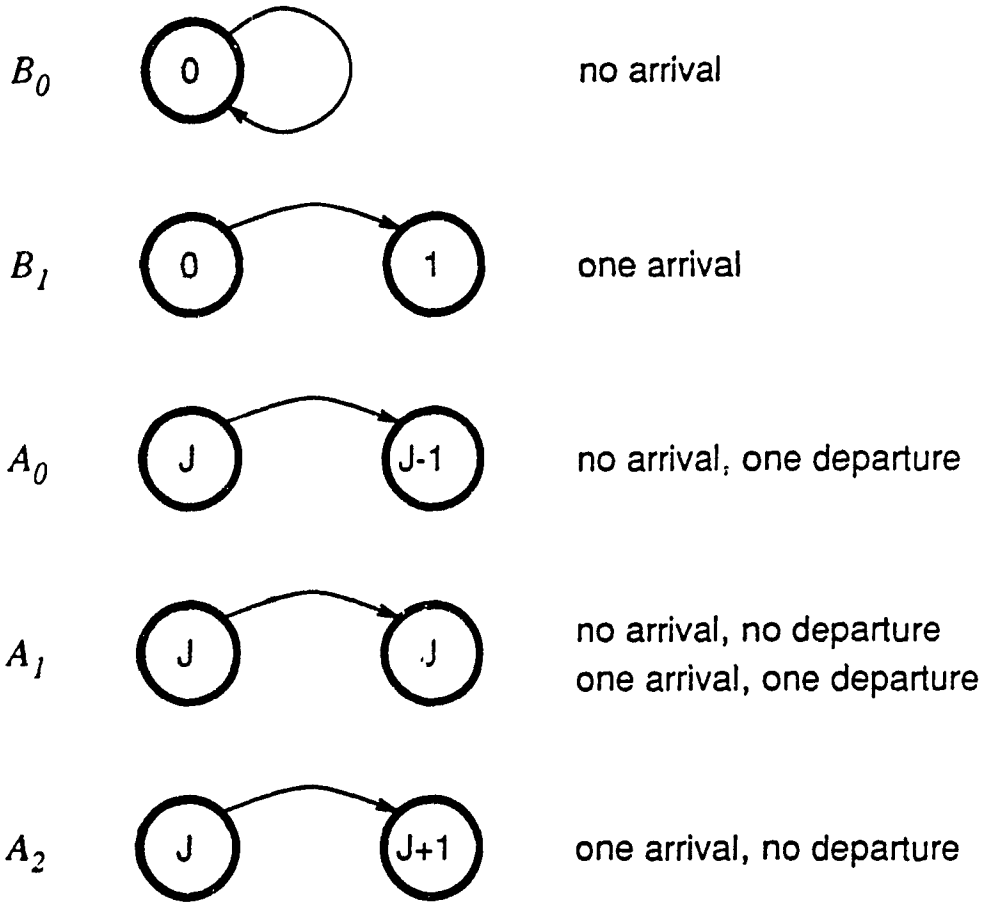



Figure 4.8. Transition of the states

The block partitioned structure can be identified in $Q$ by grouping its terms with respect to the same indices $J$ and $J'$, refer to Figure 4.8. $B_0$ describes the transitions of the case that queue is

empty and no packet arrives during a time slot. $B_1$ describes that a packet arrives at an empty queue during a time slot. $A_i$ describes the case when queue is busy therefore, there may be one departure during a time slot. $A_0$ corresponds to the case of one departure and no arrival in a time slot, while $A_2$ corresponds to the case with no departure and one arrival. The case of one departure and one arrival or the case of no departure and no arrival is described by $A_1$.

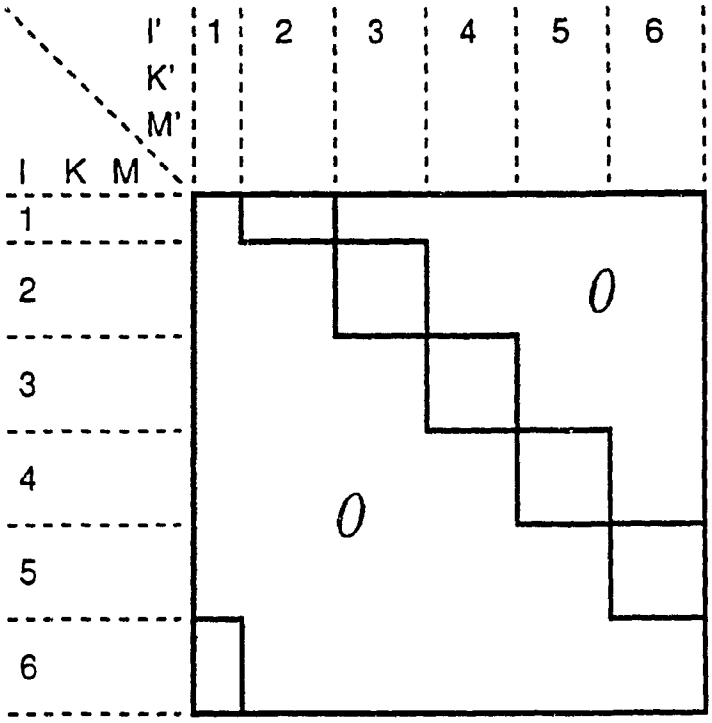



Figure 4.9. Configuration of a block matrix

The indices of the block matrices, $B_0, B_1, A_0, A_1$ and $A_2$, are $I$, $K$ and $M$ where $I = 1, 2, \ldots, N$; $K = 0, 1, 2, \ldots, N$ if $I \neq 1$; $K = 0$ if $I = 1$; $M = 1, 2, 3, \ldots, N$. Therefore, the size of each block matrix is $(N-1)(N+1)N+N = N^3$. Since the transition only happens from one priority level to the subsequent priority level, i.e., $I'$ must be $I + 1 \pmod N$, so at least $\frac{N-1}{N}$ 100% entries are all zeros, as shown in Figure 4.9. The entries of the matrix are determined as follows.

### 4.5.1.A). $Q$ Matrix for One-Shot Discipline

Assume balanced traffic and a symmetric system, therefore the behavior of an input queue in the next time slot is statistically the same as the behavior of the subsequent input queue in the current time slot. Let $R_m$ be the probability that an incoming packet requests $m$ output ports. We further assume that the copies of the packet are absolutely uniformly distributed to the output ports. Now we define $S_M^K$ as the probability that a HOL packet is successfully transmitted, given that the packet has $M$ copies and that $K$ output ports have been reserved by the higher priority packets. Because the copies of a packet are for different output ports, as long as any one of the $M$ copies of the packet does not intend to get access to any one of those $K$ reserved output ports, the packet is transmitted successfully (refer to Figure 4.10a). Moreover, the sum of $K$ and $M$ should not exceed $N$. Hence, we can write down the conditional probability

$$\begin{aligned} S_M^K &= Pr[\textit{packet transmitted} \mid \textit{packet has } M \textit{ copies, } K \textit{ outputs reserved}] \\ &= \frac{\binom{N-K}{M}}{\binom{N}{M}} = \frac{(N-K)(N-K-1)\cdots(N-K-M+1)}{N(N-1)\cdots(N-M-1)} \\ &= \prod_{l=1}^{M}\left(1-\frac{K}{N-l+1}\right) \end{aligned} \tag{4.73}$$

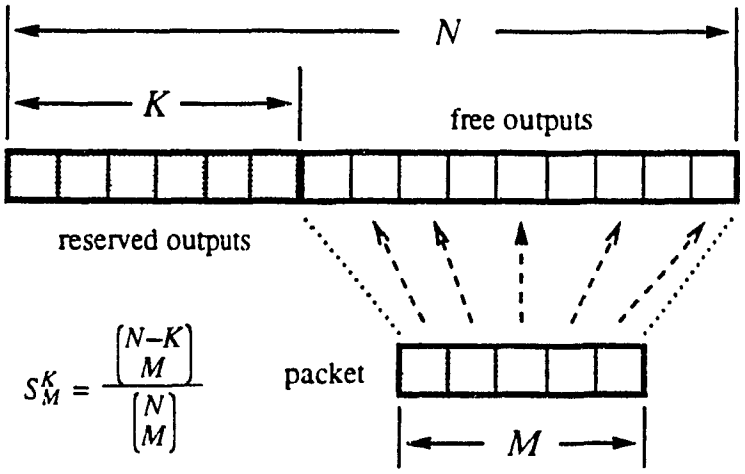
In one-shot case, since the whole packet must be transmitted or blocked at the same time, $K'$ may be either $K + M$ or $K$, i.e.

$$K' = \begin{cases} K+M & \textit{with probability } S_M^K \text{ for } \textit{successful transmission} \\ K & \textit{with probability } 1-S_M^K \end{cases}$$

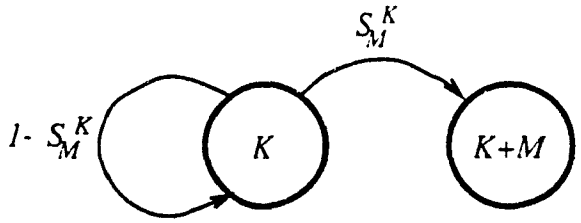
The transition of the states with respect to $K$ and $M$ is shown in Figure 10(b.c). The entries of $Q$ are given by the algorithm listed in Appendix 4A.

### 4.5.1.B). $Q$ Matrix for WS Call Splitting Discipline

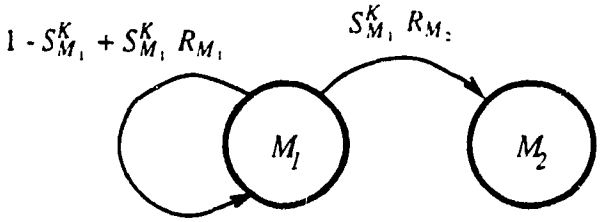
The WS call splitting discipline assumes that copies generated by the same input packet may gain access to output ports independently. A HOL packet may stay at the position until all its copies are transmitted. Thus, the process of transmission may be dispersed over several time slots. Most probably, a HOL packet is a residual packet which only holds a residual number of




( a )

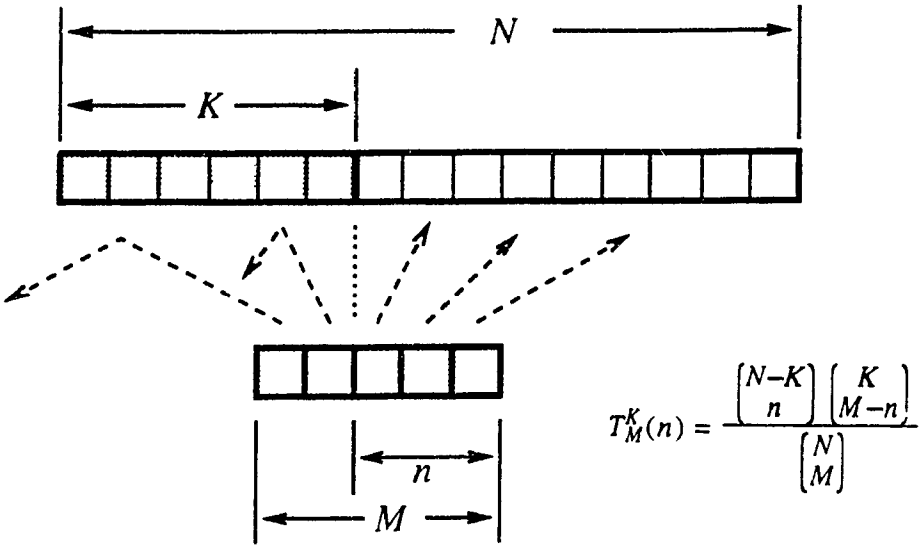



( b ) transition of states with respect to K

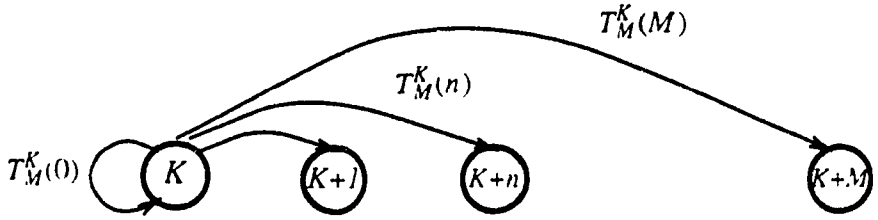



( c ) transition of states with respect to M

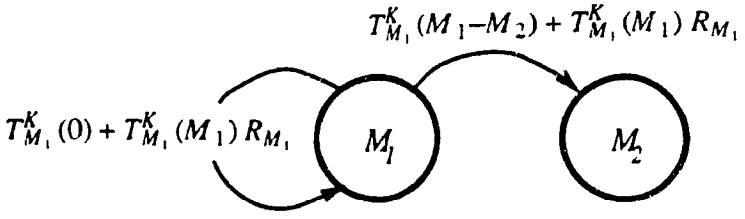
Figure 4.10. Transition of the states within block matrices for one-shot scheduling




( a )




( b ) transition of states with respect to K




( c ) transition of states with respect to M

Figure 4.11. Transition of the states within block matrices for WS call splitting

copies. We define $T_M^K(n)$ as the probability that $n$ of $M$ copies of the packet are transmitted successfully, given that $K$ output ports have been reserved by the higher priority packets. The relation between $K$, $M$, $K'$ and $M'$ is expressed as below

$$\begin{matrix} K' = K + n \\ M' = M - n \end{matrix} \qquad \begin{cases} \textit{with probability } T_M^K(n) \\ n = 1, 2, \cdots, max(M-1, N-K) \end{cases}$$

where (refer to Figure 4.11a)

$$T_M^K(n) = Pr[n \textit{ copies transmitted} \mid \textit{packet has } M \textit{ copies, } K \textit{ outputs reserved}]$$
$$= \frac{\binom{N-K}{n}\binom{K}{M-n}}{\binom{N}{M}} \tag{4.74}$$

If the whole packet gets through in current slot, then a fresh packet is coming with a complete request of destinations, such that

$$\begin{matrix} K' = K + M \\ M' = m \end{matrix} \qquad \begin{cases} \textit{with probability } T_M^K(M) = S_M^K \\ \textit{with probability } R_m \end{cases}$$

The transition of the states with respect to $K$ and $M$ for WS call splitting is shown in Figure 11(b,c). Since $B_0$, $B_1$, $A_0$ correspond to the cases that there is no possible splitting transmissions, only $A_1$ and $A_2$ need to be modified based on the algorithm for the one-shot discipline. The algorithm for the formation of the $Q$ matrix for the WS call splitting discipline is listed in Appendix 4B.

Having obtained the block partitioned structure for $Q$ matrices of both the one-shot and the WS call splitting disciplines, we are now in a position to apply the matrix-geometric technique to calculate the queueing distribution. This is the topic of the next section.

### 4.5.2. Solution via Matrix Geometric Approach

The matrix geometric method was introduced by Neuts in [6][7] and has been used successfully to analyze a number of complicated queueing systems (e.g., [8][9][10]). Here, instead of repeating the analysis in the literature, we briefly outline the major steps in applying the matrix geometric technique to calculate the packet delay. Our goal is to compute the steady state probabilities of the system. Let $x_{jikm} = Pr[J=j, I=i, K=k, M=m]$ denote the stationary probability of

the system state. We define the state probability vectors $\mathbf{x}_j$, $j \geq 0$, as

$$\mathbf{x}_j = [\ x_{jikm};\ i = 1,2,\cdots,N;\ k = 0,1,\cdots,N \text{ if } i \neq 1;\ k = 0 \text{ if } i = 1;\ m = 1,2,\cdots,N\ ]^T$$

Each $N^3$-dimensional state probability vector $\mathbf{x}_j$ corresponds to the state probabilities at the level of having $j$ packet in the queue. Now, we start the computation by introducing some standard notation.

$$A = \sum_{v=0}^{\infty} A_v = A_0 + A_1 + A_2 \tag{4.75}$$

$$G = \sum_{v=0}^{\infty} A_v G^v = A_0 + A_1 G + A_2 G^2 \tag{4.76}$$

The obvious numerical procedure for computing the matrix $G$ is by successive substitution, starting with $G = 0$. Once $G$ is computed, we further define $K$

$$K = \sum_{v=0}^{\infty} B_v G^v = B_0 + B_1 G \tag{4.77}$$

Then, we turn to calculate $\mathbf{x}_0$, which corresponds to probability that the queue is empty, based on the proof in [7],

$$\mathbf{x}_0 = \frac{\mathbf{k}}{\mathbf{k}\,\mathbf{k}_1} \tag{4.78}$$

where $\mathbf{k}$ is the invariant probability vector of $K$ ($\mathbf{k}K = \mathbf{k}$), and

$$\mathbf{k}_1 = \mathbf{e} + \sum_{u=1}^{\infty} \left[ B_u \sum_{v=0}^{u-1} G^v \right] \mu = \mathbf{e} + B_1 \mu \tag{4.79}$$

$\mathbf{e}$ is the all 1 column vector. The vector $\mu$ is given by

$$\mu = [\ I - G + \mathbf{e}\,\mathbf{g}\ ]\ [\ I - A + (\mathbf{e} - \beta)\,\mathbf{g}\ ]^{-1}\,\mathbf{e} \tag{4.80}$$

$$\beta = \sum_{v=0}^{\infty} v\, A_v\, \mathbf{e} = (\ A_1 + 2\,A_2\ )\,\mathbf{e} \tag{4.81}$$

where $\mathbf{g}$ is the invariant probability vector of $G$ such that $\mathbf{g}G = \mathbf{g}$; $I$ is the identity matrix. The vectors $\mathbf{x}_j$, $j \geq 1$, can be computed by Ramaswami's recursive equation [11]

$$\mathbf{x}_j = \left[ \mathbf{x}_0 \sum_{v=j}^{\infty} B_v G^{v-j} + \sum_{u=1}^{j-1} \mathbf{x}_u \sum_{v=j+1-u}^{\infty} A_v G^{v-j-1+u} \right] \left( 1 - \sum_{v=1}^{\infty} A_v G^{v-1} \right)^{-1} \tag{4.82}$$

which is stable and can be explicitly stated for our system, as follows

$$\mathbf{x}_1 = \mathbf{x}_0 B_1 (I - A_1 - A_2 G)^{-1} \tag{4.83a}$$

$$\mathbf{x}_j = \mathbf{x}_{j-1} A_2 (I - A_1 - A_2 G)^{-1} \qquad j \geq 2 \tag{4.83b}$$

From the vectors $\mathbf{x}_j$, we can obtain the stationary queue length distribution and mean value of the queue length, $L$. The packet delay can be calculated from Little's formula.

$$D = \frac{L}{\lambda} \tag{4.84}$$

This simple general computation procedure can be used for both one-shot and WS call splitting disciplines. The approximation of the copy distribution model is avoided. Furthermore, our model has the advantage of being able to incorporate any arbitrary packet copy distribution in the performance analysis. Some numerical results were plotted in Figures 4.12 and 4.13, which is in satisfactory agreement with the simulation results, for both the case of a modified binomial distribution for the number of copies and the case of constant number of copies. The slight discrepancy with the simulation results is due to the fact that our model still uses the approximation assumption that the copies of the blocked packets are evenly distributed over the output ports.

## 4.6. REMARKS

In this chapter, we presented some analytic tools of traffic theory for the multicast switching system. These results could serve to model the multicast packet switch and to predict the onset of congestion of the system. Although this chapter is devoted to the multicast switch, the analytic models can also be applied to other queueing problems. In consideration of future work in this area, we should realize that there are still a lot of work to be done. For example, in Section 4.5.2, we have noticed that when $N \rightarrow \infty$ and keeping a constant $\overline{m}$ (the average number of copies generated by a packet), the curve of delay will converge to an asymptote. It would be of interest to prove a monotonic convergence and derive the approximate performance of a large size switch. Finally, although the matrix geometric approach is quite satisfactory in modeling, its calculation takes long time and needs a large size of the data and stack segments (the size of the matrices for computation increases cubically, referring to switch size $N$). Hence, approximations based on reduced state space models yielding less computational demand are of particular interest.

