PERFORMANCE EVALUATION OF HYBRID BUFFER ATM SWITCH

by

YUE HE

B.Eng., Lanzhou Railway Inst., PRC

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

DEPARTMENT OF ELECTRICAL ENGINEERING

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

December, 1996

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Electrical Engineering

The University of British Columbia
Vancouver, Canada

Date Dec. 24, 1996

Abstract

Buffer management and cell scheduling are the most important factors affecting the design of packet switching architectures for ATM networks. Buffering is the major resource that dominates both the cost and performance of ATM switch fabrics. Buffer management is required whenever the instantaneous cell arrival rate at a switch output is higher than the output link rate. This thesis presents a hybrid buffer ATM switch architecture in which the buffer management scheme is realized by dedicated output buffers and a shared buffer. A new approach based on queue tail management, as well as distributed priority scheduling is incorporated in the proposed switch design. The proposed scheme aims at enhancing the performance of ATM switches by maintaining the head cells of logical output queues in relatively short dedicated output buffers, while maintaining the long tails of overflowing queues in a shared pool managed by a more intelligent buffer management unit. Under this scheme higher priority (or delay-sensitive) cells can be forwarded immediately to the output buffers potentially blocking some lower priority (but loss-sensitive) cells from advancing into the output buffers. If the shared buffer is full, then a suitable push-out scheme is employed. The output buffers employ a distributed priority scheduling technique for sending cells to output links. The proposed scheme is capable of handling multi-priority traffic and of satisfying the QoS requirements of each class using a very simple architectural scheme with simplified distributed control policies compared to other schemes previously proposed.

The hybrid buffer switch performance is evaluated using discrete time queueing theory and using simulation. We provide detailed analysis of the buffer management scheme by

decomposing the complex partial sharing buffer analysis into an equivalent queueing problem. The thesis provides semi-closed form expressions for various performance measures. The discrete time queueing model is verified by simulation. The computation complexity of the queueing model is much smaller than that of similar models in the literature.

The switch performance under multi-class traffic is critical to the network operation because real-time and non-real-time traffic demands different "quality of service" (QOS) regarding delay jitter and cell loss rate. For efficient buffer management, a simple two-class priority scheme is implemented in the output buffers to minimize cell delay and delay jitter for the delay-sensitive (or real-time) traffic. A simulation package is developed and applied to evaluate the hybrid buffering switch under two-class (loss-sensitive and delay-sensitive) traffic using two different hybrid scheduling and buffer management polices. The extensive simulation results indicate that the QoS requirements for both traffic classes can be easily met by appropriately dimensioning the hybrid priority scheduling and buffer management policies. The hybrid distributed buffer control provides for flexible scheduling and keeps the hardware implementation cost lower than other switch architectures.

Table of Contents

List of Figures

Acknowledgement

I would like to thank my thesis supervisors, Dr. Hussein M. Alnuweiri and Dr. Mabo R. Ito for defining the research topic, their continuous guidance and encouragement which contributes to the successful completion of the research work of this thesis.

I also would like to thank my former supervisor Dr. A.Antoniou, Department of Electrical and Computer Engineering, University of Victoria, for his support during my course study period in University of Victoria.

Finally, I would like to thank my parents, my sister, my brother, and my friend Irene, for their love and support to secure my completion of this work.

Publication

Yue He, H. M. Alnuweiri, M. R. Ito, "Enhancing ATM Switch Design with Shared Queue Tail Memory ", *HiNet'96 Workshop, International Parallel Processing Symp'96*, Hawaii, USA, April, 1996

Chapter 1 Introduction

The broadband integrated services digital network (B-ISDN) is expected to support a wide range of current and future generation of communication services. The network will carry a wide range of multimedia traffic with varying characteristics and quality-of-service requirements, such as voice, video, and data. A fixed-size packet switching technique, called asynchronous transfer Mode (ATM), has been selected as the ITU (formerly known as CCITT) standard for B-ISDN because it has the flexibility to handle multimedia communication services with varying traffic characteristics and quality of service requirements[1][2]. The fixed size packets are called cells in the ATM terminology.

ATM is a connection-oriented technology in which the cells travel from point to point using concept of the virtual circuits (VCs) and virtual paths (VPs). Packet (or ATM) switches are thus required to switch cells among VCs or VPs at various points in the network. To support integrated services, a fast cell switch in the ATM network must be capable of handling the requirements of different services. Certain types of services have stringent cell delay requirements, such as voice and real-time video. Other services, such as data and file transfer, can tolerate a certain amount of delay but have strict cell loss rate requirements. The input traffic in the ATM network can be smooth or bursty, with balanced or unbalanced load with respect to output destinations uniformly distributed or non-uniformly distributed [15].

The ATM switch is the crucial component of any ATM network. The switch must be capable of meeting the requirements of the high speed input links and the quality of service (QoS)

requirements. It must employ some type of cell buffering to store the cells whose service must be delayed due to contention for the same output address. There are three major buffer allocation policies: input queueing, output queueing and shared-queueing.

This introductory chapter provides the motivations for the research subject of this research. The objectives of this thesis are stated and the topics which will be covered in the subsequent chapters are outlined at the end of this chapter.

1.1 Motivation

A basic switch performs two major functions: cell switching (or routing) from input links to output links and cell buffering. The placement and dimensioning of buffers have a dramatic impact on ATM switch performance. A number of queueing disciplines have been proposed for ATM switching, such as input queueing, output queueing and shared-queueing.

In a nonblocking switch structure, cells at different input ports can be simultaneously forwarded to the requested output ports. Assuming only non-blocking structures, if dedicated buffers are placed at the input ports and are operated in First-In-First-Out (FIFO) order, the implementation is simple. But when a cell at the head-of-the-line (HOL) position of a FIFO queue waits for transmission to the destined output port, subsequent cells in the queue must also wait. Consequently, input buffered switches can achieve a maximum throughput of 58% which is considered to be very poor [4].

In a non-blocking switch with buffers dedicated for each output port, the switch does not

suffer from HOL blocking and thus there is no delay/throughput degradation. Output buffered switches can achieve the highest throughput for general input traffic as well as fair buffering. The drawback of output queueing is that it requires the switch to operate N times faster than the input/output link speed (where N is the switch size, i.e., the number of input/output ports), which increases the implementation complexity. When more functions are to be implemented, a fast control circuit is needed for each buffer. The buffer space is not efficiently used [3]. In handling multi-priority traffic, a switch requires the use of large, complex, priority or sorting queues. Implementation costs are high because of increased hardware complexity. Only high priority traffic benefits from a priority queue. Low priority traffic only contributes to increasing buffer size which may unjustifiably increase cost.

A shared-buffering switch has the same throughput/delay performance as output queueing but requires much less buffer to achieve a given cell loss [2]-[4]. In a shared-buffer switch, all input and output ports have access to a shared-buffer module capable of writing up to N incoming cells and of reading out N outgoing cells in a single cell slot. The switch can be easily modified to handle several service classes through priority control functions and multicast/broadcast functions[24]. Although a shared-buffering switch performs well under uniform or bursty traffic, it does not yield satisfactory results under non-uniform traffic because some queues may overtake most of the buffer space leaving little space to other queues. In particular, under worst case hot-spot and burst traffic conditions, the hot-spot queue can occupy all the buffer space thus preventing other queues from accessing any buffer location [5][6].

A compromise solution would combine the advantages of shared-buffer and output buffer structures by employing a hybrid buffering scheme of shared-and dedicated output buffers. The

dedicated buffers can be of the simple FIFO type or they can implement a priority queueing discipline. We will elaborate further on this proposal in the next chapter.

Early work on the analysis of shared-buffer switches assumed exponential arrival and service time distributions and product form queueing models[7]-[11]. Such assumptions are appropriate for data networks but are not suitable for ATM based networks, in which the network carries several classes of traffic and the cells are of fixed size[5]. Analysis more relevant to ATM systems can be found in [12] - [14], all of which examine the complete buffer sharing queueing discipline. As far as the author knows, there is no a known model for our proposed hybrid buffering switch.

1.2 Thesis Scope and Objectives

The buffering method proposed in this thesis combines output-buffered with shared-buffer switching. This hybrid buffer switching technique aims at combining the advantages of the enhanced cell loss of shared-queuing with the fairness and distributed priority control of dedicated output queueing.

This thesis proposes an efficient method for enhancing the performance of shared-buffer ATM switches in handling multiclass traffic. The focus of the proposed method is on handling two broad classes of cell traffic: delay-sensitive (or real-time) traffic and loss-sensitive (or data) traffic. Examples of delay-sensitive traffic include constant-bit-rate (CBR) traffic as well as real-time variable bit-rate (VBR) traffic. The main example of loss-sensitive traffic is what is currently

known as available-bit-rate (ABR) cell traffic. We propose a two-space priority scheme with two classes of traffic. Basic performance evaluation of the proposed switch under single class is carried by applying analytical queueing techniques. However, modeling and performance evaluation of the two-space priority schemes in the proposed ATM switch is obtained mainly by simulation because the traffic model and the queueing model are rather too complex to be handled by analytical techniques. Additionally, simulation provides more accurate results.

A challenging issue in the analysis of the hybrid switch is how to develop an accurate queueing model to evaluate its performance. The main difficulty in such analysis is deriving an accurate analytical model for the queue length distribution and the joint distribution of the queue-tail length distribution in the shared-buffer. Therefore, a major objective of this thesis is to derive an accurate and useful queueing model for evaluating the performance of the proposed hybrid ATM switch. In particular, an analytical model is developed for the joint distribution of output queue tails in the shared-buffer. The validity of a model has been verified by extensive simulation.

We propose two space priority schemes with two classes of traffic. Modeling and performance evaluation of the two space priority schemes in the ATM switch is obtained by simulation because the traffic model, switch architecture and routing policy are complicated and hardware emulation is too expensive to perform.

1.3 Thesis Outline

Chapter 2 discusses some problems with existing ATM switch architectures and particu-

larly the advantages and disadvantages of shared-buffer switches. It also discusses the technologies and architectures employed in existing shared-buffer switches. Chapter 3 defines the architecture, performance measures, operations, and features of the hybrid buffering switch. Chapter 4 develops an analytical queueing model for the performance evaluation of the hybrid buffering switch architecture under independent traffic with FIFO scheduling policy. Chapter 5 first defines the simulation parameters for the design in terms of performance measures, traffic models, and switch models with different scheduling policies. The second part of Chapter 5 presents simulation results for cell loss, cell delay, and related quality of service parameters. Chapter 6 summarizes the results, provides conclusions on the research and presents other possible areas for future research.

Chapter 2 ATM Switch Architecture

In this chapter, we describe the functions, operations, architectures, performance and problems in the ATM switch. Particularly, three major architectures are discussed with respect to their performance and problems.

The ATM network will carry four service classes to support connection-oriented as well as connectionless services. These services QoS range from delay-sensitive circuit emulation and variable bit-rate video to low-rate but loss-sensitive data services. ATM is a multi-rate, multi-service, flexible cell switching technology that can provide quality of service guarantees to multiple services with varying requirements. However, the excessive use of the bandwidth may cause QoS deterioration, such as higher cell loss and cell delay [1]-[4].