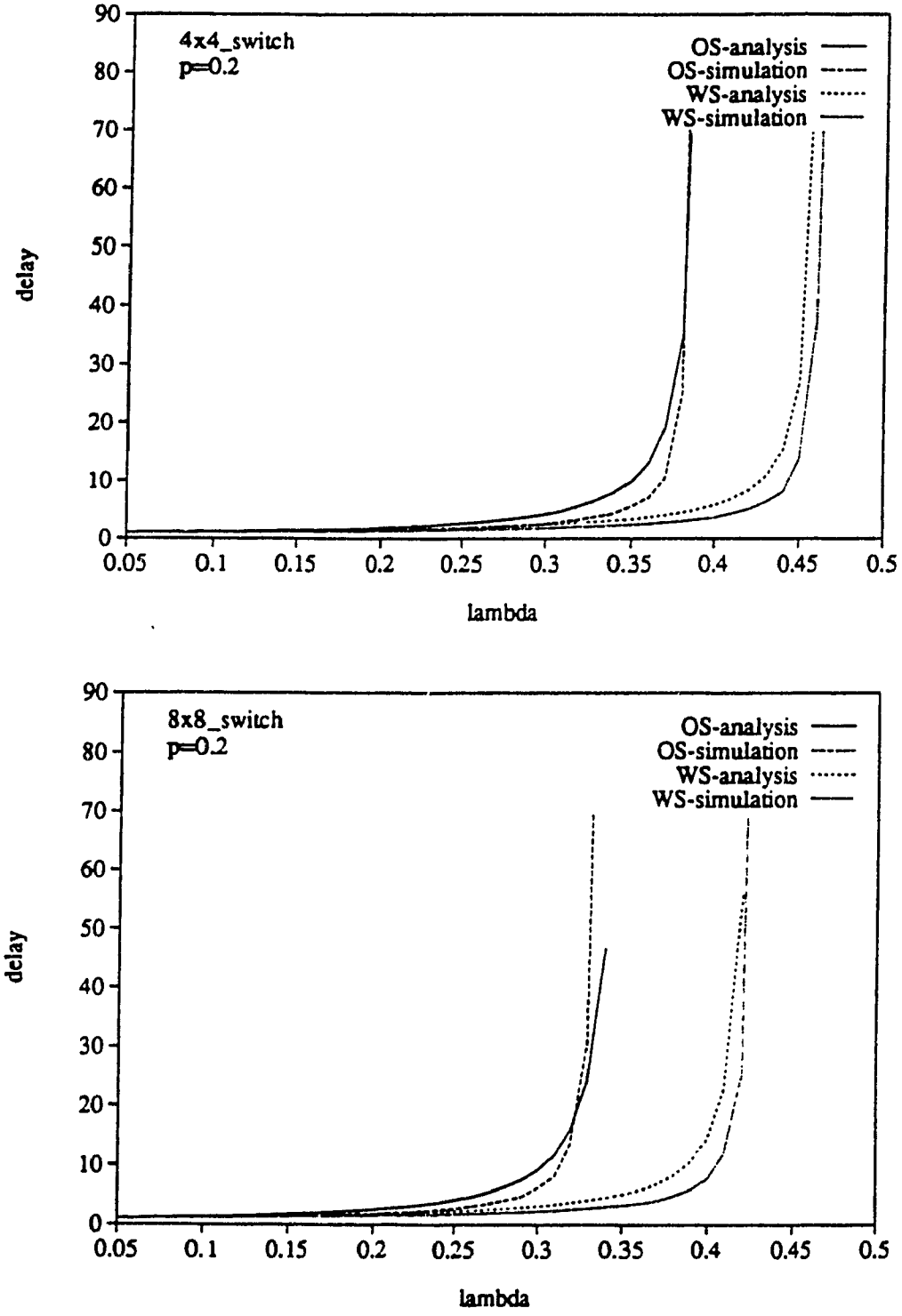



Figure 4.12. Numerical results with binomial copy distribution
(OS: one-shot; WS: WS call splitting)

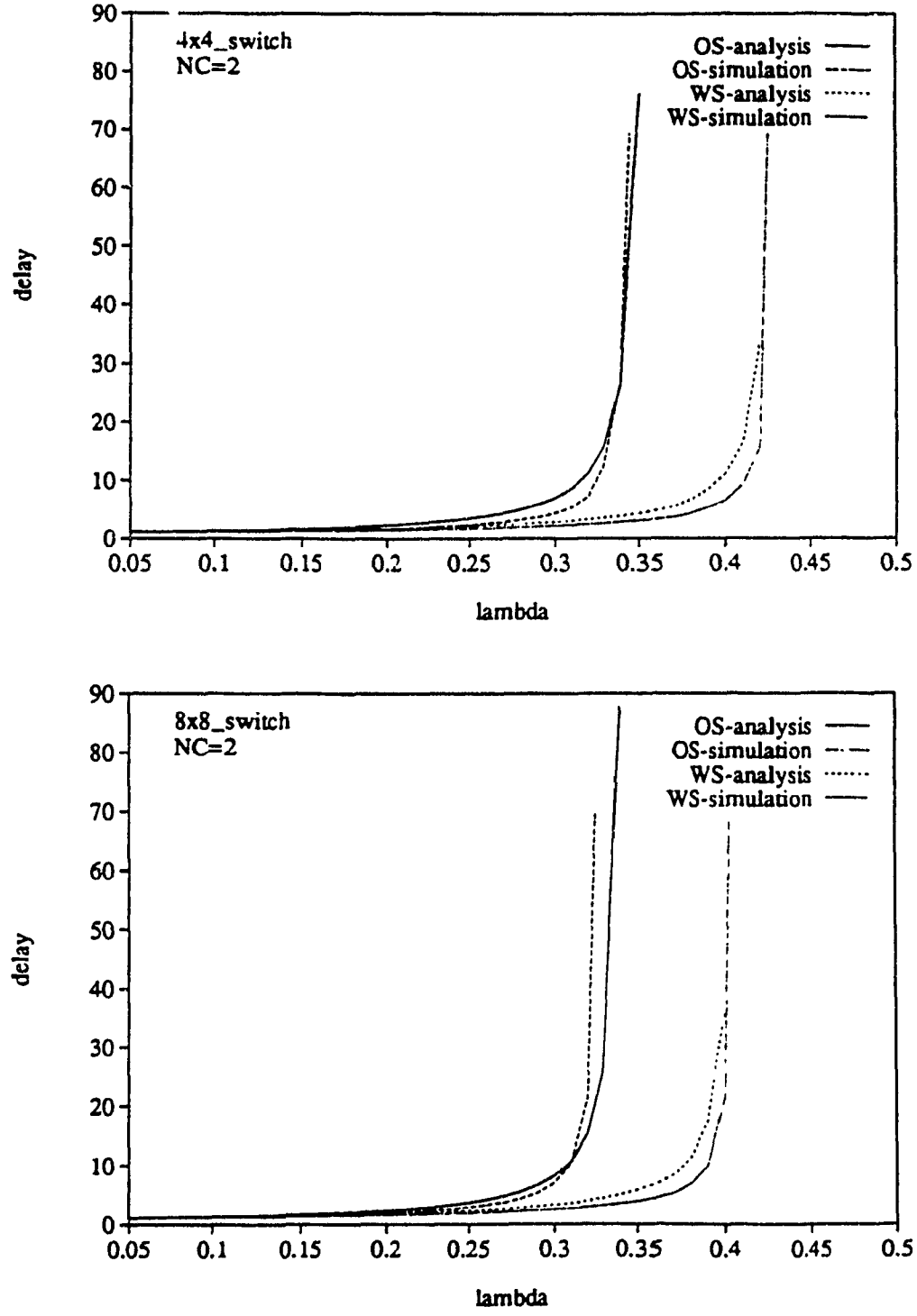



Figure 4.13. Numerical results with deterministic copy distribution (constant fanout of the packet)

## APPENDIX 4A: Algorithm for the Formation of the $Q$ Matrix (One-Shot Scheduling)

for all $I = 1$ to $N$ do begin
  for all K = 0 to $N$ do begin
    if $I = 1$ and $K \neq 0$ then break
    for all $M = 1$ to $N$ do begin
      if $I \neq N$ then $I' = I + 1$
      else $I' = 1$
      for all $K' = 0$ to $N$ do begin
        if $I' = 1$ and $K' \neq 0$ then break
        for all $M' = 1$ to $N$ do begin
          row = $M$
          if $I > 1$ then row = row + $(I - 2) * N^2 + (I + K - 1) * N$
          column = $M'$
          if $I' > 1$ then column = column + $(I' - 2) * N^2 + (I' + K' - 1) * N$

          if $I = 1$ and $K' = 0$ and $M = M'$
          or if $1 < I < N$ and $K = K'$ and $M = M$
          or if $I = N$ and $M = M'$ then
              $B_0$[row][column] = $1 - \lambda$

          if $I = 1$ and $K' = 0$
          or if $1 < I < N$ and $K = K'$
          or if $I = N$ then
              $B_1$[row][column] = $\lambda * R_{M'}$

          if $I \neq N$ and $K' = K + M$
          or if $I = N$ then
              $A_0$[row][column] = $(1 - \lambda) * S_M^K * R_{M'}$

          if $I = 1$ and $K' = M$ then
              $A_1$[row][column] = $\lambda * R_{M'}$
          else if $1 < I < N$ and $K' = K + M$ then
              $A_1$[row][column] = $\lambda * S_m^K * R_{M'}$
          else if $1 < I < N$ and $K = K'$ and $M = M'$ then
              $A_1$[row][column] = $(1 - \lambda) * (1 - S_M^K)$
          else if $I = N$ then begin
              $A_1$[row][column] = $\lambda * S_M^K * R_{M'}$
              if $M = M'$ then
                  $A_1$[row][column] = $A_1$[row][column] + $(1 - \lambda) * (1 - S_M^K)$
          end

          if $1 < I < N$ and $K = K'$ and $M = M'$
          or if $I = N$ and $I' = 1$ and $M = M'$ then
              $A_2$[row][column] = $\lambda * (1 - S_M^K)$

        end
      end
    end
  end
end

## APPENDIX 4B: Algorithm for the Formation of the $Q$ Matrix (WS Call Splitting)

```
for all I = 1 to N do begin
  for all K = 0 to N do begin
    if I = 1 and K ≠ 0 then break
    for all M = 1 to N do begin
      if I ≠ N then I' = I + 1
      else I' = 1
      for all K' = 0 to N do begin
        if I' = 1 and K' ≠ 0 then break
        for all M' = 1 to N do begin
          row = M
          if I > 1 then row = row + ( I - 2 ) * N^2 + ( I + K - 1 ) * N
          column = M'
          if I' > 1 then column = column + ( I' - 2 ) * N^2 + ( I' + K' - 1 ) * N

          if I = 1 and K' = 0 and M = M'
          or if 1 < I < N and K = K' and M = M
          or if I = N and M = M' then
              B0[row][column] = 1 - λ

          if I = 1 and K' = 0
          or if 1 < I < N and K = K'
          or if I = N then
              B1[row][column] = λ * RM

          if I ≠ N and K' = K + M
          or if I = N then
              A0[row][column] = ( 1 - λ ) * TM^K(M) * RM

          if I = 1 and K' = M then
              A1[row][column] = λ * RM
          else if 1 < I < N and K' = K + M then
              A1[row][column] = λ * TM^K(M) * RM
          else if 1 < I < N and K' + M' = K + M and K' ≥ K and K' < K + M then
              A1[row][column] = ( 1 - λ ) * TM^K(K'-K)
          else if I = N then begin
              A1[row][column] = λ * TM^K(M) * RM
              if M ≥ M' and M - M' + K ≤ N then
                  A1[row][column] = A1[row][column] + ( 1 - λ ) * TM^K(M-M')
          end

          if 1 < I < N and K' + M' = K + M and M' ≤ M
          or if I = N and M' ≤ M and M - M' + K ≤ N then
              A2[row][column] = λ * TM^K(M-M')

        end
      end
    end
  end
end
```

## REFERENCES 4

[1] M. J. Karol, M. G. Hluchyj, S. P. Morgan, "Input versus output queueing on a space-division packet switch", IEEE Trans. on Comm., Vol.35, No.12, pp.1347-1356, Dec. 1987.

[2] J. F. Hayes, R. Breault, M. K. Mehmet Ali, "Performance analysis of a multicast switch", IEEE Trans. on Comm., Vol.39, No.4, pp.581-587, April 1991.

[3] J. Y. Hui, T. Renner, "Queueing strategies for multicast packet switching", Proc. of IEEE Globecom'90, pp.1431-1437, San Diego, CA, Dec. 1990.

[4] L. Kleinrock, *Queueing Systems. Volume 1: Theory*, Wiley, 1975.

[5] H. M. Taylor, S. Karlin, *An Introduction to Stochastic Modeling*, Academic Press, Orlando, FL, 1984.

[6] M. F. Neuts, *Matrix-Geometric Solutions in Stochastic Models - An Algorithmic Approach*, Baltimore and London: The John Hopkins University Press, 1981.

[7] M. F. Neuts, *Structured Stochastic Matrices of M/G/1 Type and Their Applications*, New York: Marcel Dekker, Inc., 1989.

[8] H. Ahmadi, R. Guerin, "Analysis of a class of buffer storage systems with Markov-correlated input and bulk service", IBM Research Report, RC 15320 (No.67501), Nov. 1989.

[9] H. Ahmadi, R. Guerin, K. Sohraby, "Analysis of a rate-based access control mechanism for high-speed networks", IBM Research Report, RC 15831 (No.70366), June 1990.

[10] D. M. Lucantoni, "New results on the single server queue with a batch Markovian arrival process", Commun. Statist. -Stochastic Models, 7(10, pp.1-46, 1991.

[11] V. Ramaswami, "A stable recursion for the steady state vector in Markov chains of M/G/1 type", Commun. Statist. -Stochastic Models, 4(1), pp.183-188, 1988.

# CHAPTER 5

# SWITCH ARCHITECTURE

5.1. INTRODUCTION

5.2. A SURVEY OF MULTICAST PACKET SWITCHES

5.2.1. Banyan-Based Space-Division Switch

5.2.2. Knockout Switch

5.2.3. Shift Switch

5.2.4. Shared Buffer Memory Switch

5.3. A SHARED BUFFER MEMORY SWITCH WITH MAXIMUM QUEUE AND MINIMUM ALLOCATION

5.3.1. Classifications of Shared Memory Switches

5.3.2. The Switch Architecture

5.3.3. Estimation of Packet Loss Probability

5.3.3.1 Balanced Traffic

5.3.3.2 Unbalanced Hot-Spot Traffic

5.3.4. The Capabilities of Multicasting, Modularity and Priority

5.4. CONCLUSION

REFERENCES 5

## 5.1. INTRODUCTION

In this chapter, we will discuss multicast packet switch architectures. We begin by looking at multicast packet switches that have been proposed in literature. Various ATM switches have been proposed and developed, the most promising of which are able to provide multicast operation without exception. Many survey papers of these fast packet switches have been published recently [1][2][3][4][5][6]. In general, there were two ways of classifying switches. The various switch fabrics may be classified into three categories: the shared memory type, the shared medium type, and the space-division type. They may also be classified in terms of the arrangement of buffer memories, into four categories: input buffer switch, output buffer switch, shared buffer switch and crosspoint buffer switch. Multicast operation was only taken as an extra function. However, here, our survey will concentrate on the switches with the multicast feature.

In the second part of this chapter, we will propose a shared buffer memory switch architecture. In this switch, a shared buffer pool and a dedicated buffer pool are properly handled for switching and buffering the packets. Since the policy prevents the monopolization of the shared buffers and, at the same time, insures a full utilization of all the $N$ output channels, the sharing policy with maximum queue and minimum allocation provides good traffic handling characteristics especially under unbalanced or bursty traffic. The packet loss probability is examined under both balanced and unbalanced traffic. Finally, we will describe the multicast operation of this switch which is more efficient than Kuwahara's shared buffer memory switch [7]. The capabilities of modularity and priority are also taken into consideration.

## 5.2. A SURVEY OF MULTICAST PACKET SWITCHES

Due to limited space, we shall only summarize the following four types of switches which are able to accommodate a multicast capability: the banyan-based space-division switch, the knockout switch, the shift switch, and the shared buffer memory switch.

### 5.2.1. Banyan-Based Space-Division Switch

Banyan-based space-division multicast switches have been proposed in many papers [8][9][10]. The primary model of the multicast packet switch consists of a copy network and a routing network which are sequentially combined to copy and switch fixed length packets in slot times.

*Routing Networks*

As we have mentioned in Chapter 3, a banyan network itself is a blocking network. In the past, various efforts have been made to overcome blocking. One solution is known as the buffered-banyan switching fabric, where packet buffers are placed at the inputs of each internal stage switch node, such as [9]. Another way to develop a nonblocking, self-routing network for packet switching is a combination of Batcher's bitonic sorting network [11] and a banyan network. This arrangement is based on the observation that the banyan network is internally nonblocking if the destinations of input requests are sorted in ascending or descending order [8].

An example of 8×8 Batcher-banyan nonblocking routing network is illustrated in Figure 5.1. The header of each input packet to the switch contains local routing information, an activity bit (AC), a source address (SA) and a destination address (DA). The sorting network sorts the synchronized input packets based on the destination addresses preceded by the activity bit as the most significant bit. The compact set of ordered packets are then routed to their final destinations by the self-routing banyan network, which is done by decoding the header of each incoming packet. That is, a node at stage $k$ sends the packet out on link 0 (up) or link 1 (down) according to the $k$th bit of the header. No centralized routing control is needed.

*Copy Networks*

The copy network receives multicast packets on input lines and produces a required number of copies on output lines. We shall review several of these that have been proposed.

The first design of the copy network might be the Starlite copy network [8], which is a receiver initiated system. The inputs to the network are original source packets and empty copies, as a duplicate data field, generated by receivers (Figure 5.2). These inputs are sorted on their source addresses. The original source packet and the associated empty copies with the same

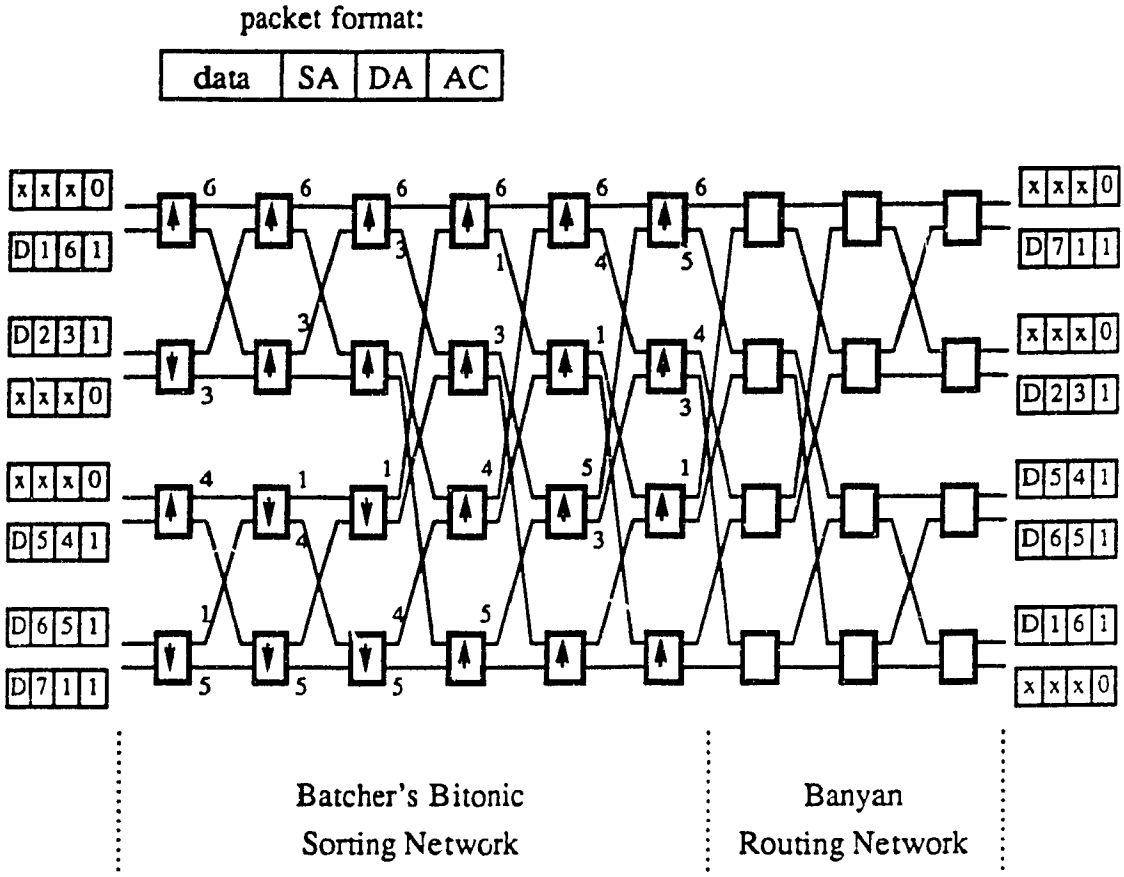



Figure 5.1. A 8×8 Batcher-banyan nonblocking routing network

source address will appear contiguously at the input of copy network. Then the copy network replicates the data in each source packet and inserts them into the empty data fields of the associated empty copies. Therefore, the process of packet replication in this design assumes the synchronization of the source and destinations. It is not feasible to implement this approach in a broadband packet network, since original source packet and empty copies may suffer unpredictable variant delay in reaching the switch node.