In an ATM switch, the input links transmit cells into switch input ports at a high rate, e.g. 155 Mbits/s, and the switch routes the input cells to their output links. Each input link sends one cell and each output link only accepts one cell during one time unit. However some cells arriving simultaneously at the input ports may be destined to the same output port but the output port can only receive one of them at a time. The rest of the cells must be queued in buffers to be resent in subsequent time units to the output link. As shown in Figure 2.1, there are three cells destined to output port 1 but only one of them can be routed to output link 1. A buffer is needed to save the rest two cells destined to output port 1 inside the switch. This is called "Output contention" problem in an ATM switch. There are three major buffer allocation policies: input buffer, output buffer and buffer sharing.

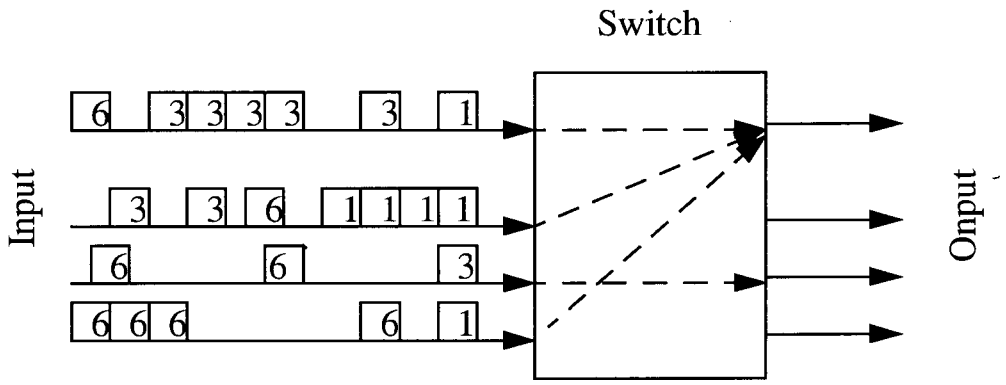

Figure 2.1 Output port contention.

2.1 Input-Buffered Switch

When the buffers are placed at the input ports of the switch as in Figure 2.1, it is called an input buffer switch. The cells enter the buffers from the input links and the cells at the buffer heads are routed to output ports by the switch. The switch operates as fast as the input/output links. The output contention problem is solved by the input buffer in the manner that if there are more than one cell at the buffers heads with the same output address, only one cell is switched to an output port; the rest of the cells to this port are stored in waiting for routing in the next time unit. However, the cells, which fail in output contention and stay at the buffer heads, block the cells behind them in the buffer which may be addressed for the idle outputs. As in Figure 2.2, the cell addressed to output 2 in input buffer 1 will not contend with the other cells because it is

destined to an idle output port. But it may be blocked by the HOL cell in front of it. This type of blocking is called HOL blocking, and it degrades the switch throughput to only 58%.

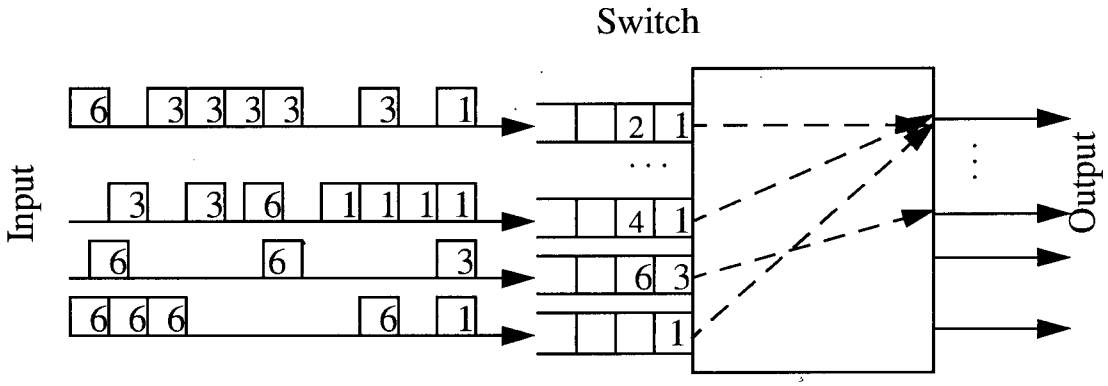

Figure 2.2 Input queueing switch.

2.2 Output Buffered Switch

By speeding up the switch internal operation of the switch and adding buffers to the output ports, the throughput of an input buffered switch can be improved. If the speed up factor is increased to match the number N of input ports or the switch size, when all HOL cells of input queues can be sent to the output ports without blocking, the N input buffers are no longer necessary in this case but large buffers are needed at the output ports since each output link can only transmit one cell during one time unit. Output buffered switches achieve the highest through-

put under general input traffic.

Because buffers are dedicated to each output port, fairness is also achieved by this scheme because different flows (to different destinations) are isolated from each other in the output buffer. One problem with output buffering is that large capacity (memoryless) interconnection fabrics are required to pass cells to the output buffer with minimal blocking. For large size switches, the implementation complexity of such switch scheme increases rapidly. Multicast and priority routing implementation is somewhat costly in such a switch. When more traffic management functions are to be implemented, a fast control circuit is needed for each buffer [2]-[4].

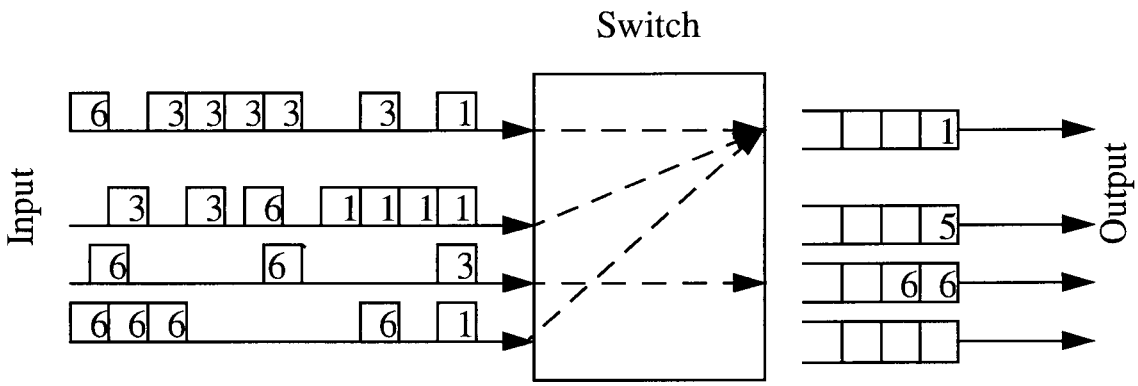

Figure 2.3 Output queueing switch.

2.3 Shared-Buffer Switch

The shared-buffer switch is considered to be one of the effective architectures for small and medium scale ATM switches. A shared-buffer switch has the same throughput/delay/cell-loss performance as an equivalent output-buffered switch, while requiring much less memory[24]. Buffer utilization is high because the buffer can be shared-by all ports. Multicasting functions, priority routing, are easy to implement. In the shared-buffer switch, all input ports and output ports have access to a fast shared-buffer able to write up to N incoming cells and to read out N outgoing cells in a single switching time cycle, throughput is not reduced by output port contention. Furthermore, to provide a specified level of service a smaller number of buffers is required for all traffic patterns. A shared-buffer switch also has some additional features, e.g., its basic architecture can be easily modified to handle several QoS classes through priority control and buffer management functions to meet different service requirements, multicasting and broadcasting can be also easily implemented. In the following, we provide a brief survey of shared-buffer architectures.

2.3.1 Time-Division Shared-Buffer Switch

One of the first shared-buffer switch architecture using the ATM-like concept was the Prelude switch described in [2]. It is a modified conventional time-division non-blocking switch. In this approach, the switch size N must be equal to the number of bytes in a cell. The switch can be divided into parts performing the following functions: serial/parallel conversion, multiplexing, control, buffering, demultiplexing and parallel/serial conversion. Serial-to-parallel conversion of

incoming cells reduces the demand for physical memory speed. Incoming cells join clock adaptation queues from where they will be extracted and aligned in such a manner that they present a shift of one byte from each other, so that headers on different lines arrive in consecutive time slots.

The multiplexing function is performed by a space division switch that sets its internal paths every time cycle according to a cyclic algorithm. As a result, all the headers of different input cells are multiplexed on the first output line of the multiplexer, all the first user information bytes on the second output, etc. The header bytes are successively processed and translated by a central controller to define the switch action. The cells are stored diagonally in the buffer which is organized in N banks, each bank dedicated to one byte of the cell, i.e., the header is stored in an empty location of bank 1, and the remaining bytes of the same cell are stored in consecutive banks. The address where the cell header is placed is stored by the controller in a dedicated queue for the destined output port.

To route cells to their output links, the controller, in each time cycle, delivers the address of a cell header to the first memory bank and the the remaining bytes of the cell are easily retrieved from the other banks during the following time cycles increment. Retrieved cells are fed to a rotative space-division switch which acts in the reverse order as in the multiplexing stage, thus reconstructing the cells. Finally, the cell are converted from parallel to serial form and are transmitted to the corresponding output lines.

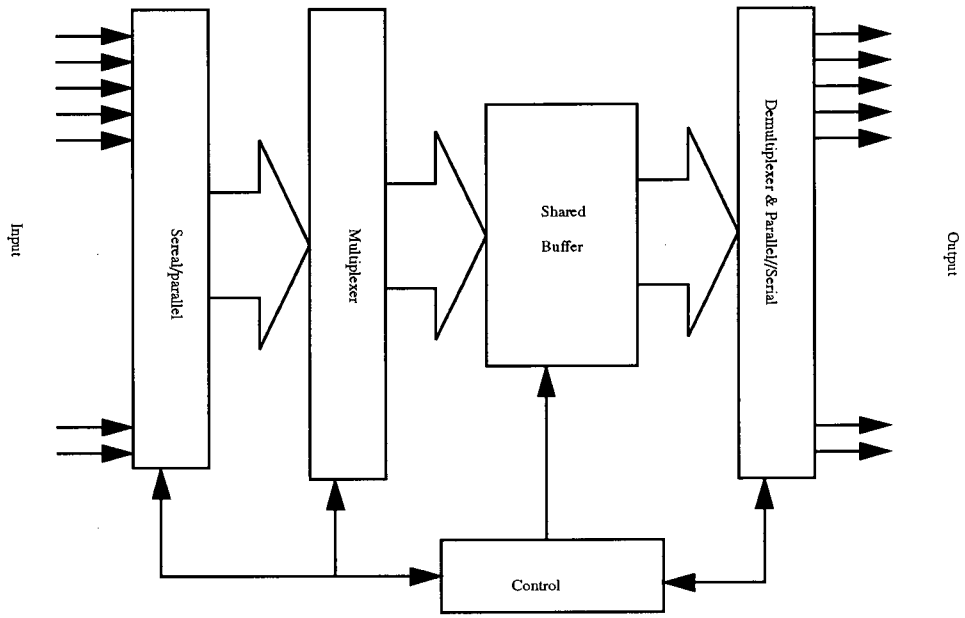

Figure 2.4 Time-division Prelude Shared-Buffer Switch.