Turner's copy network [9] employs a banyan network and a set of broadcast and group translators (BGT's), as shown in Figure 5.3. The header of an input multicast packet contains two fields, a fan-out indicating the number of copies required by the packet, and a broadcast channel number (BCN) used by the BGT's to determine the final destination of the copies of the packet.

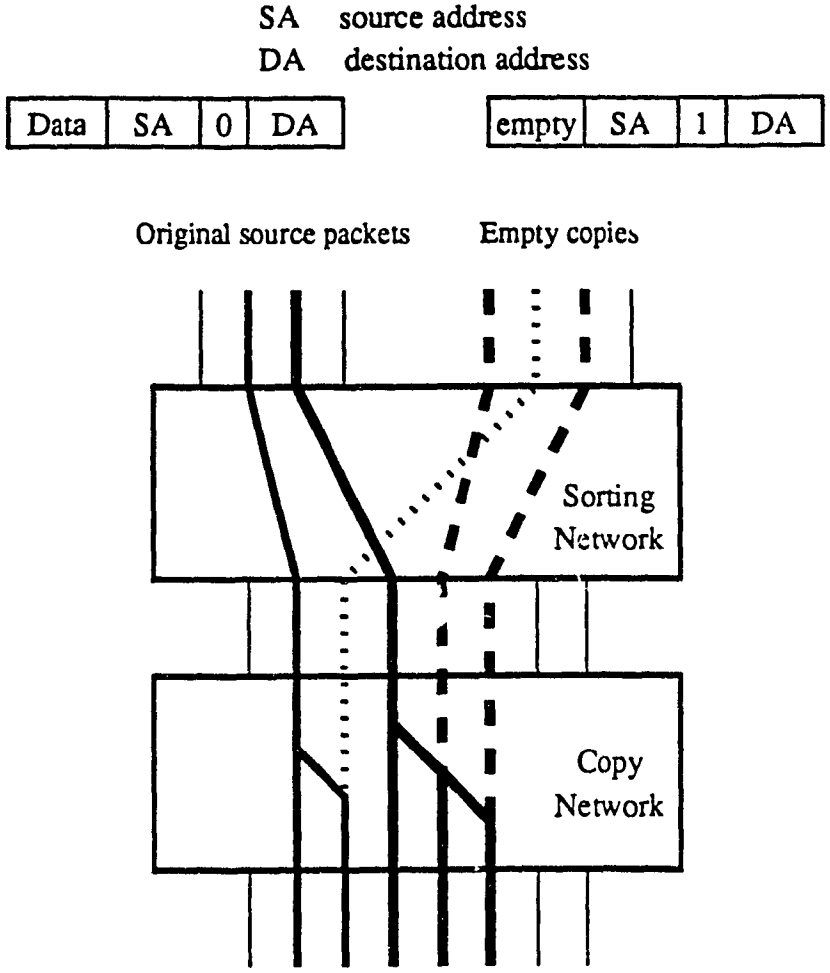



Figure 5.2. Copy network of Starlite switch

The packets are replicated by splitting the fan-out in the banyan network. However, packet collisions can not be avoided in this scheme, and it is blocking. Buffers are required for every internal node to prevent packet loss. Furthermore, packets may be out of sequence upon reaching the output of the copy network. Turner's algorithm delays copying packets as long as possible.

Recently, a two-phase copying process was proposed in [12], and it seems a derivative of Turner's copy network. The two-phase copy network uses a dynamic routing algorithm for packet replication which permits packets to fan out at the earliest possible stage. Once contention

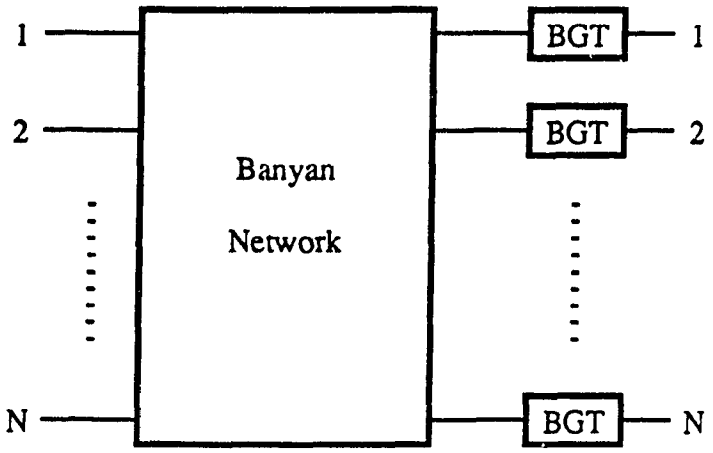



Figure 5.3. Copy network of Turner's broadcast packet switch

occurs when two packets reach a switching node in the broadcast network simultaneously, the replication of either packet will be delayed until the next available switching node is reached. This process is repeated until the packet has generated the required number of copies. Clearly, this scheme needs more complicated control logic signals.

Lee's copy network [13] is based on cascaded concentration-encoding-copying networks. The advantages of this copy network are non-blocking and constant latency time for packet replication. Although the hardware complexity is very high, it is still a promising scheme; accordingly, we take some time to detail Lee's copy network.

The header of each incoming packet includes a broadcast channel number (BCN) and a copy number (CN) which indicates the total number of copies of the packet which are required. The copy network comprises five principal elements (Figure 5.4):

(1) Running Adder Network: for generating running sums of copy numbers specified in the headers of input packets.

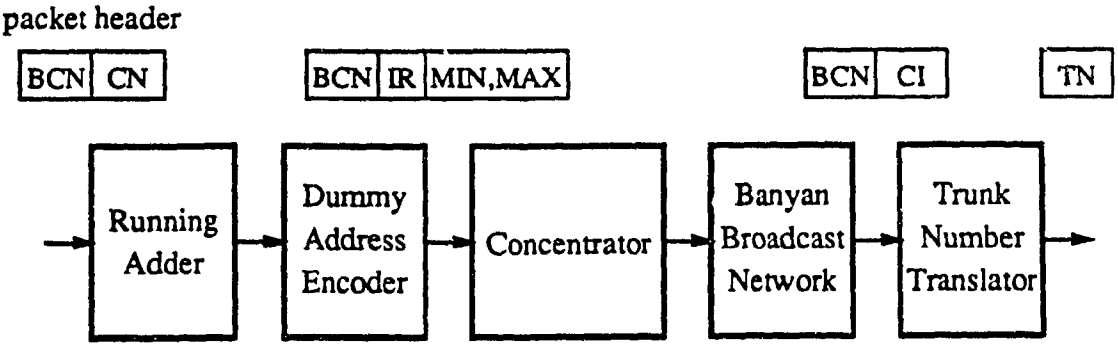



Figure 5.4. A block diagram of Lee's copy network

(2) Dummy Address Encoders: for generating adjacent running sums to form new packet headers used for routing in the copy network. These include an address interval defined by a minimum address (MIN) and maximum address (MAX) as well as an index reference (IR) set equal to the minimum address. The dummy address is used in broadcast networks only.

(3) Concentrator Network: for removing the idle inputs between active inputs which must be compact to satisfy the nonblocking condition (monotone and concentration) [13] of a broadcast banyan network. One of the concentrators is proposed in [8]. The reverse banyan network routes the active packets based on the sums of the activity bits so that all of the packets will emerge contiguously at the outputs.

(4) Broadcast Banyan Network: comprising an array of interconnected identical switch elements for performing packet replication and routing so that a copy of each packet is routed to an output address contained in its associated address interval (MIN, MAX). As a result of processing according to the Boolean Interval Splitting algorithm, this address must be the port address where the copy comes out. Then we can obtain the copy index (CI) by CI = output address - IR.

(5) Trunk Number Translator: for determining the outgoing trunk number (TN), or destination address, of each copy emerging from the broadcast banyan network. The destination address assignments can be accomplished by a simple table look-up, with attributes BCN and CI as primary key of the table.

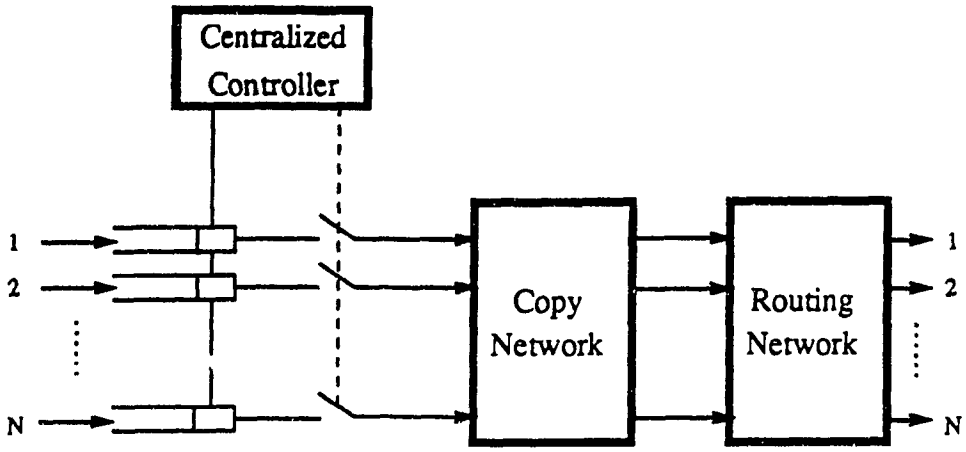



Figure 5.5. Multicast switch with centralized controller

*Overall Structure*

There are two problems inherent in the design of a banyan-based space-division multicast packet switch which we have mentioned many times in previous chapters. One is the occurrence of overflows in the network when the total number of destination requests exceeds the number of outputs. The other is output port conflicts when multiple packets request the same output port concurrently. The overall structures of the multicast switches proposed in [8][9][10] were focused on the solution to these problems. But without exception, all these three algorithms do not persist in the FIFO principle. Copies may be out of sequence upon reaching the destinations. This may happen, even though we have non-blocking, self-routing copy networks and routing networks in hand. We have argued that the structure with a centralized controller as shown in

Figure 5.5 is a feasible solution so far as the FIFO principle is concerned. Under this arrangement, no buffer is needed between the copy network and routing network. The packet walking time between the input and the output of the switch is constant.

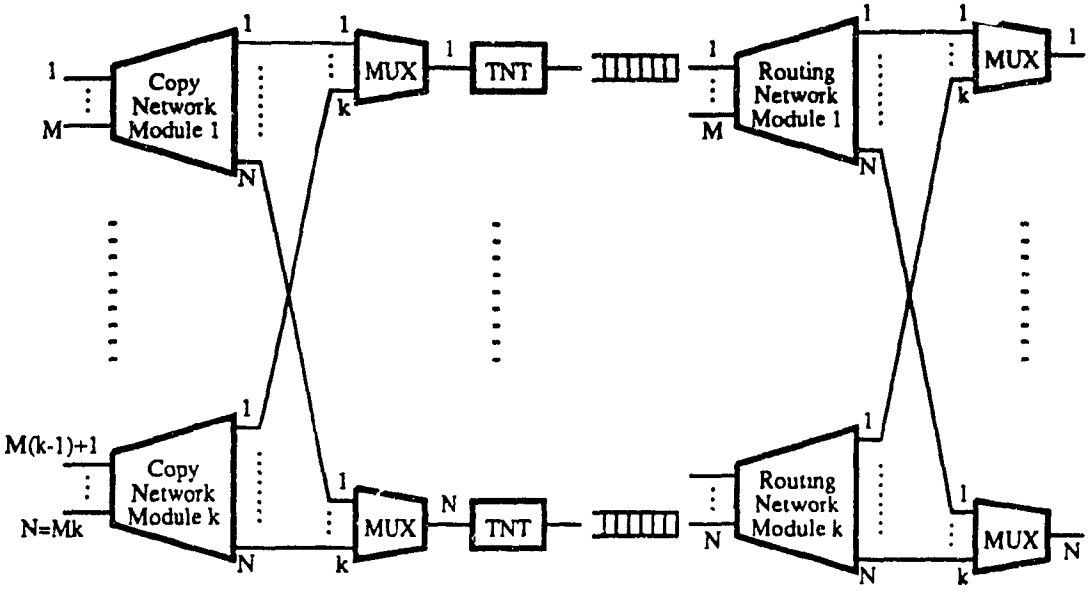
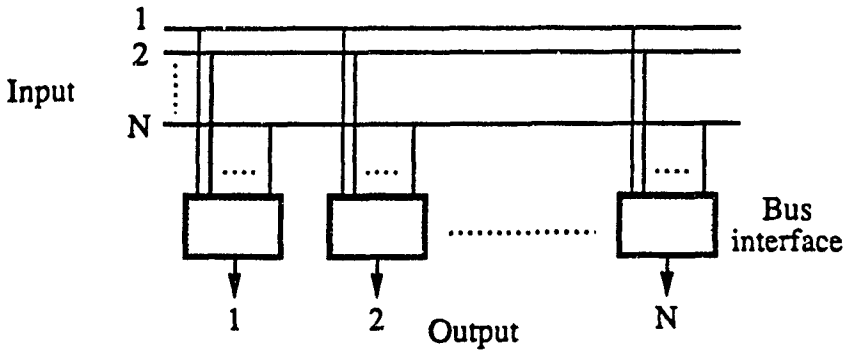
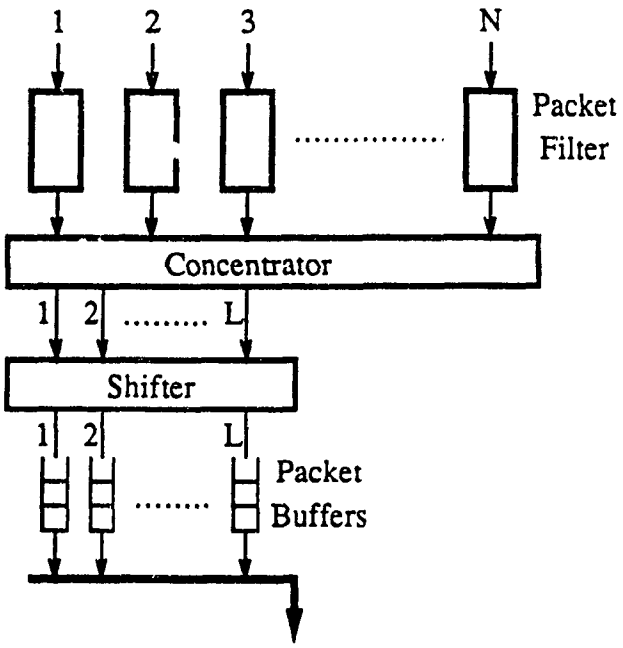



Figure 5.6. A modular structure of a Batcher-banyan multicast switch

To build a large size multicast switch, a modular structure is most appropriate. A modular structure of a Batcher-banyan based space-division multicast switch is depicted in Figure 5.6. The set of all $N$ inputs is equally partitioned into $k$ subsets. Each subset of $M$ inputs, where $M = N / k$, are then duplicated toward up to $N$ output ports, in a copy network module. Each copy network module consists of a running adder, a dummy address encoder, a concentrator and a broadcast network. The broadcast network is an $M$:$N$ expansion network constructed by binary trees and square banyan networks [12][14]. The outputs of $k$ copy network modules are multiplexed into $N$ output lines. The destination address of each copy is translated from their multicast channel number and copy index by trunk number translator (TNT). The offered copies are further transmitted to the destinations through the modular routing network [14] which also contains expansion networks.




(a) Interconnection Fabric




(b) Bus Interface

Figure 5.7. Knockout switch

### 5.2.2. Knockout Switch

The knockout switch is recently proposed [15][16] for high-performance packet networks which can provide multicast services under both light and heavy traffic circumstances.

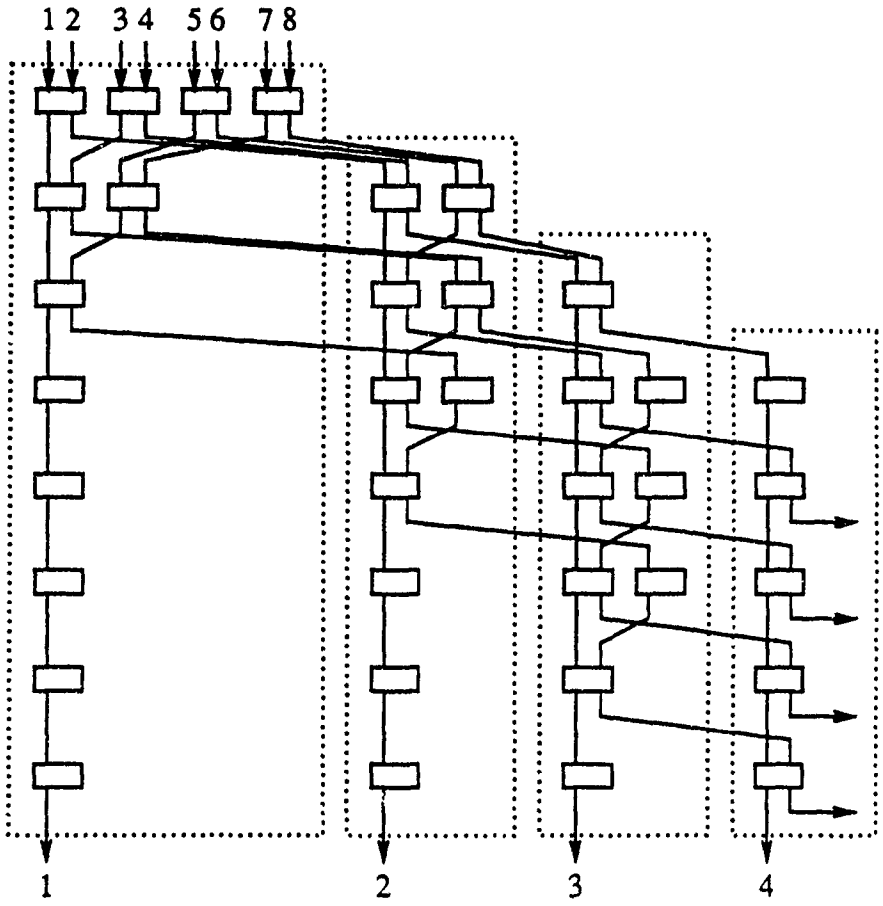



Figure 5.8. A 8:4 concentrator

*The Knockout Switch Architecture* [15]

The knockout switch uses a disjoint interconnected path fabric topology (shown in Figure 5.7) to passively broadcast all input packets to all outputs so that no blocking occurs. Each Bus Interface associated with a particular output is wired to the $N$ input signals from the $N$ input broadcast buses. Inside each Bus Interface, the input packets are first passed through $N$ parallel packet filters where those destined for other outputs are discarded. Those packets addressed to the output pass through the filters and into a $N$:$L$ concentrator, as shown in Figure 5.8, whose function is to select up to $L$ ($L \ll N$) packets out of a total of up to $N$ contenders. This selection is

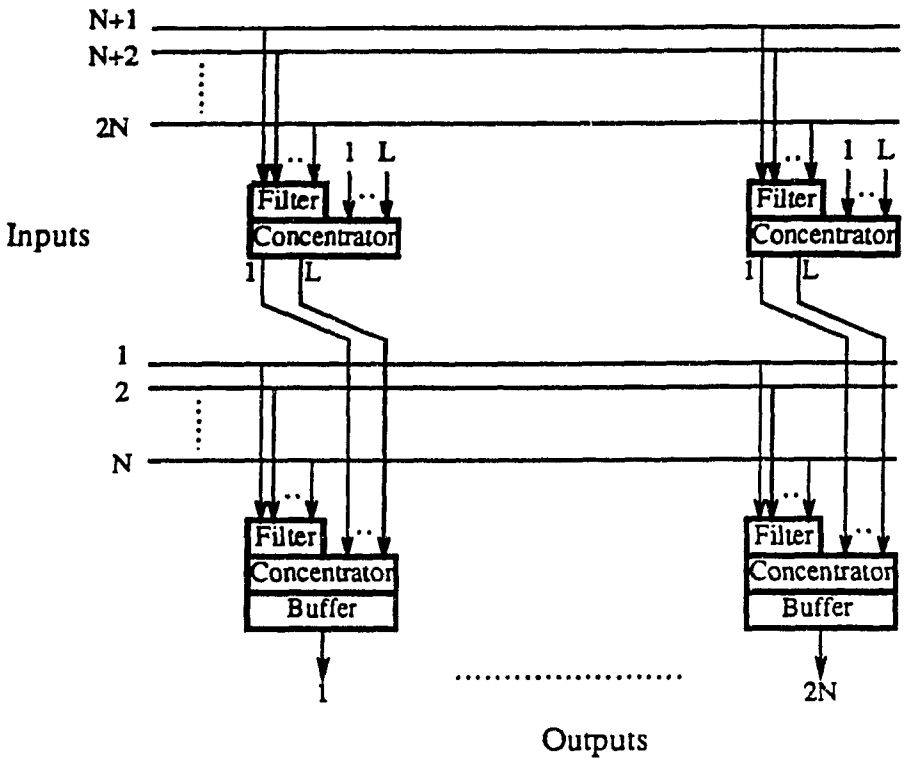



Figure 5.9. Modular growth of knockout switch

based on a knockout tournament strategy where up to $L$ winners are chosen. The losers are simply discarded even though they have passed packet filter. A proper value of $L$ is chosen to guarantee that the probability of losing a packet within the concentrator is no greater than a specified threshold. This knockout contention scheme, being the most novel aspect of the Knockout Switch, reduces the overall complexity of the switch. The $L$ winning packets then enter an $L$-input, one output shared buffer. The final unloading of the packets from the shared buffer is performed according to a first-in first-out (FIFO) discipline so that the original packet sequence is maintained.