2.3.2 Linked-List Memory Switch

Another type of shared-buffer switch employs linked-lists to logically organize the shared-buffer space [2]. The basic idea is the use pointers to group the cells of each output queues in a single linked list. Thus, the shared-buffer for $N \times N$ switch will contain N linked lists, one for each output port. Note that implementing pointers requires the use of an additional pointer memory as well as an idle-address FIFO to keep track of available buffer address in the shared-buffer.

Basic operation is as follows. Cells from all inputs are multiplexed and written into the shared-buffer. The buffer is logically organized as linked-lists, one for each output port. A linked list is a set of chained memory locations that indicate the place where successive cells for a particular output are stored, maintaining, in that way, the cell sequence. A pair of address registers control the proper operations in the shared-buffer. The basic design can be easily modified to accommodate different service classes by organizing the cells destined to a given output port into multiple linked lists, and applying the proper functions to each services.

The multicast function requires the addition of a multicast circuit. Multicast and broadcast cells form a broadcast queue in the shared-buffer, irrespective of their destinations. This queue has the same structure as all other output queues. Before a multicast cell is sent out from the switch, its header is analyzed to define the output port numbers to which the cell is to be multicast and this information is stored in a multicast routing table. The routing information is used to deliver multiple copies of the same cell to the specified output ports.

2.3.3 Shared and FIFO Address Buffer Switch

In this approach, the control mechanism route the cells in the shared-buffer to their output ports using N dedicated FIFO buffers, one for each output port as shown in Figure 2.5. Thus, instead of arranging cells with same output port destination in a linked list by the address chain pointer, their addresses are stored in a FIFO buffer which is dedicated to a specific output port.

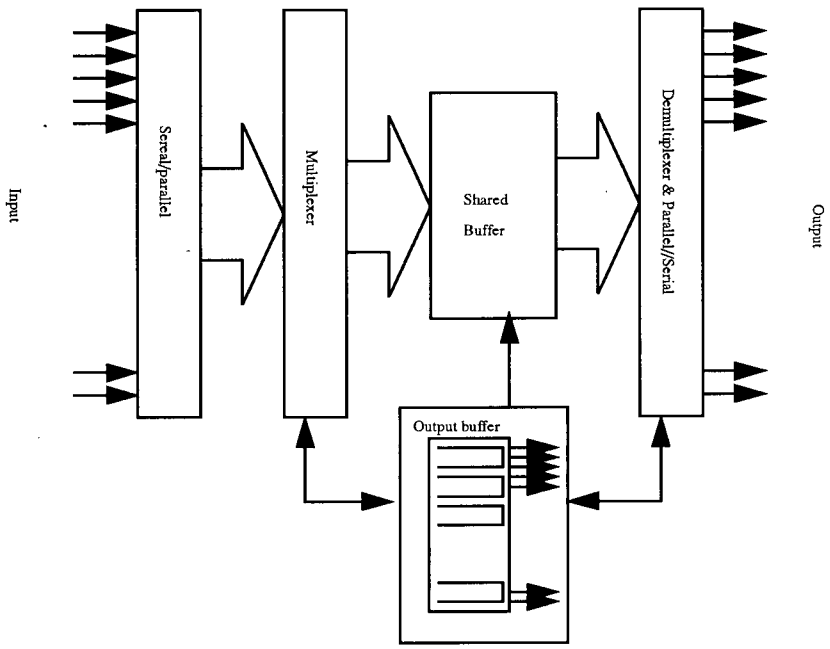

Figure 2.5 shared-and FIFO Address Buffer Switch.

Serially incoming cells are sequentially multiplexed in time and stored in a shared-buffer. Their respective buffer addresses are recorded on separate FIFO buffers associated with the specific destination of every cell. A particular FIFO buffer is selected by examination and conversion of the routing header information of each cell. The Write addresses to store the incoming cells into the memory pool are provided by an idle address FIFO buffer that holds all the empty buffer locations of the shared-buffer. The addresses in the FIFO buffers are used to read the cell out from the shared-buffer, and to send them to their destinations. The new free addresses are returned to the idle address FIFO.

Several other operations of the switch, such as multicast support, is the same as that of the linked-list shared-buffer switch. Priority control can be easily achieved by allocating several

address FIFO queues priority class. During the the Read phase, the FIFO queues associated with a particular output port are scanned according to their priorities. Another important feature of the switch is that it is easy to shift from a complete sharing to a complete partitioning scheme of the shared-buffer space since the sizes of the address FIFO buffers determine the number of cells which can be stored into the shared-buffer with same output destination.

2.3.4 Performance

Buffer management and cell scheduling are the most important factors affecting the design of cell switching architectures for ATM networks. Buffering is the major resource that dominates both the cost and performance of ATM switch fabrics. Buffer management is required whenever the instantaneous cell arrival rate at the switch is higher than the output rate. Developing efficient buffer management policies has become even more important with the advent of ABR (available bit-rate) services that employ feedback congestion control to throttle sources during times of congestion. ABR connections are characterized by a minimum cell rate (MCR), but will accept any available bit-rate (higher than MCR) in an ATM network. In the case when the bandwidth available to a user decreases, the switches must provide sufficient buffering to avoid excessive cell loss while sources are converging on lower rates. Issues of dynamic bandwidth allocation, congestion control, fairness, are essential to the success of ABR services [37].

In bursty and non-uniform traffic environments, the performance of a shared-buffer switch

may not be as good as an nonblocking output-buffered switch [5]-[6]. There are three major buffer allocation schemes: dedicated buffering, complete-buffer sharing and partial-buffer sharing. In a dedicated buffering scheme, a buffer is dedicated to each out port. This is the most fair strategy, but it can suffer from sever buffer under utilizationl. In a complete-buffer sharing scheme, the entire memory is treated as a common pool to store the cells. A queue length can have any value as long as there is sufficient memory space. Unfairness can result because some queues may take most of the buffer space in bursty traffic.

A compromise solution to this problem by proposing a queuing structure that combines dedicated output queueing and complete buffer sharing, the two above schemes, have been discussed, such as the method proposed in [5]. Many buffer management schemes have been proposed to improve the performance of a shared-buffer switch [21]-[25]. These schemes are efficient in reducing cell loss and in enhancing fairness and buffer utilization for all ports. Partial-buffer sharing provides good performance under non-uniform traffic, but the buffer management requires more control functions to be implemented in the switch.

With the increasing line speed, switching functions, and switch size, the switching efficiency of a shared-buffer ATM switch becomes limited mainly by memory-access and the speed of the centralized buffer controler. Most previous research emphasized increasing the efficiency of centralized switch control or memory bandwidth [7]- [21]. In this thesis, distrib-uted switch control and scheduling are employed not only to increase the switch size, routing flexibility and speed but to reduce the control complexity.

2.3.5 Analysis

Early work in the analysis of shared-buffer switching can be found in [7]-[11]. They assume exponential interarrival and service time distributions and employs product form queueing models to obtain solutions for complete sharing of buffer space (CS), complete partitioning of buffer space (dedicated output buffer, CP), buffer sharing with minimum allocation (SMA) and buffer sharing with maximum queue length (SMXQ). The above traffic assumptions are not necessarily appropriate for ATM based networks, which carries a variety of traffic types and the cells are of fixed size [5]. Analyses more relevant to ATM switching system can be found in [12]-[14], all of which examine the complete buffer sharing (CS) model. The complexity of the interaction of logical queues in a shared-buffer switch presents an exact Markov model for the system. Thereofre, considerable research has been done on various approximations to reduce the complexity [12]-[14]. Turner in [12] presented a Markov model with a state space that keeps track of the overall buffer occupancy level and determines the individual logical queue levels by probabilistic means. The performance results tend to be overly optimistic compared to simulations. Monterosso and Pattavina [13] termed Turner's approach as the *scalar model* and introduce a more accurate though more computationally costly *vector model* with state space that maintains individual virtual queue occupancy information. Late, Bianchi and Turner [14] introduced three improved models: the *uniform scalar model* which still only keeps track of overall occupancy but has more intelligent way of properly determining individual queue length, the *bidimensional model* which keeps track of overall occupancy and the number of non-empty virtual queues, and the *interval model* which is a simplified bidimensional approach.

The above queueing models are not necessarily suitable for the hybrid buffering switch

proposed in this thesis. The difficulty is in deriving an accurate analytical model for the logical queue length distribution and the joint queue length distribution in the shared-buffer. Chapter 4 develops a queueing model for the hybrid buffer switch. The queueing analysis makes use of a discrete time queueing approach whoes computational complexity is independent of buffer size. The analysis deals only with the dedicated logical queues in the switch. Our approach is based on reducing the shared-buffer analysis problem to a simpler dedicated-buffer problem.

Chapter 3 Modeling the Hybrid Buffer Switch

In this chapter, we introduce the proposed hybrid buffer switch for improving the performance of the output buffer switches. The design principle, operation, and features are described. Particularly, the queue tail management, distributed buffer control and hybrid scheduling concepts are presented.

3.1 Queue Behavior

In an ATM switch, the input links transmit cells into switch input ports at a high rate, e.g. 155 or 622 Mbits/s, the switch routes the input cells to the output links according to their cells' output addresses: each input link sends one cell and each output link only accepts one cell during one time unit. However cells from different input links may have the same output destination. In this case, cells must be queued in this output port to be transmitted in subsequent time units.

Under non-uniform or bursty traffic, some output ports and their associated queues are busy while others are idle. Therefore, the instantaneous queue lengths for the different output vary considerably as shown in Figure 3.1. The cells in the long queue tails will experience more delay and are prone to be dropped, and this dominates the QoS of cell loss and delay jitter in the switch. Management of the cells in the switch buffers is a critical issue.