The knockout switch architecture has low latency, and is self-routing and nonblocking. Moreover, its simple interconnection topology allows for easy modular growth from $N \times N$ to $JN \times JN$. One way to do this is illustrated in Figure 5.9 where $L$ additional inputs are added to

each concentrator for a total of $N+L$ inputs and $L$ outputs. The interface for each output in a $JN \times JN$ knockout switch consists of $J$ separate $N$-bus interfaces daisy chained together. Specifically, each of the $J$ interfaces for one output contains a row of $N$ packet filters and an $(N+L)$–*to*–$L$ concentrator, with only the first interface (for buses 1 — $N$) also containing the shared buffer structure with shifter and $L$ FIFO buffers. These $J$ individual components for each output are connected together by attaching the $L$ outputs of the concentrator in the $j$th interface ($j = 2, 3, \ldots, J$) to the $L$ extra inputs on the concentrator in the $j-1$st interface.

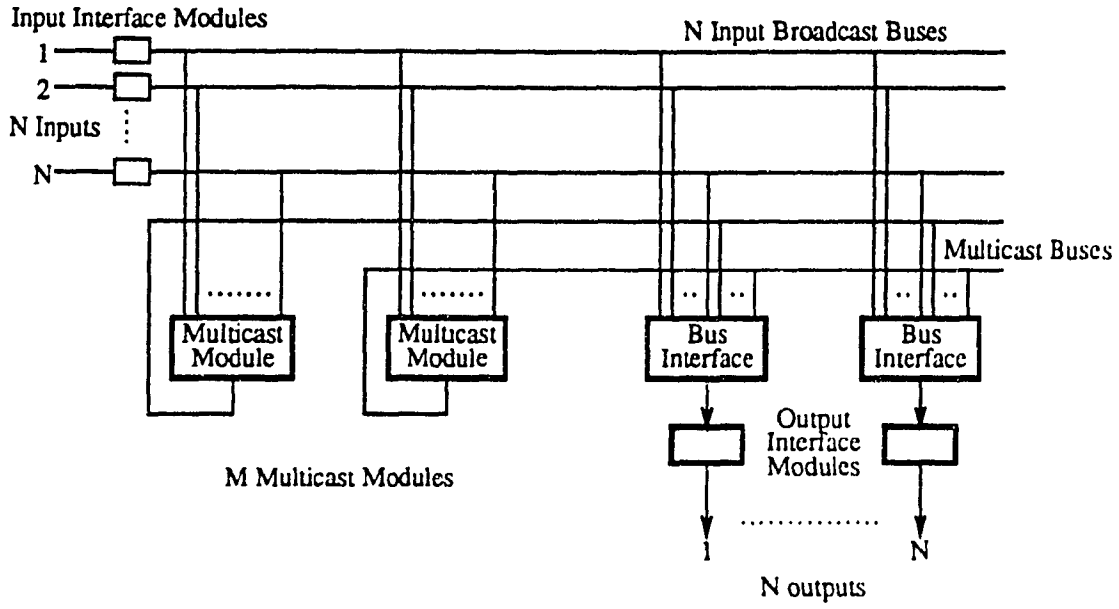



Figure 5.10. A knockout switch with multicast modules

*Multicast Operation* [16]

The modification necessary to support multicast operations is simple. In addition to the $N$ Bus Interfaces, $M$ multicast Modules specially designed to handle multicast packets are attached onto the input buses (Figure 5.10). As shown in the diagram, each Multicast Module has $N$ inputs and one output. The output drives one of $M$ bus wires as part of the arrangement for broadcasting to all the $N$ Bus Interfaces. Hence the Bus Interface has $N+M$ inputs instead of $N$.

A control header (or field) has to be added to each arriving packet in the Input Interface Module for the self-routing function. When a multicast packet arrives at an Input Interface Module, it is recognized and destined solely to a Multicast Module where processing will take place to relay this packet to the various outputs as intended. There are two different approaches to implement the Multicast Module as discussed separately in the following.

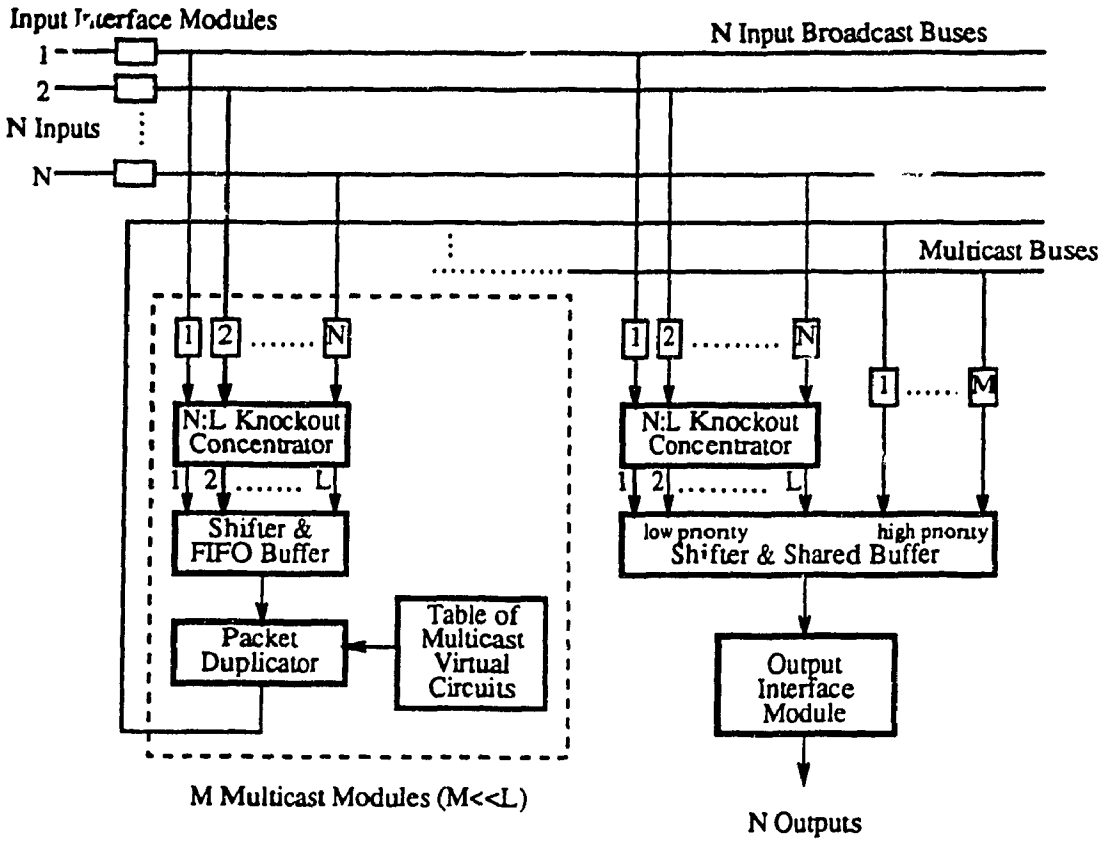



Figure 5.11. Multicast operation with packet duplicator approach

(1) The Packet Duplicator Approach

A block diagram of the Multicast Module with packet duplicator is illustrated in Figure 5.11. The incoming packets are selected through the packet filters which are set to accept multicast packets only. Upon exit from the FIFO buffer, a multicast packet enters a Packet Duplicator

whose function is to make duplicate copies of the packet with different destination addresses in the header and send them out along the broadcast bus shown in Figure 5.11, to the Bus Interface. The various destination addresses are obtained through a table look-up of the multicast virtual circuit. The duplicates should be allowed to bypass the knockout concentrator and proceed directly to the shared buffer.

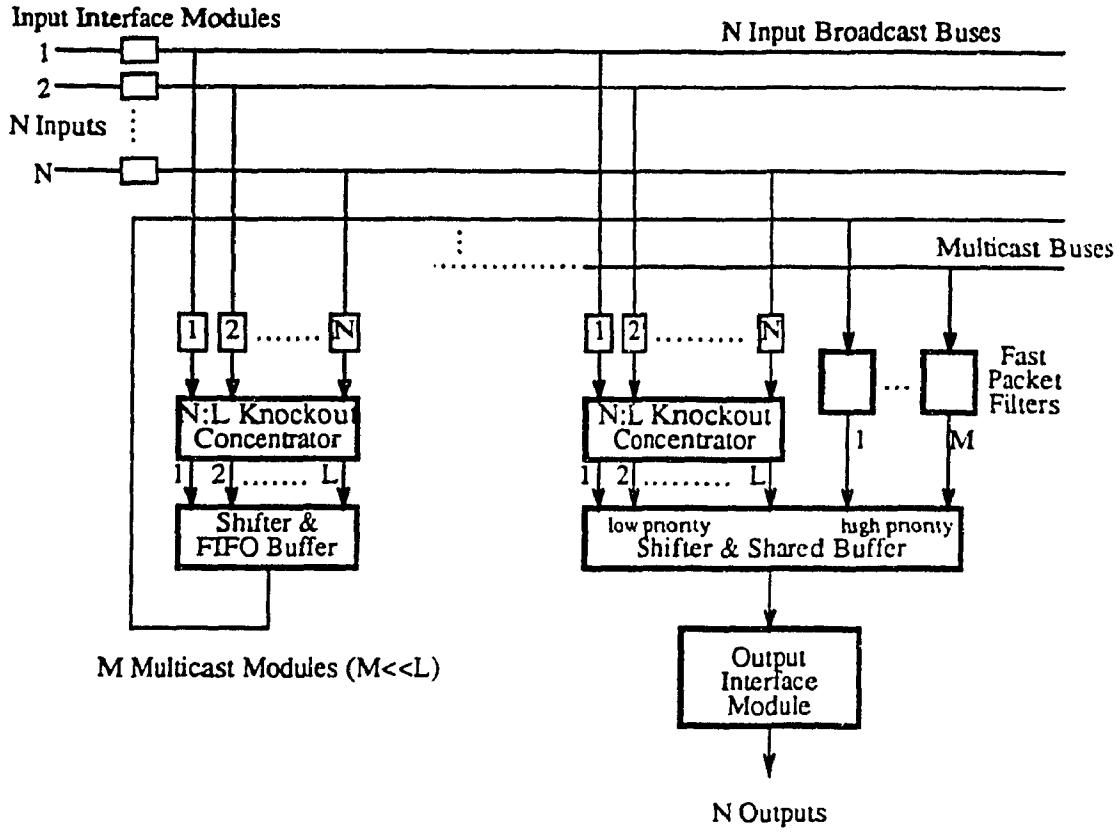



Figure 5.12. Multicast operation with fast packet filter approach

(2) The Fast Packet Filter Approach

This approach is based on the notion of a Fast Packet Filter capable of determining whether an incoming multicast packet is or is not destined for an output within a few bits intervals. As is shown in Figure 5.12, once a multicast packet emerges from the $N:L$ knockout concentrator as a

winner, it is buffered and then broadcasted directly to all the Bus Interface without any alteration. A list of multicast virtual circuits, each of which has this Bus Interface as a member of its multicast group, can be stored in the Bus Interface. The job is now to compare the multicast virtual circuit number in the header of an incoming packet to this list; if there is an agreement, the packet is accepted, otherwise the packet is discarded.

Both the Packet Duplicator and Fast Packet Filter methods can be implemented in a modular fashion. The Fast Packet Filter has low delay and is preferred in applications where the multicast traffic is heavy and involves a large number of simultaneous destination for each packet. However, the Packet Duplicator method seems to be more appropriate in situations where heavy multicast traffic is not demanded.

There are two drawbacks in this original model of knockout switch. The first is that the allowable packet loss in the concentrator may bring a significant effect, in case the lost packet happen to be a critical message. One way to prevent this mistake is to assign priority level to each of the packets when they were sent out according to their significance. Meanwhile, we need to introduce some facility to deal with packets of different priorities. Here, we suggest to use sorting network instead of knockout network to serve as a concentrator. An activity bit followed by several bits of priority index composes an address which is used for self-routing in the sorting network. A bus interface with such a sorter-type concentrator is illustrated in Figure 5.13, where a 8×8 sorting network not only performs as a concentrator but also endow the network with priority operation.

The second drawback is in the modular structure shown in Figure 5.9, where $J$ separate Bus Interfaces chained together. The $L$ outputs of the concentrator in $J$th interface will suffer extra delay when they arrive the $J-1$th interface. Identically, the $L$ outputs of the concentrator in $J-1$th interface suffer extra delay when they arrive the $J-2$th interface and so forth. So, address-filtered packets can not compete with each other fairly to come out $L$ winners. Especially when $J$ is very large, say 64 for example, the worst packet may suffers 63 concentrator-delay time units before it joins the competition with one lucky packet coming from buses 1 through $N$. One alternative

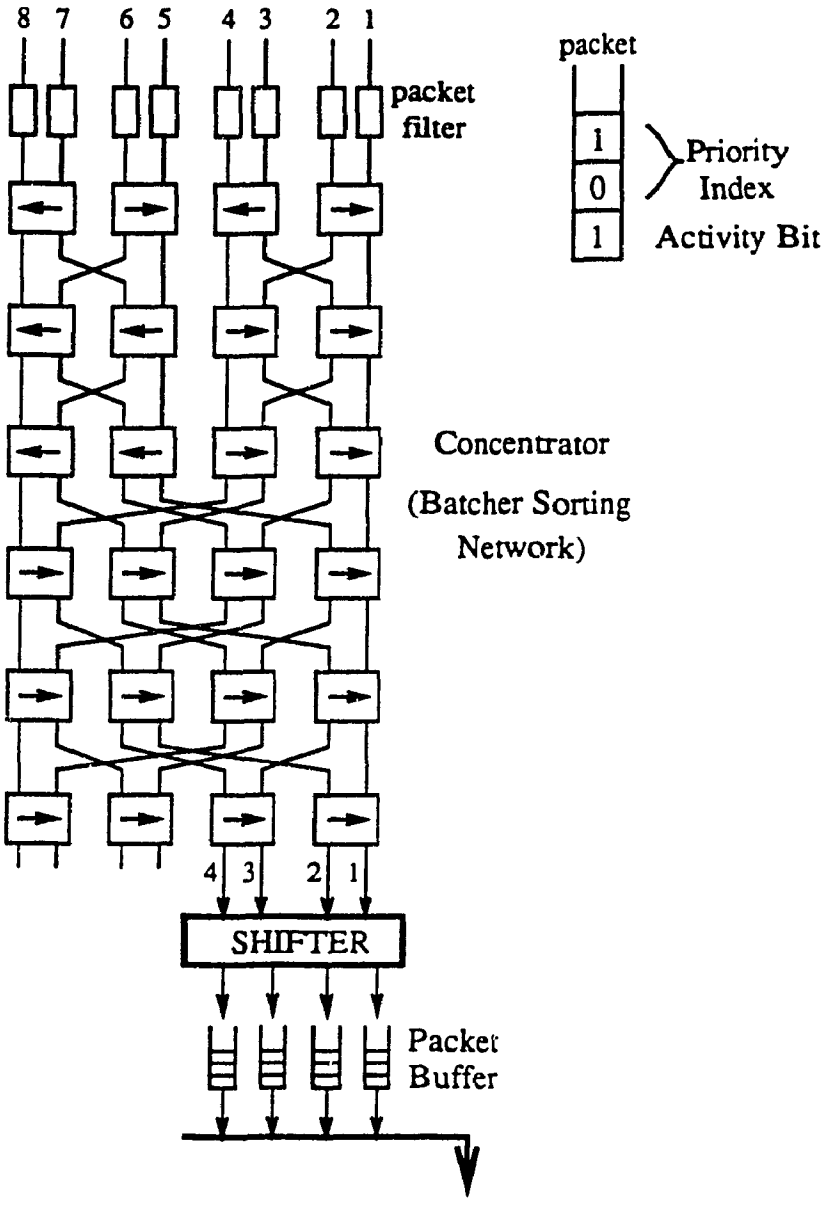



Figure 5.13. Bus interface with sorter-type concentrator

scheme to avoid this problem is to use two-stage concentrator without regard to $J$, as shown in Figure 5.14. All the output lines of the $J$ interfaces for one output are connected to a second-stage concentrator which is made of sorting network. For example, to construct a 1,024×1,024 switch, we choose $N = 32$, $J = 32$ and $L = 8$; hence 32,768 32-input bus interfaces, 1,024 256:8

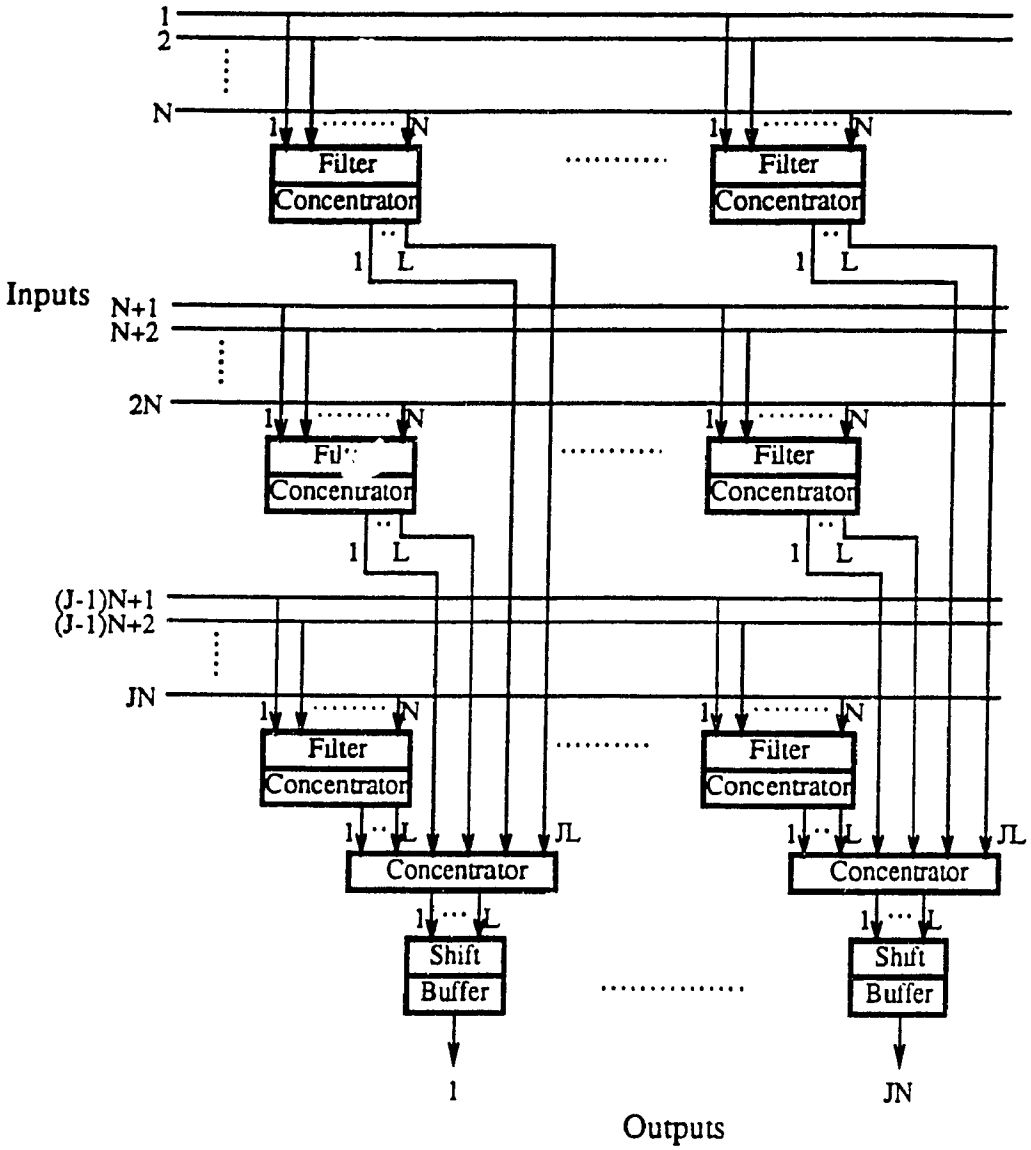



Figure 5.14. An alternative of modular structure of knockout switch

second-stage concentrators and 1,024 shifter-buffers are needed. The 256:8 concentrator may be also constructed by modular structure.