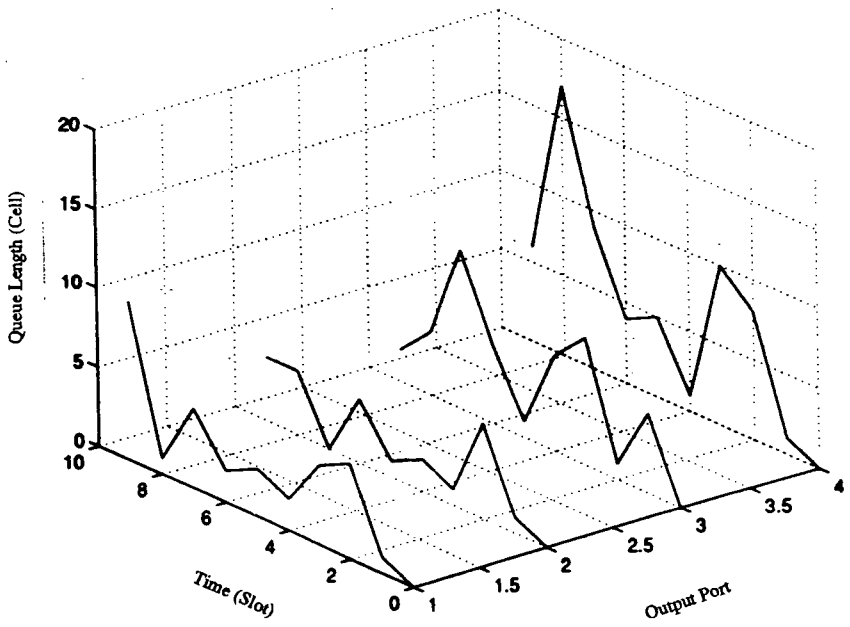

Figure 3.1 Queue behavior

3.2 Hybrid Buffering Switch Architecture

This section proposes an ATM switch model that employs two main levels of buffering: a shared-buffering level, and a dedicated output buffering level. The model provides a comprise solution between pure output buffering and completely shared-buffering. It has the advantage of providing a simple, yet effective, solution to the problem of fairness in shared-buffer architectures, while retaining their low cell loss performance. However, the major advantage of the two-level buffering structure is its ability to handle multi-priority traffic quite efficiently. In particular, it will be shown in Chapter 5, that the proposed buffering structure can satisfy the requirements of two-priority input traffic classes (one loss sensitive and one delay sensitive)

The basic principle of the proposed switch model is to provide incoming cells with two paths to output ports. One path sends incoming cells directly to output buffers if their logical queues in the shared-buffer are empty. The second path queues an incoming cell at the tail of its logical queue in the shared-buffer. Thus, the front portions of the logical output queues are always located in the dedicated output buffers, while their tails (if they exist) share the address space of the shared-buffer. In fact, the queue addressed to one output port is divided into two parts, one in the output buffer and one in the shared-buffer as a queue tail. We call this queue a "logical queue" as it is formed by two tandem queues.

A possible architecture for the proposed switch model is shown in Figure 3.2. It consists of a set of dedicated output buffer (DOB) modules and a shared-buffer (SB). The logical queue assigned to an output port consists of two parts: a queue front (or head) which contains the cells in the DOB and a queue tail which contains those cells of the logical queue that reside in the SB. Note that the queue tail part exists only when the logical queue size is greater than the DOB size. When a cell enters the switch, if the output buffer to which it is destined is not full, i.e., its logical queue length is smaller than DOB size, the cell is passed directly to the output buffer. However, if the designated output buffer is full, i.e., the cell is stored in the shared-buffer, i.e., in the queue tail. The cells in the output buffer are scheduled for transmission in same order (e.g., FIFO or priority). The cells in the shared-buffer (in the queue tails) are transmitted to the output buffer in FIFO order.

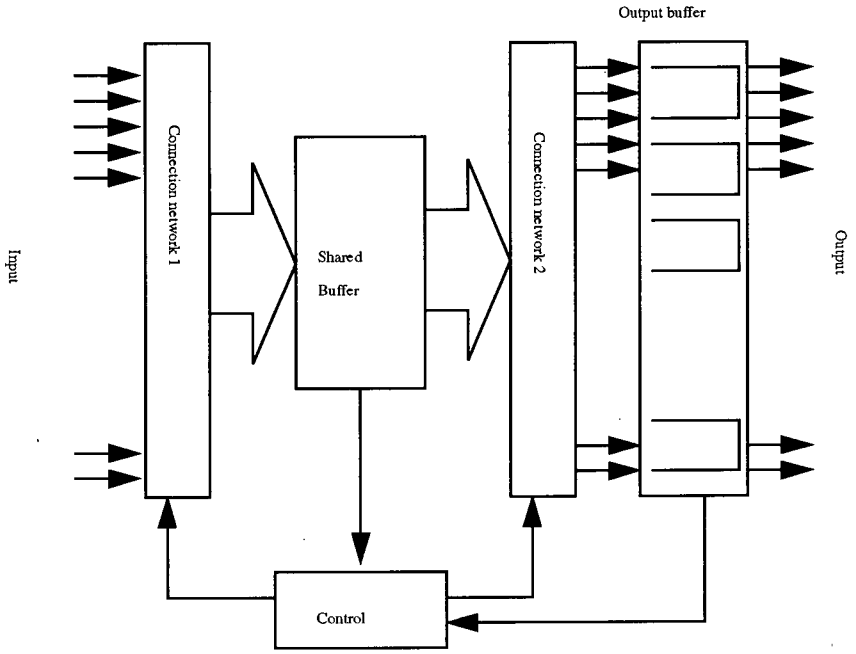

Figure 3.2 Hybrid buffering switch

3.3 Queue Tail Management with a Hybrid Buffering Switch

A hybrid buffer switch does not only offer a higher buffer utilization but also provides improved cell loss performance under bursty or non-uniform incoming cell traffic. The shared-buffer offers flexible cell management which is well suited for QoS control and congestion control. Long queue tails may be caused by bursty or non-uniform traffic stream. Such queue tails can be managed by applying dynamic buffer allocation policies in the shared-buffer. In the output buffers, short queue fronts are dedicated to each port, and they do not interact with one another. Therefore, our buffer allocation policy guarantees fairness among queue fronts, while improving

cell loss in the queue tails through buffer sharing.

In the shared-buffer, the delay sensitive cells, (e.g., CBR or real-time VBR) can be switched with high priority to the output buffer and also are dropped with high priority once they have passed their deadlines. The loss sensitive cells, (e.g., in connectionless ABR) are dropped and switched with low priority.

We develop a buffer management scheme that makes generic ATM switches capable of supporting ABR traffic in the presence of other delay-sensitive CBR or VBR traffic. The proposed scheme aims at enhancing the performance of ATM switches by maintaining the head cells of output queues in relatively short dedicated output buffers, while maintaining the (possibly long) tails of overflowing queues in a shared-pool managed by a more intelligent cell scheduling unit, called the queue tail manager (or QTM). Various congestion control and fair allocation functions can be exercised by the QTM, such as the techniques developed in [38].