### 5.2.3. Shift Switch

The shift switch was recently reported in [17]. The idea is based on self-routing through

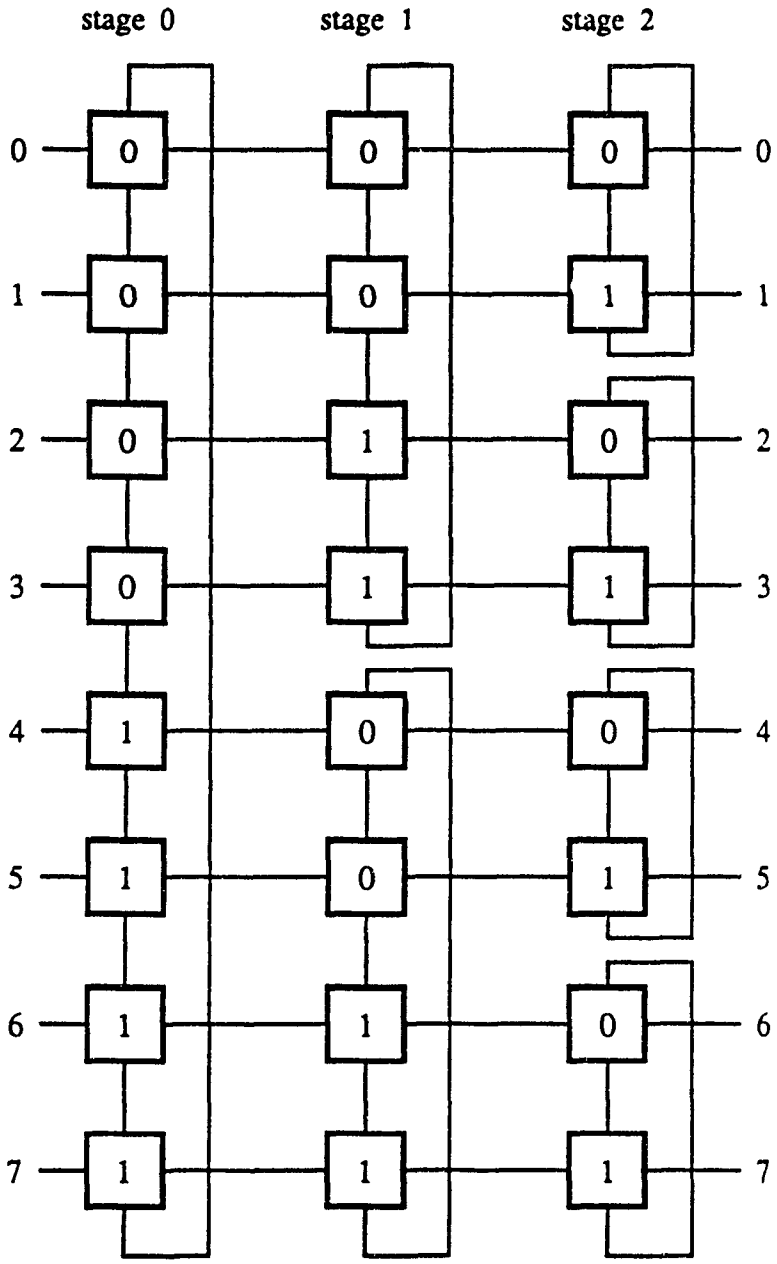



Figure 5.15. A shift switch

multistage circulated shift registers introduced by Imagawa [18]. The $N \times N$ shift network is a non-blocking multistage network with $\log_2 N$ stages and $N \log_2 N$ switching elements, as shown in Figure 5.15. Let $kl$ denote the label of switching element in the $k$th stage and the $l$th row. The

stages are labeled from 0 to $n-1$ where $n=\log_2 N$ and the rows are labeled from 0 to $N-1$. Each switching element has operational label either "0" or "1" obtained from $k+1$th bit of binary number $l$. The operational label is used in the routing of the packet. The switching elements are connected vertically in circular manner and horizontally to the next stage. The size of the rings in $k$th stage is $2^{n-k}$. Each switching element shifts the packet either horizontally or vertically according to the destination address bit.

The operation of the shift network is synchronous. Each input port accepts a packet at the beginning of each time slot. The slot time is divided into $N$ subintervals. The routing of the shift network does not require central control; it is done by each switching elements simply checking a bit of destination address and the counter in the packet header. Let $d_0 d_1 \cdots d_i \cdots d_{n-1}$ be the binary destination address of the network of size $2^n \times 2^n$. Each switching element in the $i$th stage has a label of "0" or "1" and it shifts the packet horizontally to the next stage if $d_i$ of the destination address matches with the label, or it shifts the packet $2^{n-1-i}$ times vertically if it does not match. It has been shown that no two packets from different input ports will be shifted to occupy any switching element even if the packets have the same destination address, as long as the number of subintervals is greater than or equal to $N$. The access to output buffering must be as $N$ times fast as the input/output lines. Therefore, there is no contention for the switching elements and no blocking occurs in the shift network.

The structure of the shift network allows implementation of multicast connection assuming the header of the packet contains addresses of the desired output ports. The shift network shifts the packet through the switching element in the same fashion as before but now a packet can be shifted vertically and horizontally at the same time. Figure 5.16 shows the routing operation of the multicast traffic where the input port 1 sends three copies of a packet to outputs 0, 3 and 7. This operation is also non-blocking since all the packet occupies the switching elements at distinct time. The fatal shortcoming of this multicasting scheme is that the routing of the multicast packet requires that the header of the multicast packet contains all the addresses of the desired output ports. It is not feasible if too many destinations are required by the packet due to too long

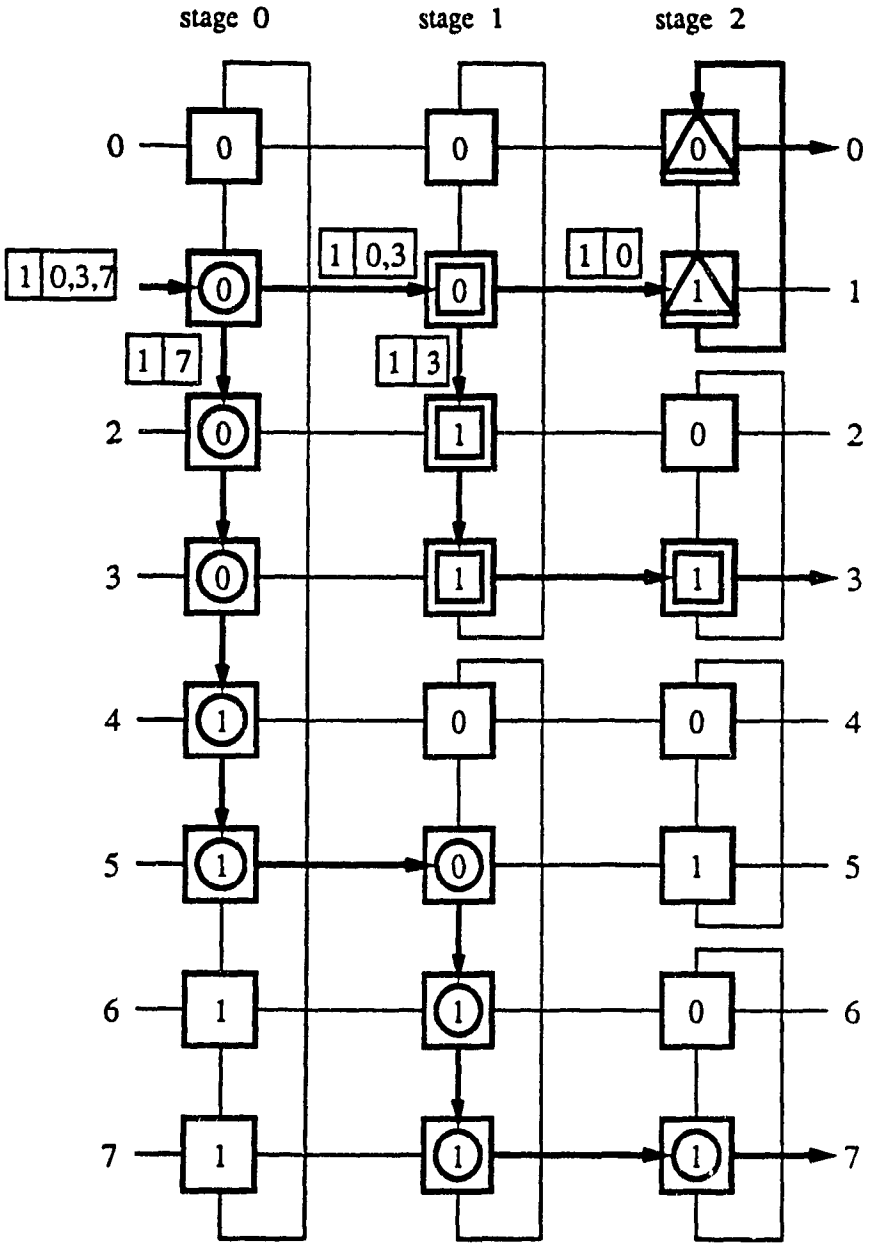



Figure 5.16. Multicast operation in a shift switch

header and complicated handling of the header.

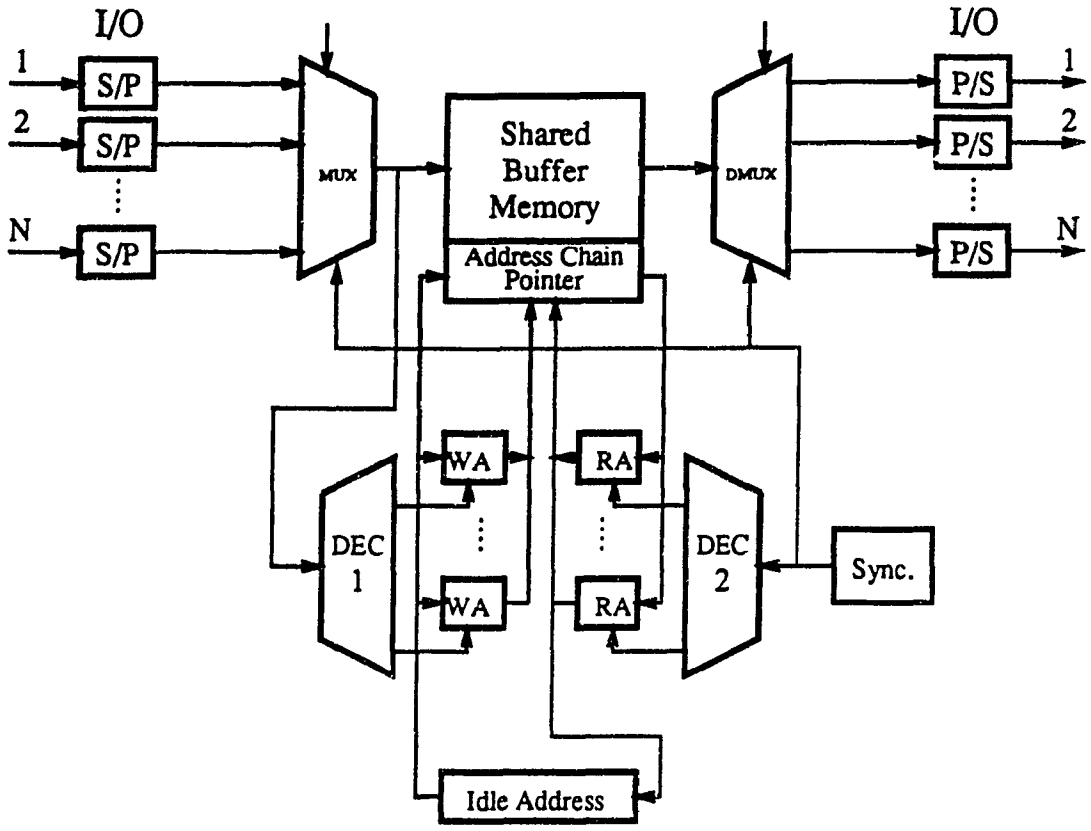



Figure 5.17. Kuwahara's shared buffer memory switch

### 5.2.4. Shared Buffer Memory Switch

Kuwahara suggested a shared buffer memory switch, the configuration and operation of which were described in great detail [7], as shown in Figure 5.17. The shared buffer memory switch mainly consists of a buffer memory, $N$ pairs of write address registers (WA) and read address registers (RA), an idle address buffer, and I/O controllers. Both the buffering and switching functions are carried out by reading from and writing to a memory. Incoming packets are converted from a serial data format to a parallel data format. They are then written one by one into the buffer memory according to the write address obtained from WA corresponding to the destination output port. An address pointer keeps track of locations of data and empty part of the buffer memory through the address chain as shown in Figure 5.18. Every packet is saved and

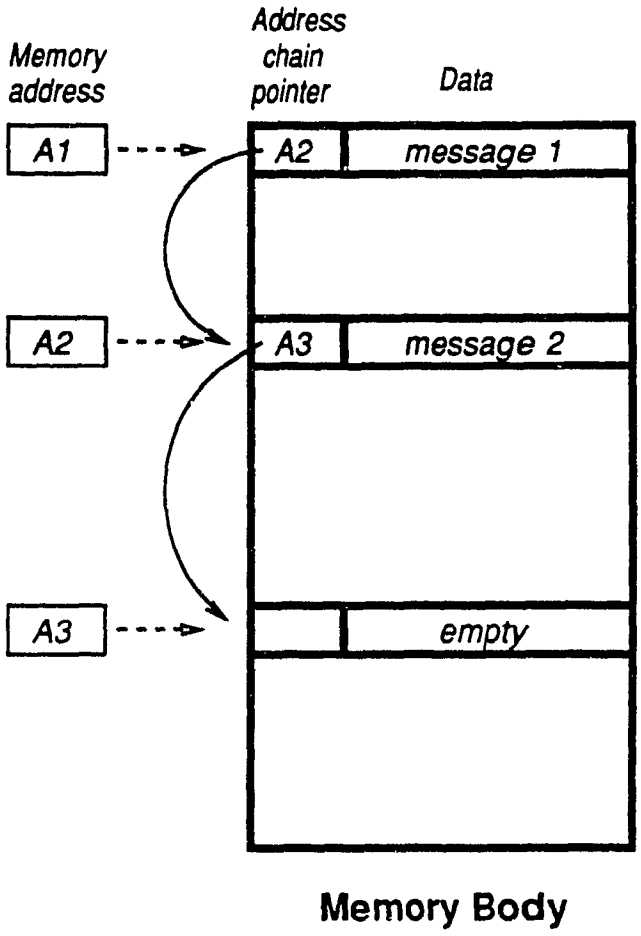



Figure 5.18. Output queue chain in shared buffer memory switch

accompanied by an address of the next packet in the queue. The last packet in the queue is accompanied by an address of an empty buffer to which the next incoming packet will be written. Packets are read out from the buffer memory, designated by RA, in FIFO fashion and then converted to a serial data format from a parallel data format. The addresses of removed packets are replaced by the next chain address by the address pointer for the next set of packets to be removed. WA and RA indicate the header and the end of the chain, respectively. This mechanism allows each buffer memory address to be allocated temporarily to any output port. Thus, all output ports share the buffer memory and required size of buffer memory is therefore reduced. It

is non-blocking, internally and externally, even though more than one incoming packets seek to the same output port.

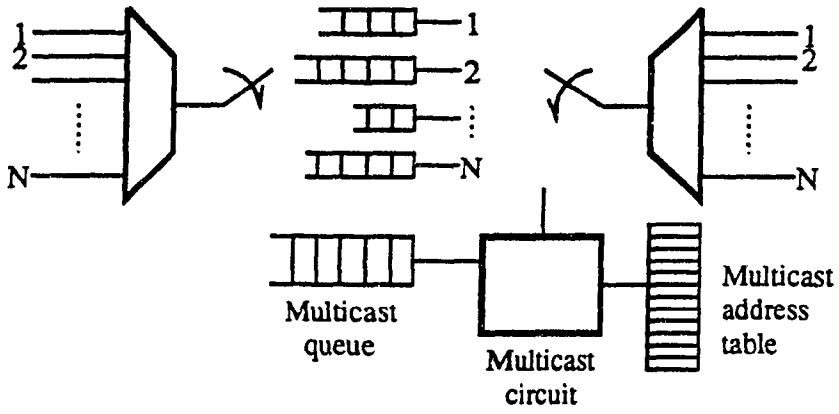



Figure 5.19. Multicast operation in shared buffer memory switch

The multicast function can be implemented with the addition of a multicast circuit, as shown in Figure 5.19, to the switch. The multicast packets queue up in a dedicated multicast queue when arriving at the input port of the switch irrespective of their destination output ports. The multicasting operation is achieved during the read-out procedure rather than write-in procedure, and has a higher priority than the unicast operation. The packet read-out from the buffer is carried out by first checking the multicast address, if it is active, then it is read out and the unicast queue is skipped; when it is inactive, it enables the unicast queue to be read-out. All of the copies of the multicast packet will go through in a single slot. But, only one multicast packet could go through in a time slot. Therefore, multicasting arrival rate can not exceed $1/N^2$ for each input line. The switch seems to be applicable only to the cases where the multicast traffic is light. The poor throughput for multicast traffic may be a main drawback of Kuwahara's shared buffer memory switch. Another drawback of this structure is the possible overloading of the shared buffer by a few output queues. In the following section, we will discuss the memory type switch and propose a novel structure to meet a more sophisticated traffic requirement.

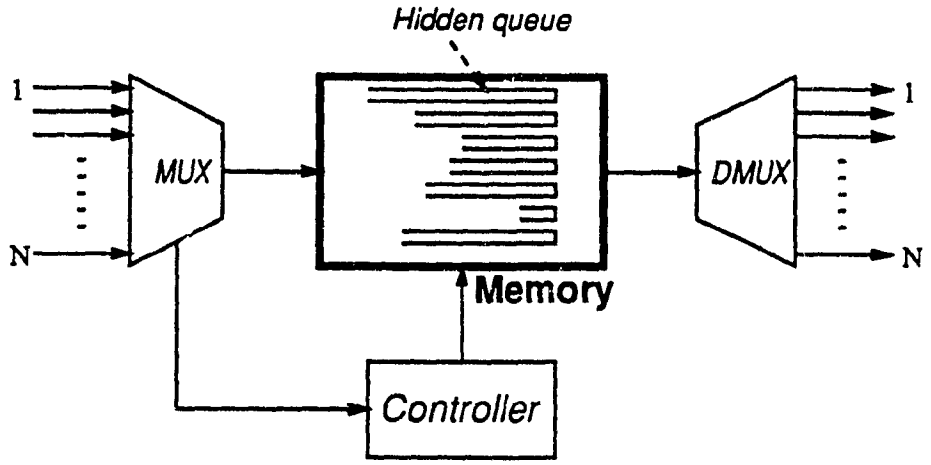



Figure 5.20. Concept of a memory-type switch

## 5.3. A SHARED BUFFER MEMORY SWITCH WITH MAXIMUM QUEUE AND MINIMUM ALLOCATION

Buffering is required in any type of packet switch structure in order to avoid packet loss whenever there are multiple input packets arriving for the same output. The buffer memory could be located either at the input or at the output or at the internal stage of the switch. In memory-type switch, both the buffering and switching functions are carried out by reading from and writing to a memory. The concept of a memory-type switch is illustrated in Figure 5.20. The packet from all input ports are multiplexed and written one-by-one into the memory. The write address is designated by the controller according to the destination of the packet. The HOL packets are read out from the memory and sent out through demultiplexer.

Of the various ATM switches as proposed to date, the memory-type switch seems to have the highest efficiency in hardware-utilization and to have the potential to provide good traffic characteristics especially under unbalanced and bursty traffic. The implementation achievements were reported by Hitachi and Toshiba in [19][20]. The memory-type switch can fully utilize its available bandwidth. There is no blocking caused by output port contention.

### 5.3.1. Classifications of Shared Memory Switches

We may classify the memory-type switches from the point of view of the memory sharing schemes [21]. The five possible schemes are summarized as follows.

1) Complete partitioning: The entire finite memory is permanently partitioned among the $N$ output ports hence no sharing is provided. One example of this scheme is ATOM switch proposed by Suzuki [22]. The switch uses dedicated output buffers for each output port. Packets arriving at input ports are converted from serial to parallel and transferred to the data bus using time-division multiplexing. The packets are caught by corresponding address filters and are buffered and read out on a FIFO basis. The drawback of this scheme is that the switch requires a very large memory to attain a low packet loss probability.

2) Complete sharing: It is the other extreme to the complete partitioning in that an arriving packet is accepted if any memory space is available, independent of the destination to which it is directed. One example is Kuwahara's shared buffer memory switch [7], summarized in Section 5.2.4. Complete sharing has a better performance (smaller probability of blocking) than complete partitioning under normal traffic conditions with fairly balanced input systems. But, highly asymmetrical input traffic may overload the shared buffer with a few output queues. In particular, with bursty traffic it is likely for a few output queues to become momentarily heavily loaded and to monopolize the use of the shared buffers, to the detriment of other output queues, for some period of time following the congestion. Moreover, even with perfectly balanced arrival rates, under overload conditions, complete sharing scheme fails in securing a full utilization of all the $N$ output channels.

3) Sharing with maximum queue length: This scheme is promoted by above considerations. In order to avoid the possible utilization of the entire space by any particular output queue, we impose a limit on the number of buffers to be allocated at any time to any output queue. Of course, the sum of the imposed limits on these output queues must be greater than the capacity of the memory if some sharing is to be provided. Fried [23] proposed an implementation of such scheme. An incoming packet is placed into a single physical memory and the

memory address is then placed into a FIFO buffer which is dedicated to the output port addressed by the packet. But, this scheme still does not guarantee a full utilization of the output channels under heavy traffic conditions.

4) Sharing with a minimum allocation: This scheme is motivated by the consideration of utilization of the output channels. A minimum number of buffers is permanently reserved for each output port and, in addition, a common pool of buffers is to be shared among all output queues, with no further constraints on the queue size. But again, with this scheme, the shared area tends to be unfairly utilized as mentioned earlier.

5) Sharing with maximum queue and minimum allocation. Actually, this is a combination of last two schemes. The memory is shared with maximum and minimum allocation constraints for each queue. It appears to be an excellent sharing scheme which has the ability to avoid the deficiencies of the above four scheme.

Here, we proposed a shared buffer memory switch structure, which employs the fifth sharing scheme in which a shared buffer pool and a reserved buffer pool are deployed for switching and buffering the packets. Section 5.3.2 gives a detailed description of the proposed switch structure. We will roughly estimate the size of buffer memory required for certain packet loss probability under both uniform and nonuniform traffic pattern in Section 5.3.3. As a vital switch for BISDN, its capabilities of multicasting, modularity and priority are also of considerable interest. These functions are implemented quite easily without inherently changing the switch architecture, as will be seen in Section 5.3.4.

### 5.3.2. The Switch Architecture

This new design was influenced by the work in [23] and extended to fit the fifth sharing strategy. The same concept can be found in [24] as queue management scheme.

The structure of the proposed switch is depicted in Figure 5.21, which consists of a buffer memory pool (BMP), a FIFO queue recorder (QR), an idle memory address pool (IMAP), a destination address decoder (DEC), and I/O interfaces. BMP is composed of $B$ cells each of which is

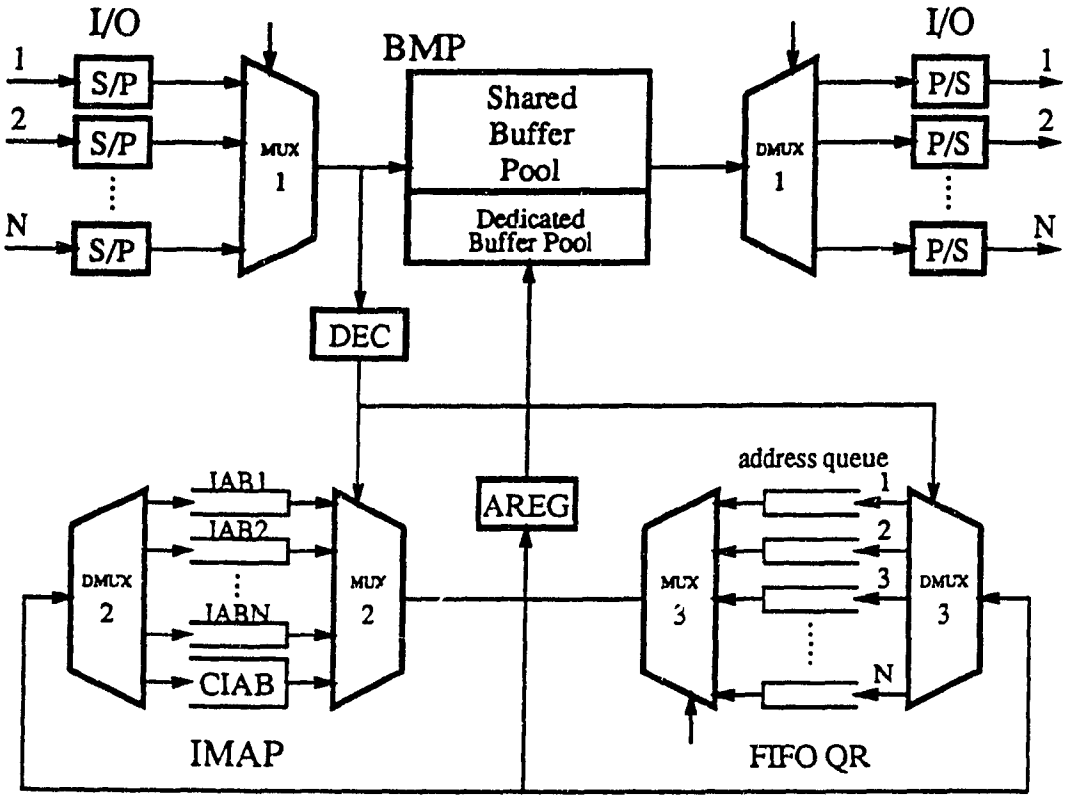



Figure 5.21. A shared buffer memory switch with maximum queue and minimum allocation

designed to hold a whole packet. Among $B$ cells (refer to Figure 5.22), $B_1$ cells are allocated to share among the users, hence they form a shared buffer pool. The remainder $B_2$ cells constitute a dedicated buffer pool which is equally partitioned into $N$ parts permanently reserved by $N$ users. Of course, $B_1 + B_2 = B$. The users of BMP are the $N$ output ports. To comply with the FIFO service discipline, a FIFO QR which contains $N$ FIFO address queues, is needed, and each buffer has $K$ cells, where $K$ is the maximum length of each output port queue, as depicted in Figure 5.23. Each cell in QR is designed to hold an address in BMP. The addresses in which packets are stored in BMP are kept in QR, in the order of their arrival and corresponding to their destinations. An IMAP provides addresses for incoming packets. The addresses of the idle dedicated

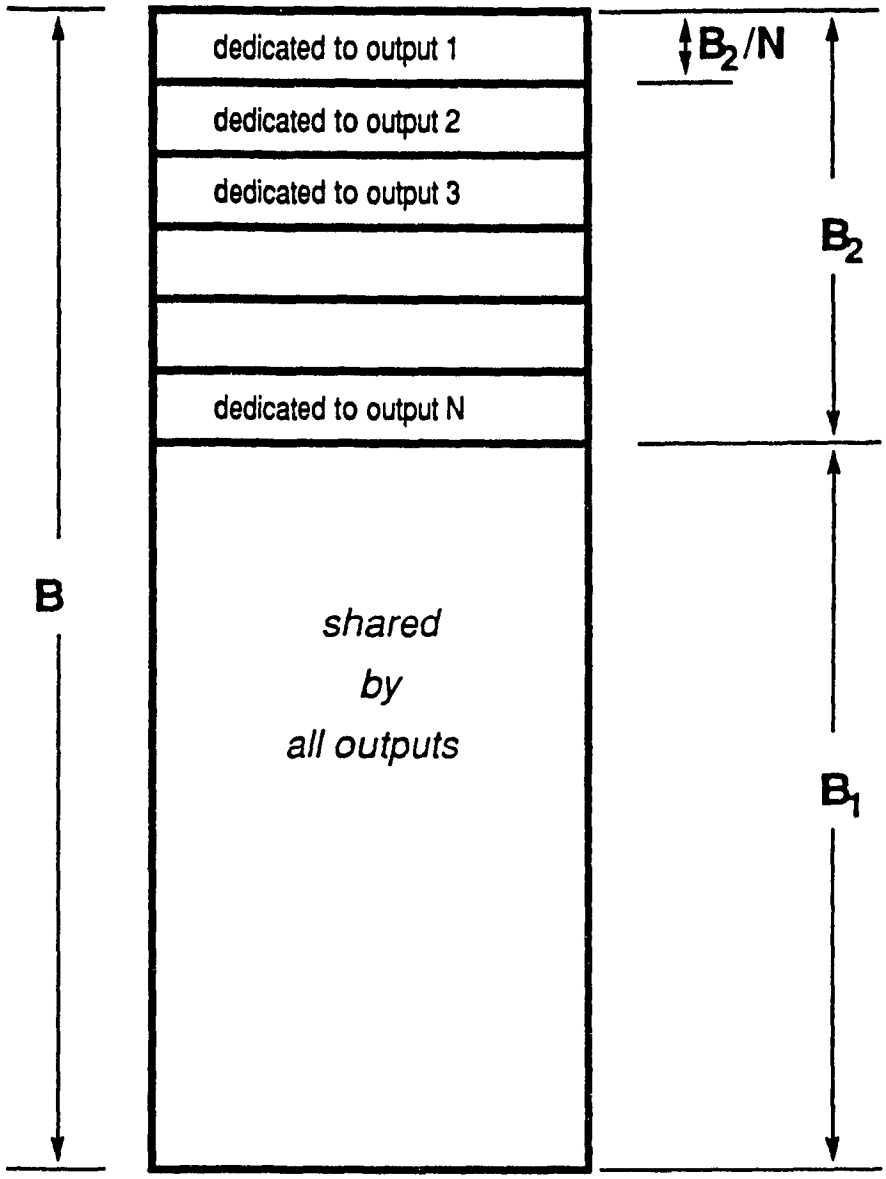



Figure 5.22. Memory allocation in proposed switch

memory cells for certain outputs are kept in corresponding idle address buffer (IAB) which has $B_2/N$ cells. The addresses of the idle shared memory cells are kept in a common idle address buffer (CIAB) holding a maximum of $B_1$ addresses. The $N+1$ separated buffers in IMAP could be operated in either FIFO fashion or LIFO fashion. The operation of this switch structure is as

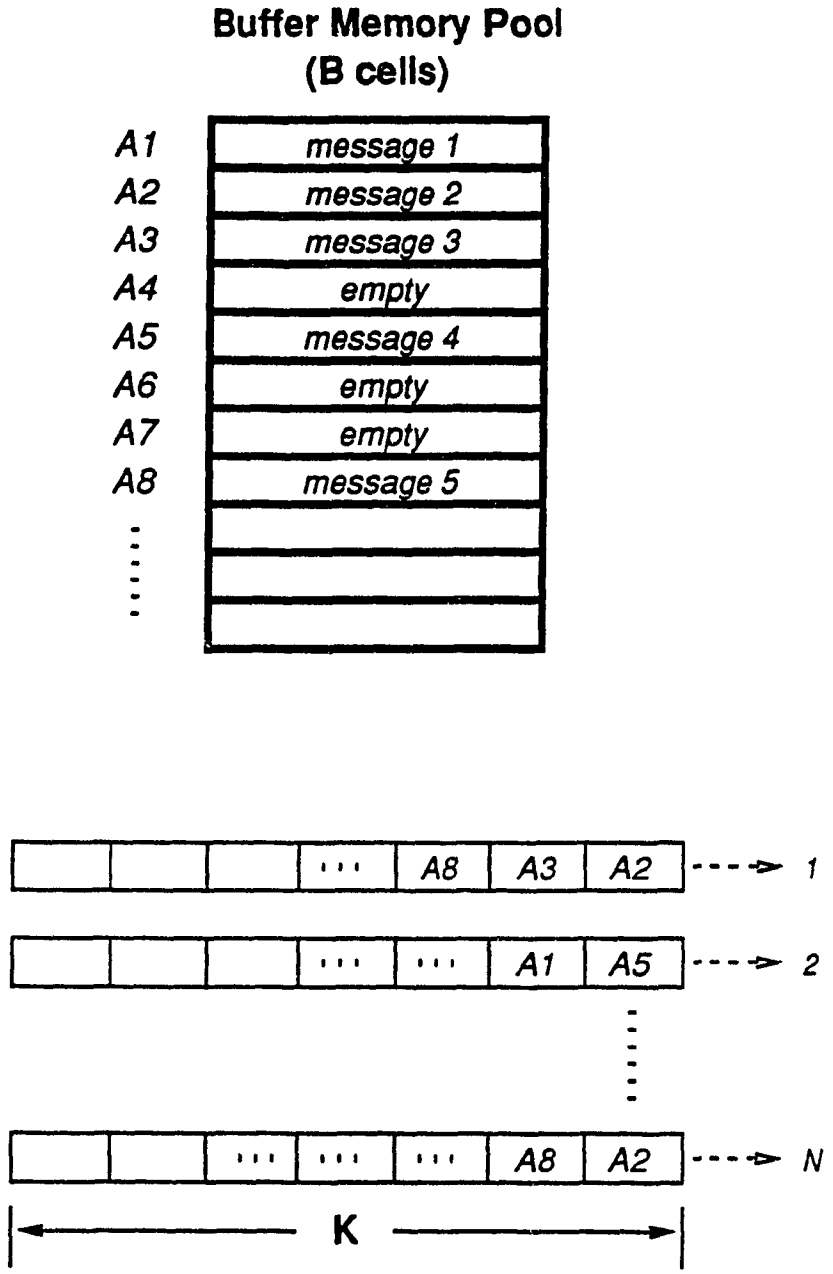



Figure 5.23. FIFO address queue

follows.

At each ATM time slot during which up to $N$ packets may enter the switch and up to $N$ packets may depart, there are $2N$ times of access operation, either write-in or read-out, on BMP.

We divide a time slot into $N$ subslots during each of which a pair of write-in and read-out is implemented in sequence. In the write phase of $i$th subslot ($i$ = 1, 2, ..., $N$), an incoming packet from input port $i$ through MUX1 is placed in the data bus. If the address queue in FIFO QR corresponding to the requested output port (say $j$) of the incoming packet, is full (contains maximum $K$ addresses) at the moment, the writing operation is terminated and the incoming packet will be lost; otherwise the packet will be stored in BMP at the address specified by address register (AREG). The write address is obtained from the idle address pool, corresponding to the requested output port $j$ of the incoming packet. If there is any idle dedicated memory cell hence corresponding address remains in IAB$j$, the address is placed into AREG; if IAB$j$ is empty, then one idle address in CIAB is chosen to be sent to AREG; if both IAB$j$ and CIAB are empty, then no address and no writing are expected in this phase. When the packet is written into the memory, its address is simultaneously sent into QR and lines up at address queue $j$. The control signals are obtained from DEC by decoding the destination address. In read phase of $i$th subslot, an address in the header of address queue $i$ in QR is placed into AREG; accordingly, a packet stored in this address is read out then sent to output port $i$ through DMUX1. Meanwhile, the cell from which the packet is read out becomes idle, its address is sent back to IMAP and kept in IAB$i$ if the cell is dedicated for output port $i$, or in CIAB otherwise. The process is repeated in each subslot as well as in each ATM slot.

Instead of designating the dedicated buffer memory pool and $N$ separate IAB's, another possible implementation could be a single shared buffer memory pool and a single idle address buffer with a control algorithm to ensure the minimum allocation to each output queue. Each output address queue has a flag to indicate whether the queue contains minimum allocated number of addresses. We introduce two counters, one records the number of idle addresses; another records the required number of buffers for the minimum allocation. The second counter set to $B_2$ in the beginning is decreased by 1 in the writing phase or increased by 1 in the reading phase, only if the corresponding address queue's flag indicates that the minimum allocation has not been reached. If counter 1 is less than counter 2, the remaining idle buffers are reserved by those out-

put queues where the minimum allocation has not been reached; the incoming packets destined for the other outputs will lose.

### 5.3.3. Estimation of Packet Loss Probability

For a switch with an output queue, the main issue is the packet loss probability instead of delay-throughput relation. In this section, we derive the relation between packet loss probability and the buffer sizes, under both uniform (balanced) and nonuniform (unbalanced) traffic patterns. We first give general equations which can be used to estimate the packet loss probability in terms of $a_i$, the probability that $i$ packets arrive at an output queue in a given time slot.

The packet may suffer two types of losses in the proposed switch. One is due to the restriction of length of the address queue for each output port. A packet destined for output port $i$, finding that the output queue $i$ is full with $K$ packets, will lose whether the shared memory resource is available or not. The other is caused by the saturation of the shared buffer memory pool such that no cell is available for the incoming packet even if the corresponding output queue may be free.

We first calculate the packet loss probability due to the maximum queueing size $K$ at each output port by using the discrete time Markov chain model. Let $p_n$ be the probability that there are exactly $n$ packets in each output queue at the end of a slot interval. We have [25][26]

$$p_n = \frac{1}{a_0}\left[p_{n-1} - \sum_{i=1}^{n-1} a_{n-i+1}\, p_i - a_{n-1}\, p_0\right] \qquad n = 1,2,\cdots K \tag{5.1}$$

where $p_0$ must be such that

$$\sum_{n=0}^{K} p_n = 1 \tag{5.2}$$

This suggests a simple procedure to find $p_0, p_1, \ldots, p_K$. We initialize $p'_0 = 1$ and find $p'_1, p'_2, \ldots, p'_K$ from equation (5.4). The final values are found from normalization

$$p_n = \frac{p'_n}{\sum_{j=0}^{K} p'_j} \qquad n = 0,1,\ldots,K \tag{5.3}$$

Since the queue has a finite size of $K$, when a packet arrives and the queue is full, an overflow will result. Consequently, the average packet throughput $\rho_0$ is less than the average packet arrival rate $\lambda$. The average packet throughput can be computed from the probability that the queue is idle

$$\rho_0 = 1 - p_0 \tag{5.4}$$

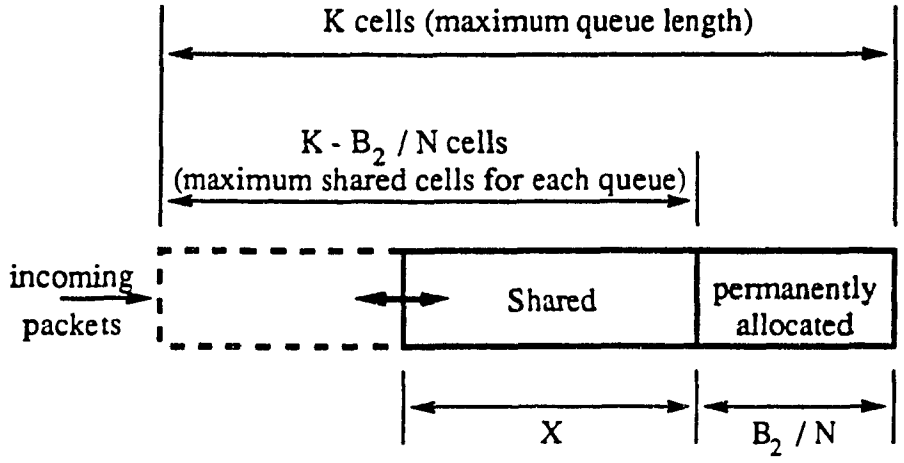



Figure 5.24. Memory resource used by an output queue

The loss probability due to $K$, or the overflow probability of the queue can be expressed as the average fractions of the total number of arriving packets rejected by the output queue in QR, which is equal to

$$Loss_K = \frac{arrival\ rate - throughput}{arrival\ rate} = \frac{\lambda - \rho_0}{\lambda} = 1 - \frac{1 - p_0}{\lambda} \tag{5.5}$$

We now tend to find the packet loss probability caused by the limited capacity of the shared buffer pool. Let $X_i$ be the number of cells in the shared buffer pool taken by the packets at output queue $i$ as depicted in Figure 5.24. The number of packets in the shared memory pool, denoted by $Y$, is the sum of $N$ random variables $X_i$ for $N$ output queues.

$$Y = \sum_{i=1}^{N} X_i \tag{5.6}$$

It is assumed that $N$ random variables $X_i$ are independent of each other. The packet loss probability due to the size of shared memory pool is determined by the traditional approach of truncating the tail of the distribution function [27]. The Chernoff bound is available for bounding the tail of the sum of a large number of independent random variables [28][7]. Assume that all of the $X_i$ are identically distributed. Let $M_X(v)$ be the moment generating function of $X$, given by

$$M_X(v) = E[e^{vX}] \tag{5.7}$$

Suppose there are $n$ packets ($n = 0, 1, ..., K$) in each output queue with the probability $p_n$, since the packets first take up dedicated buffer memory then take up the shared buffer memory after the dedicated buffers are used out, we have $X_i = max(\, n - B_2/N,\, 0\,)$. The probability distribution of $X_i$, denoted by $q_j$, is found by

$$q_0 = \sum_{i=0}^{B_2/N} p_i \tag{5.8}$$

and

$$q_j = p_{j+B_2/N} \qquad j = 1,2,....,K - B_2/N \tag{5.9}$$

Thus, the moment generating function of $X$ becomes

$$M_X(v) = \sum_{i=0}^{K-B_2/N} e^{vi}\, q_i \tag{5.10}$$

Furthermore, we define the semi-invariant generating function

$$\gamma_X(v) = \log_e M_X(v) \tag{5.11}$$

The Chernoff bound for the tail of a density function takes the final form as

$$Pr[Y \geq N\gamma_X^{(1)}(v)] \leq \exp[N(\gamma_X(v) - v\gamma_X^{(1)}(v))] \tag{5.12}$$

where $v \geq 0$, and $\gamma_X^{(1)}(v)$ is the first derivative of $\gamma_X(v)$ with respect to $v$. Let $N\gamma_X^{(1)}(v) = B_1$, $v$ might be calculated. Substituting this value back into equation (5.12), we can obtain the upper bound of the loss probability due to $B_1$, denoted by $Loss_B$. As we have noted, $X_i$'s are actually not independent of each other since they share a common limited source. The summation of $X_i$'s is actually less than $Y$ obtained by equation (5.6). Therefore, the upper bound remains valid and

may be a little bit pessimistic.

Eventually, the union upper bound of the packet loss probability can be obtained by

$$\begin{aligned} Loss &= Pr[Loss\ due\ to\ B_1\ or\ K] \\ &\le Pr[Loss\ due\ to\ B_1] + Pr[Loss\ due\ to\ K] \\ &= Loss_B + Loss_K \end{aligned} \tag{5.13}$$

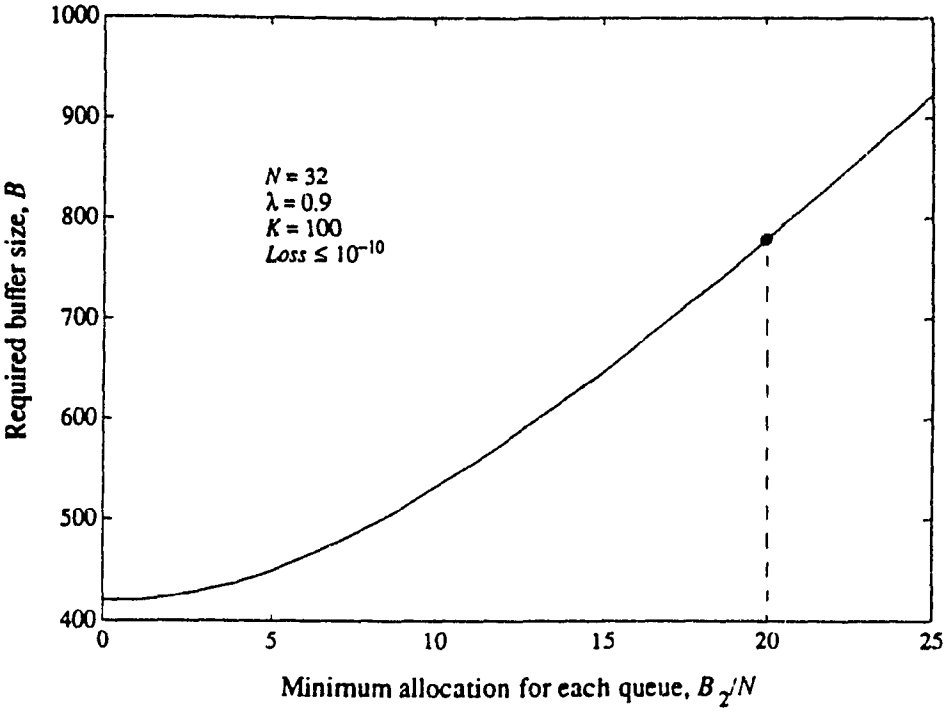



Figure 5.25. Buffer requirement vs. $B_2/N$

### 5.3.3.1 Balanced Traffic

For balanced traffic, we assume that in each time slot a packet arrives at each input port with the same probability $\lambda$, and packets are distributed uniformly to the $N$ output ports. Thus, the probability of $i$ packets arriving at an output port, $a_i$, is given by the following binomial distribution

$$a_i = \binom{N}{i} \left(\frac{\lambda}{N}\right)^i \left(1 - \frac{\lambda}{N}\right)^{N-i} \qquad i = 0, 1, \cdots N \tag{5.14}$$

which, for $N=\infty$, becomes

$$a_i = \frac{\lambda^i e^{-\lambda}}{i!} \qquad i = 0,1,\cdots,N=\infty \tag{5.15}$$

Substituting equation (5.14) into equation (5.1) and following the steps discussed above, we find the upper bound of the packet loss probability. The relationship between the buffer size $B$ and the packet loss probability *Loss* for various sharing schemes is shown in Figures 5.25 through 5.27. Several conclusions can be drawn as follows:

(i) To attain a certain value of packet loss probability, sharing schemes require less memory than complete partitioning scheme. For example, under the condition of packet loss probability $Loss \le 10^{-10}$, given load $\lambda = 0.9$, switch size $N = 32$, maximum queue length $K = 100$, and no permanent allocation $B_2 = 0$, such sharing scheme requires 420 buffer cells, which is only 13.2% of the buffer size (3200 buffer cells) of the complete partitioning scheme, as shown in Figures 5.25 and 5.26.

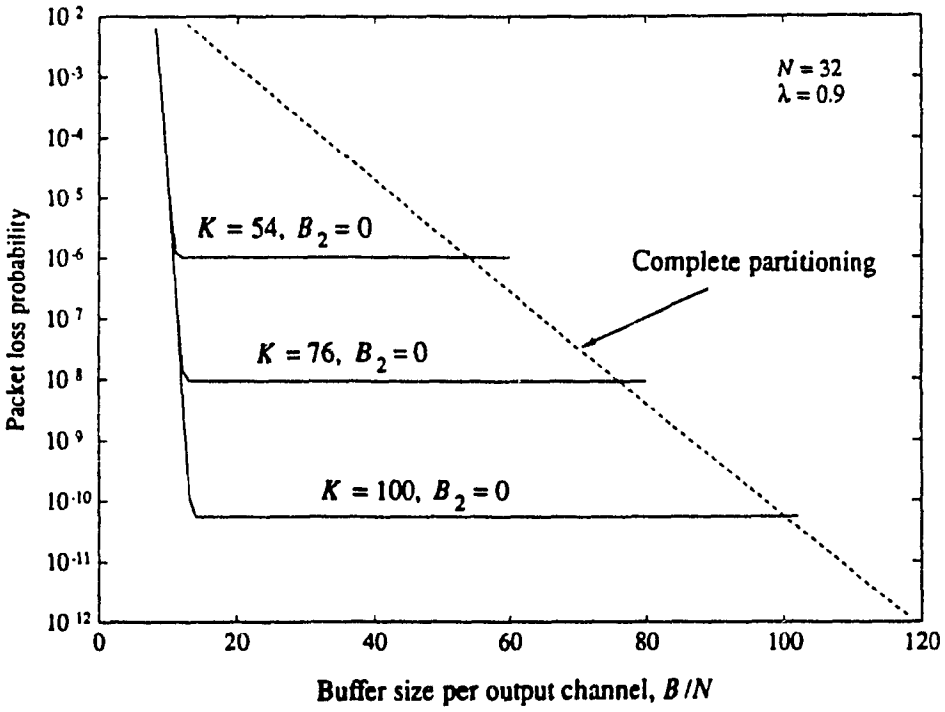



Figure 5.26. Packet loss probability vs. buffer size for sharing with maximum queue length

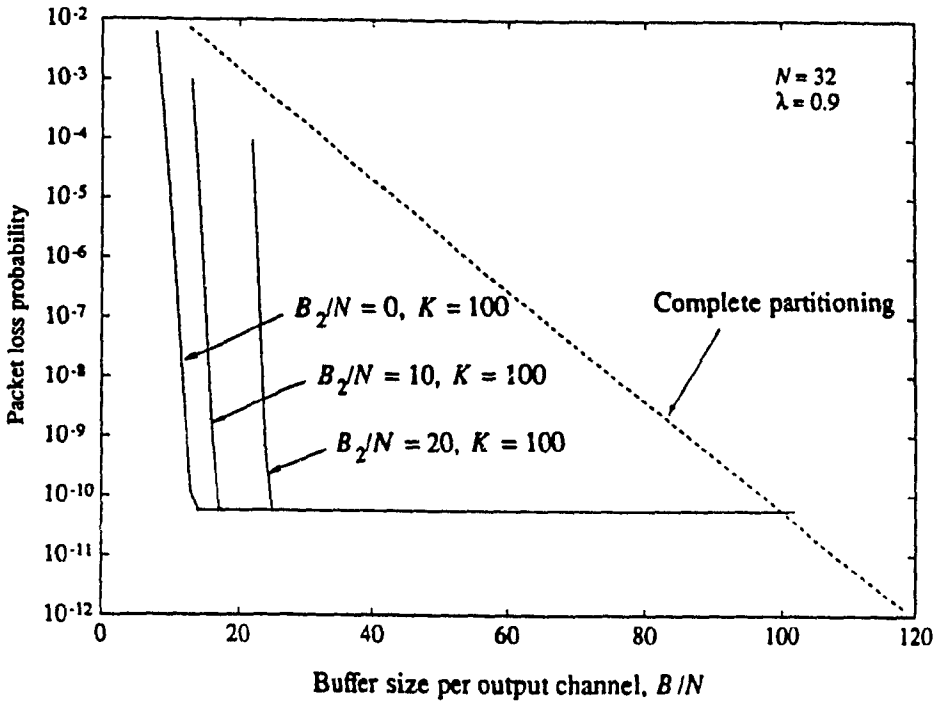



Figure 5.27. Packet loss probability vs. buffer size for sharing with maximum queue and minimum allocation

(ii) For the sharing scheme, under the same condition except changing the maximum queue size $K$, we find a saturation point of packet loss probability for each value of $K$, as depicted in Figure 5.26. It is because at this point, $K$ becomes the main factor of the packet loss, we can not decrease the packet loss probability by increasing the buffer size $B$ only, and vice versa. Therefore, when we design the switch, we may first select $K$ to satisfy the requirement of packet loss probability, for example enable $Loss_K \le \frac{1}{2} Loss$; then estimate $B$ to ensure $Loss_B \le \frac{1}{2} Loss$, given parameter $K$.

(iii) To make full use of output links, a set of $B_2/N$ cells is permanently allocated to an output port, which is a portion of the output queue. Obviously, with the increase of $B_2/N$, an increase of the buffer size is required, as shown in Figure 5.27. For example, under the conditions that $Loss \le 10^{-10}$, $\lambda = 0.9$, $N = 32$, $K = 100$ but $B_2 = 20$ rather than $B_2 = 0$, the sharing scheme requires 780 cells (refer to Figure 5.25), which is 24.4% of the buffer size of the complete

partitioning scheme. The problem remaining under study is how to evaluate the effect of $B_2/N$ numerically and how to choose it.

### 5.3.3.2 Unbalanced Hot-Spot Traffic

Hot-spot traffic refers to a traffic pattern where many sources try to communicate with one destination (hot spot) virtually at the same time. We use the hot-spot traffic model described in [29][30]: an unbalanced traffic pattern consisting of a single hot spot of a higher access rate superimposed on the background of a balanced traffic. Let $h$ be a fraction of packets directed to the hot-spot. Then, $\lambda$, the packet arrival rate on an input port can be expressed as $\lambda = h\lambda + (1-h)\lambda$; i.e. $h\lambda$ packets are directed to the hot-spot and $(1-h)\lambda$ packets are directed uniformly to the $N$ outputs. With this hot-spot traffic model, the probability that $i$ packets arrive at the hot-spot is given by

$$a_i(h) = \binom{N}{i} \left[ h\lambda + \frac{(1-h)\lambda}{N} \right]^i \left[ 1 - h\lambda - \frac{(1-h)\lambda}{N} \right]^{N-i} \qquad i = 0,1,\cdots,N \quad (5.16)$$

and the probability that $i$ packets arrive at any one of the other output ports is given by

$$a_i(\bar{h}) = \binom{N}{i} \left[ \frac{(1-h)\lambda}{N} \right]^i \left[ 1 - \frac{(1-h)\lambda}{N} \right]^{N-i} \qquad i = 0,1,\cdots,N \quad (5.17)$$

We can substitute $a_i(h)$ into equation (5.1) to obtain $p_0$ then estimate the $Loss_K$ from equation (5.5), for hot-spot traffic. Alternatively, we can find required queue length $K$ for hot-spot traffic under a certain packet loss probability.

There are two kinds of traffic distributed over $N$ output queues, which complicate the analysis. Therefore, we are going to give a looser upper bound by assuming that the hot-spot traffic has taken maximum $K$ buffer cells, and the remaining buffer cells are shared between the other $N-1$ independent output queues to accommodate the non-hot-spot traffic. Thus, the equation (5.12) becomes

$$Pr[Y \geq (N-1)\,\gamma_X^{(1)}(v)] \leq \exp[(N-1)\,(\gamma_X(v) - v\,\gamma_X^{(1)}(v))] \quad (5.18)$$

where $\gamma_X(v)$ refers to the non-hot-spot traffic. Let $((n-1)\,\gamma_X^{(1)}(v) = B_1 - (K - B_2/N)$ which is the number of buffer cells available for sharing between the $N-1$ output ports, we can obtain the $v$

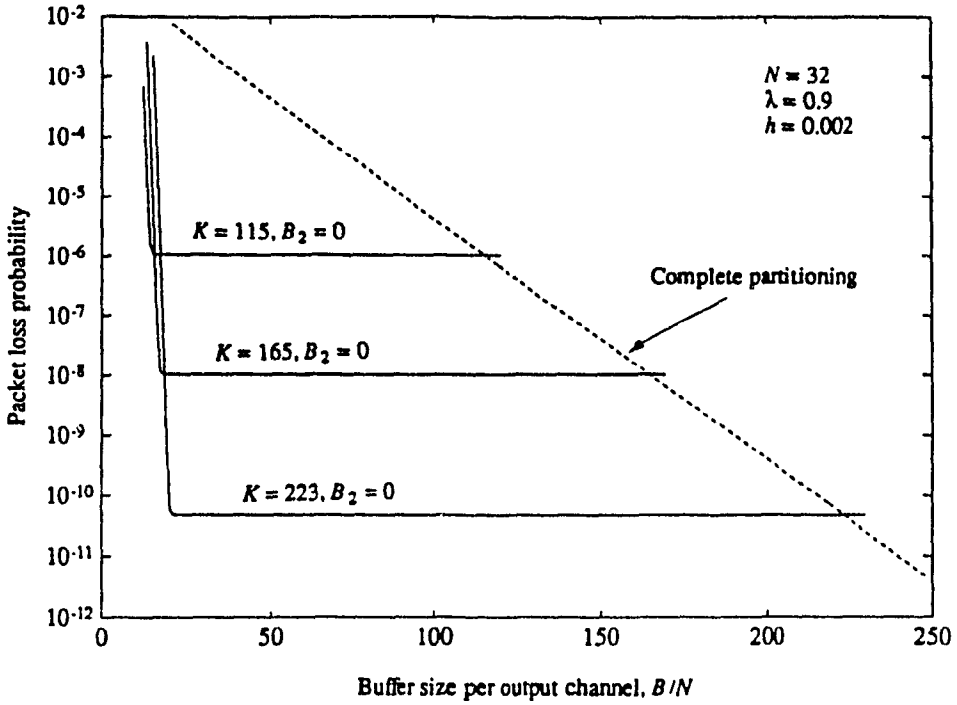



Figure 5.28. Packet loss probability vs. buffer size for sharing with maximum queue length under unbalanced traffic

then the $Loss_B$.

The relationship between the packet loss probability and the memory size is plotted in Figures 5.28 and 5.29, where $h = 0.002$ corresponding to a slight unbalanced traffic. The required memory resource under a certain packet loss probability is more than two times as what for balanced traffic. Under the conditions that $Loss \leq 10^{-10}$, $h = 0.002$, $\lambda = 0.9$, $N = 32$, $B_2/N = 0$ and $K = 223$, the buffer reduction ratio is 10.3%, as compared with the complete partitioning scheme shown in Figure 5.29.

### 5.3.4. The Capabilities of Multicasting, Modularity and Priority

The capabilities of multicasting, modularity and priority are essential for the future BISDN. The proposed switch is capable of providing these functions to meet the needs of a wide range of communications applications.

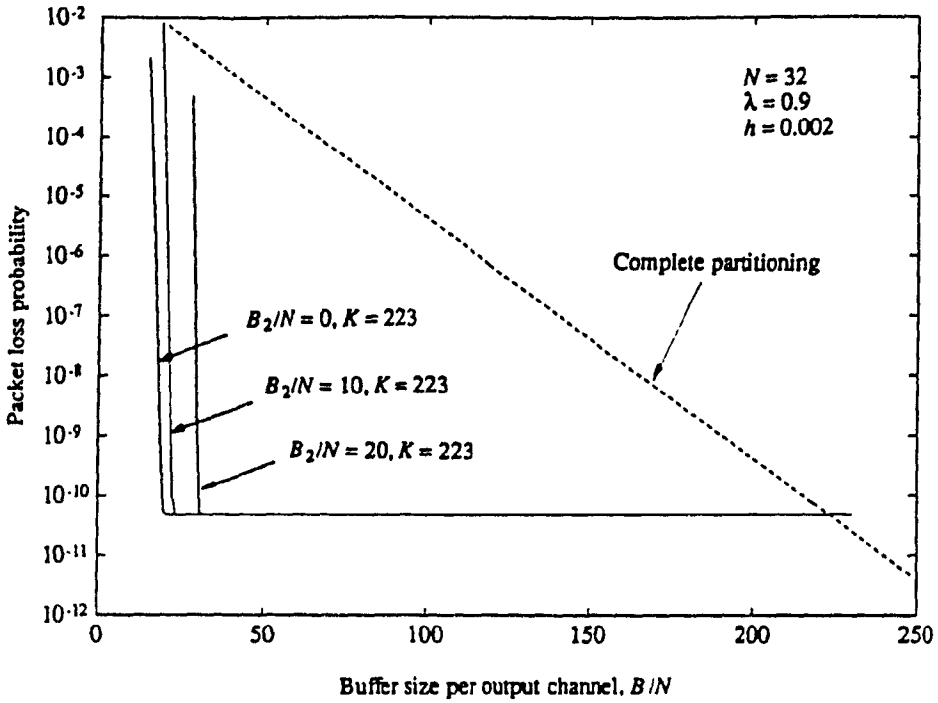



Figure 5.29. Packet loss probability vs. buffer size for sharing with maximum queue and minimum allocation under unbalanced traffic

The multicast function can be implemented with the addition of a circuit, as shown in Figure 5.30, to a shared buffer memory switch. The incoming multicast packet is detected by DEC1 and DEC2. An idle address in CIAB is picked out and placed into AREG, therefore, the multicast packet is stored somewhere in the shared memory pool rather than the reserved memory pool. The multicast operation is similar to the broadcast control in address controller of ATOM switch [22]. In DEC1, the multicast control signal (MCS) in bit-map form is worked out by a table look-up method with respect to multicast channel number brought by the multicast packet. The MCS activates the correspondent gates, and the address in which the multicast packet is stored is transferred into a set of address queues in FIFO QR with respect to the multicast destinations, as long as these address queues are unsaturated. Because each address queue is operating independently, a multicast packet may be transmitted to all its destination channels over several time slots. Therefore, we can not transfer the address into IMAP until all its copies are delivered. For

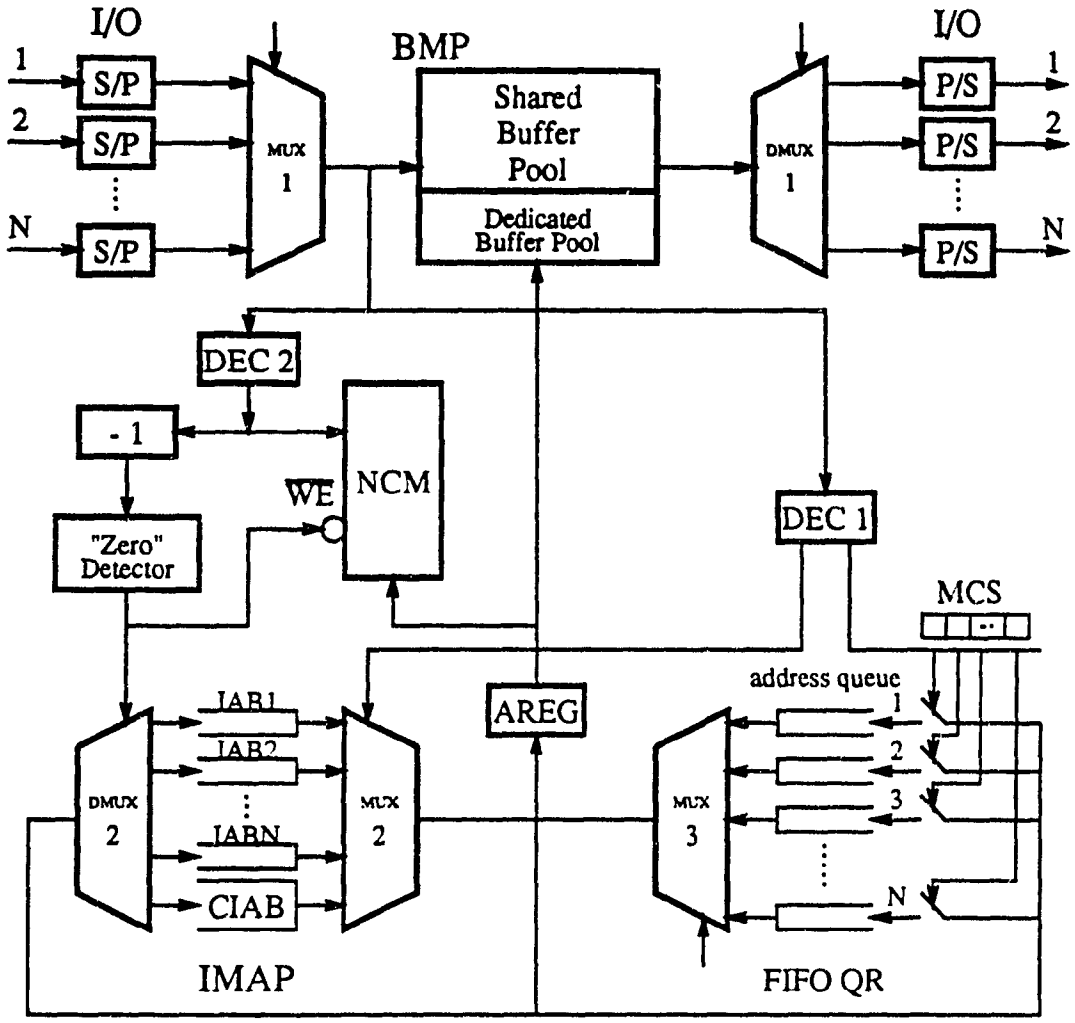



Figure 5.30. Multicast operation in proposed switch

this reason, we use an extended memory (NCM) which consists of $B$ cells and shares the address bus with BMP. When a multicast packet is written into BMP, the number of the copies of the packet decoded by DEC2 is written into NCM at the same address. When a copy of the packet is sent out, the content of NCM at the same address as the one that the packet is stored is read out and decreased by one. If the result after decrement is zero, which means that the packet has sent all its copies out, the address of the packet can be sent back to the IMAP; otherwise the result will be sent back to NCM. The additional circuit for multicasting is suitable for unicasting case

It is clear, as far as multicast is concerned, the proposed switch has advantage over Kuwahara's shared buffer memory switch. It features as simple and powerful in multicasting as ATOM switch does.

Similar to [7], this architecture can be easily modified to handle priority service by providing an extra set of address queues in FIFO QR, to correspond with each combination of service class and output port, respectively. Because of the buffer sharing effect among the output ports and service classes, no increase of the capacity of the BMP is needed.

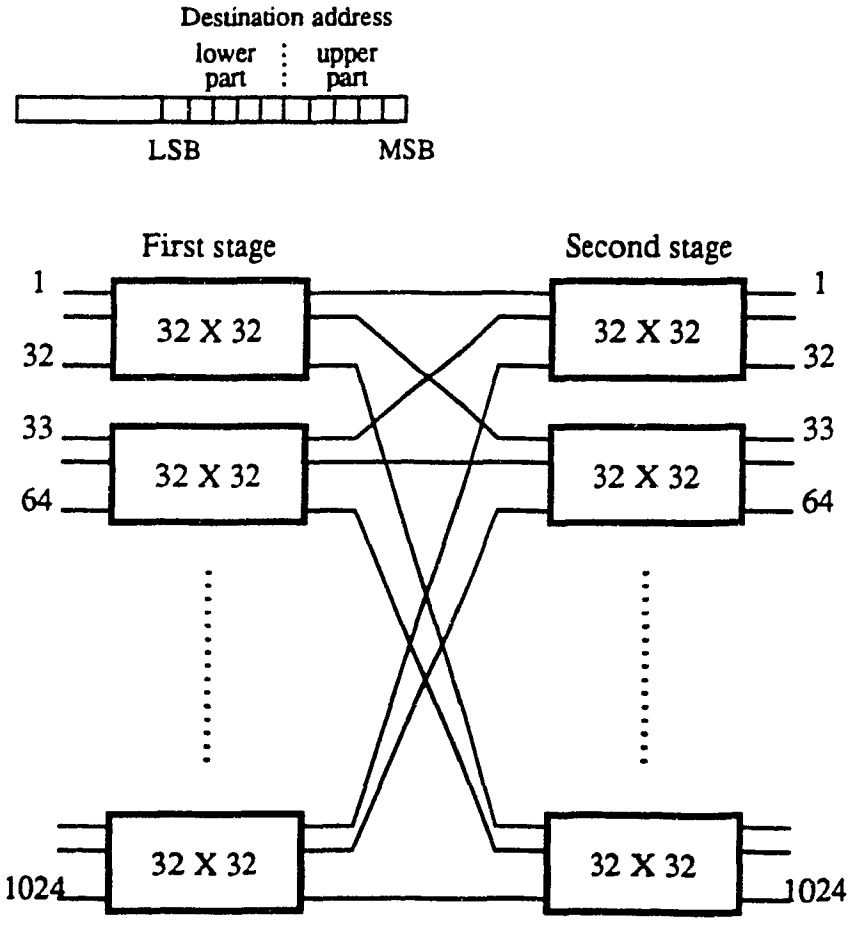



Figure 5.31. Modular structure of a 1024×1024 shared buffer memory switch

Because of its prominent characteristics of strict non-blocking and shared output buffering, the shared buffer memory switch is very suitable for the unit switch of large-size modular structure. Figure 5.31 shows how a 1024×1024 switch is built up with 64 32×32 switch modules, which are arranged in two cascade stages. Each packet carries a 10-bit destination address, which can be divided into two ports, upper part and lower part, to be employed as routing information in the first stage and second stage, respectively.

This two-stage switch structure can also provide multicasting function by using multicast switch model. Likewise, the multicasting procedure is divided into two steps. The upper part of the address of the copy is taken as the routing information in the multicast switch model at the first stage. The multicast packet may send one copy to an output link attached to a second stage switch model, if there is at least one copy that wants to go to the link (no matter how many copies want to go as long as these copies have the same upper part but different lower part of the destination addresses). The copy of the multicast packet coming from the first stage to the second stage is still a multicast packet, called as a local multicast packet. The destination addresses of the local multicast packet can be obtained by a simple table look-up, with attributes multicast channel number and the local address, which is exactly the sequence number of the switch model, as primary key of the table. The local multicast packet only generate copies to a set of destination output ports, in the second stage switch model. Therefore, we need different local multicast routing table for each of the second stage modules.

## 5.4. CONCLUSION

This chapter surveyed various architectures that have been proposed for multicast packet switches. It is hard to single out any one among the architectures as the best; each has its advantages and disadvantages. The real test will come with detailed implementation. The trade-off is not only based on the complexity of the switch structure, but also on the performance, the multicast standard and the increasing service requirements.

The design of a shared buffer memory switch proposed in this chapter provides the flexibility to meet various requirements, especially the prevention of the monopolization of the use of

shared buffers and secure a full utilization of all the $N$ output channels under unbalanced or bursty traffic. We also save on the total amount of buffer memory needed to achieve a desired packet loss probability. The functions of multicasting, modularity and priority are implemented quite easily. The future effort should be put on the implementation of the switch fabric.

## REFERENCES 5

[1] G. E. Daddis, Jr. and H. C. Torng, "A taxonomy of broadband integrated switching architectures", IEEE Communications Magazine, pp.32-42, May 1989.

[2] H. Ahmadi, W. E. Denzel, "A survey of modern high-performance switching techniques", IEEE J. Select. Areas Commun., Vol.7, No.7, pp.1091-1103, Sept. 1989.

[3] M. Listanti, A. Roveri, "Switching structures for ATM", Computer Communications, Vol.12, No.6, pp.349-358, Dec. 1989.

[4] F. A. Tobagi, "Fast packet switch architecture for broadband Integrated Services Digital Networks", Proceedings of the IEEE, Vol.78, No.1, pp.133-167, Jan. 1990.

[5] T. Takeuchi, H. Suzuki, T. Aramaki, "Switch architectures and technologies for asynchronous transfer mode", IEICE Trans., Vol.E 74, No.4, pp.752-760, April 1991.

[6] F. A. Tobagi, "Fast packet switching, ATM standards, and photonic networks", Tutorial at IEEE Infocom'91, Miami, April 1991.

[7] H. Kuwahara, et al., "A shared buffer memory switch for an ATM exchange", Proc. ICC'89, pp.118-122, Boston, June 1989.

[8] A. Huang, S. Knauer, "Starlite: a wideband digital switch", Proc. of Globecom 84, pp.121-125, Atlanta, GA., Dec. 1984.

[9] J. S. Turner, "Design of a broadcast packet switching network", IEEE Infocom'86, pp.667-675, 1986. (also see IEEE Trans. on Commun., Vol.36, No.6, pp.734-743, June 1988).

[10] T. T. Lee, R. Boorstyn, E. Arthurs, "The architecture of a multicast broadband packet switch", Proc. IEEE Infocom'88, pp.1-8, New Orleans, LA, March 1988.

[11] K. E. Batcher, "Sorting networks and their applications", AFIPS Proc. Spring Joint Comput. Conf., pp.307-314, 1968.

[12] C. -L. Tarng, J. S. Meditch, "A high performance copy network for B-ISDN", IEEE Infocom'91, pp.171-180, Miami, April 1991.

[13] T. T. Lee, "Nonblocking copy networks for multicast packet switching", IEEE J. Select. Areas Commun., Vol.6, No.9, pp.1455-1467, Dec. 1988.

[14] T. T. Lee, "A modular architecture for very large packet switches", IEEE Trans. Comm., Vol.38, No.7, pp.1097-1106, July 1990.

[15] Y. S. Yeh, M. G. Hluchyj, A. S. Acampora, "The knockout switch: a simple, modular architecture for high-performance packet switching", IEEE J. Select. Areas Commun., Vol.5, No.8, pp.1274-1283, Oct. 1987.

[16] K. Y. Eng, M. G. Hluchyj, Y. S. Yeh, "Multicast and broadcast services in a knockout packet switch", Proc. IEEE Infocom'88, pp.29-34, New Orleans, LA., March 1988.

[17] H. S. Kim, "ATM packet switch for broadband ISDN", Ph.D Dissertation, University of Toronto, 1990.

[18] H. Imagawa, et al., "A new self-routing switch driven with input-to-output address difference", Proc. of Globecom'88, pp.1607-1611, Hollywood, FL., Nov. 1988.

[19] T. Kozaki, et al., "32×32 shared buffer type ATM switch VLSI's for B-ISDN's", IEEE J. Select. Areas Commun., Vol.9, No.8, pp.1239-1247, Oct. 1991.

[20] Y. Shobatake, et al., "A one-chip scalable 8×8 ATM switch LSI employing shared buffer architecture", IEEE J. Select. Areas Commun., Vol.9, No.8, pp.1248-1254, Oct. 1991.

[21] F. Kamoun, L. Kleinrock, "Analysis of shared finite storage in a computer network node environment under general traffic conditions", IEEE Trans. on Comm., Vol.28, No.7, pp.992-1003, July 1980.

[22] H. Suzuki, H. Nagano, T Suzuki, T Takeuchi, S Iwasaki, "Output-buffer switch architecture for asynchronous transfer mode", Proc. ICC'89, pp.99-103, Boston, MA., June 1989.

[23] J. Fried, "A VLSI chip set for burst and fast ATM switching", Proc. ICC'89, pp.128-135, Boston, MA., June 1989.

[24] H. J. Chao, "A novel architecture for queue management in the ATM network", IEEE J. Select. Areas Commun., Vol.9, No.7, pp.1110-1118, Sept. 1991.

[25] W. W. Chu, "Buffer behavior for batch Poisson arrivals and single constant output", IEEE Trans. on Comm. Tech., Vol.18, No.5, pp.613-618, Oct. 1970.

[26] J. F. Hayes, *Modeling and Analysis of Computer Communications Networks*, Plenum Press, 1984.

[27] A. E. Eckberg, T. -C. Hou, "Effects of output buffer sharing on buffer requirements in an ATDM packet switch", Proc. Infocom'88, pp.459-466, New Orleans, LA., March 1988.

[28] L. Kleinrock, *Queueing Systems, Vol.1: Theory*, John Wiley & Sons Inc., 1975.

[29] H. Yoon, M. T. Liu, K. Y. Lee, "The knockout switch under nonuniform traffic", Proc. Globecom'88, pp.1628-1634, Hollywood, FL., Nov. 1988.

[30] G. F. Pfister, V. A. Norton, "Hot-spot contention and combining in multistage interconnection networks", IEEE Trans. on Computers, Vol.34, No.10, pp.943-948, Oct. 1985.

# CHAPTER 6

# CONCLUSIONS AND FUTURE RESEARCH

6.1 CONCLUSIONS

6.2 FUTURE RESEARCH

## 6.1. CONCLUSIONS

The multicast not only brings promising services, but also raises new research topics which do not arise in unicast switches. In this dissertation, we discussed various aspects of multicast packet switching with emphases on the issues of call scheduling disciplines, contention resolution algorithms, mathematical analysis and switch architecture.

In order to regulate the input access traffic for a input port queueing switch, a novel scheme — revision scheduling was proposed, the purpose of which is to maximize the number of packets getting through and the number of output lines being busy simultaneously. By comparison, the revision scheduling is the most promising approach as far as the delay-throughput performance is concerned, and would give an indication of the best performance under the assumption of input port queueing.

A related issue of call scheduling is the implementation aspect of the scheduling disciplines, or the contention resolution algorithm. We proposed two novel algorithms, the cyclic priority access scheme with its combinational logic implementation and the neural network based algorithm, both with lattice structure. The combinational logic scheme features simple and high speed operation, in comparison with the other methods such as the ring-reservation algorithm and the three-phase algorithm. It is compact, reliable and growable. The neural network based algorithm provides better performance, higher throughput and lower delay. Furthermore, it demonstrates a potential of optimal scheduling.

We presented some analytic tools of traffic theory for the input port queue of the multicast switching system. Computer simulation was carried out to verify the mathematical models, in turn mathematical analysis may provide more statistical measures which are hardly observed through simulations. We discussed the mathematical analysis of both the random selection policy and the cyclic priority input access scheme for one-shot discipline. Then, we introduced a general unified mathematical model for both one-shot and WS call splitting input access disciplines, by using matrix-geometric techniques in an effort to trace the effect of HOL blocking. These results could serve to model the multicast packet switch and to predict the onset of

congestion of the system. Although the discussions are devoted to the multicast switch, the analytic models can also be applied to other queueing problems.

We summarized some multicast packet switches proposed in literature, including banyan-based space-division switch, knockout switch, shift switch and shared buffer memory switch. Then we proposed a shared buffer memory switch structure with maximum queue and minimum allocation, where a shared buffer pool and a reserved buffer pool are properly handled for switching and buffering the packets. The size of buffer memory, required for certain packet loss probability under both balanced and unbalanced traffic pattern, was estimated by giving an upper bound. The proposed switch possesses multicasting, modularity and priority functions to meet the needs of a wide range of communications applications.

## 6.2 FUTURE RESEARCH

Many issues discussed in last four chapters are still needed to be investigated and studied extensively. For example, we need to justify the switch structure to implement multicast switching efficiently, not only based on call scheduling disciplines and contention resolution algorithms, but also in consideration of economical aspects and adaptation to practical networks. As to mathematical analysis, a more accurate and closed form solution is sought to model the multicast packet switch with unicast and broadcast switching as the special cases. Another open question is the performance evaluation under the correlated input traffic or the stream-like traffic, such as voice, video and file transfer.

Many related issues which we did not mention in the dissertation are also of considerable interest and will be our future research projects, such as multicasting standard, addressing of the multicast packet, multicast routing and so on.

Also, many new issues of multicasting packet switching will come out with the deployment of BISDN. We will never see the end of the study.