Under this scheme higher priority (or delay-sensitive) cells can be forwarded immediately to the output buffer potentially blocking some lower priority (but loss-sensitive) cells coming from the shared-buffer. If the shared-buffer is full, then a suitable push-out scheme must be employed. The shared-buffer and the QTM greatly enhance the buffer utilization in output-buffered ATM switches. This is because the dedicated output buffer sizes can be kept small.

The total number of queue tails in the shared buffer can be smaller than the total number of logical queues in the switch because statistically it is likely that only a portion of the logical queue lengths are larger than the DOB size. The available shared-buffer can be shared-among a larger number of outputs than is possible with previous proposed schemes. This is because only queue

tails are stored in the SM and memory access is only operated upon the queue tails in the shared-buffer.

3.4 Distributed Buffer Control and Hybrid Scheduling

The both time priority scheme and space priority scheme can be implemented by the switch. The shared buffer and the output buffer hardware can support space priority scheme in which loss-sensitive cells have high priority service by partial sharing and push-back. Priority queueing in the switch buffers provides time priority schem in which delay-sensitive cells have high priority service.

The dedicated output buffer size or DOB is an important parameter[38]. The impact of the trade-off between the shared-buffer and dedicated buffer sizes on performance will be the subject of the performance evaluation in Chapter 5. The switch can employ distributed buffer control and priority scheduling to provide very flexible routing options. The output buffers can execute complex scheduling policies, such as priority queueing, deadline scheduling. The shared-buffer performs buffer size management and support multicast and broadcast.

By one-class priority queueing in the output buffers instead in the shared-buffer, the control complexity and implementation cost are optimal. If the DS sensitive cells are only saved in the output buffer, the DS cell delay can be guaranteed by the DOB size. With multi-class DS priority queueing output buffer, the highest priority cell delay can still be guaranteed by the DOB size. The shared-buffer saves LS cells only by simple FIFO and its speed and size can be

increased. THe DS cells will be dropped if the output buffer is full.

For compressed DS traffic with cell loss constraints, e.g., CBR, VBR, more complex scheduling control is required upon the switch. Those cells will stay in the hard buffer if the output buffer can not accept them because the output buffer is full of DS cells or cannot push back some low priority cells back into the shared buffer. Then priority queueing must be applied in the shared-buffer. The optimal priority queueing design is to implement only one-class priority queueing in the shared-buffer but multi-class priority in the output buffers. In this way, the degree of control complexity in the shared-buffer is low but cell delay will not be large because the DS cells are routed out first and DS logical queue tail length in the shared buffer will not be large.

In addition to the priority queueing in the shared buffer, not only multi-class priority queueing is implemented in the output buffer, but also push-back scheme is employed in the output buffer. Then the DS logical queue tail length in the shared buffer is shorter than that without push-back scheme in the output buffers. The efficient scheduling and buffering yet not complex implementation can support CBR and VBR traffic.

3.5 Methodology

In the following chapters, the proposed hybrid buffering switch performance is evaluated by queueing theory and simulation. The focus is on the most important performance measures of network in terms of delay jitter and cell loss probability/rate. The queueing analysis is applied to the switch model with FIFO queues and uniformly distributed independent traffic arrival model.

Simulation models a switch with FIFO queues in the shared-buffer and two priority scheduling policies in the output buffers under a two-class traffic model.

Chapter 4 Queueing Analysis

This chapter describes the derivation of a semi-closed form solution to the queueing model of the hybrid buffer switch. Specifically, a semi-closed form model is derived for the joint distribution of queue tails in the shared-buffer. Similar expressions are derived also for cell loss and cell delay. The analysis assumes uniformly distributed and independent input traffic. In the network, according to the research of [30], the traffic characteristic approaches Possion process. To obtain an efficient semi-closed form model in terms of queue length distribution, queue length tail distribution (cell loss), mean delay and delay jitter, in order to evaluate the performance of switch in the network is desirable.

4.1 Logical Queue Model

In this section develops a simple queueing model for the output logical queues in the proposed ATM switch model. As mentioned before a logical queue can be extended over two main sections of the switch: a queue in the dedicated buffer and a queue in the shared-buffer. We assume that time is divided into fixed-size time units, the arrival of an ATM cell is aligned with a time unit. The cells destined to a particular output port form a logical queue which is divided into two tandem queues, one in a dedicated output buffer (DOB) and one in the shared-buffer (SMB). The cells in any logical queue are serviced in FIFO order. The analytical modeling is based on the discrete-time queuing analysis developed in [31]. The basic definition of the model is given

below.

The cell traffic arriving at the switch with different destinations will form different logical queues. The cells lodged in the queue head (<DOB) are saved in the output buffer and those in the tail (>DOB) are buffered in the shared-buffer. The behavior of logical queues is same as a FIFO Geo/D/c queue.

To model the logical queue by a discrete time queueing model, the following simplifying assumptions are used:

1) The logical queue has infinite length

2) The capacity of a logical queue sever, i.e., the number of cells that can be read out of queue in one time units, is c (c>0). Therefore, we can assume that each output buffer transmits c cells over its output links in one unit of time.

3) Time is divided into fixed-length intervals, referred to as time units, such that one time unit suffices for the transmission of one cell via each of the output links of the buffer. The transmission of a cell via an output channel of a buffer starts at the beginning of a time unit and ends at the end of a time unit. Cells cannot leave the buffer at the end of the same time unit during which they arrive.

4) New cells enter the buffer according to a general uncorrelated arrival process, i.e., the number of cells arriving in the buffer during consecutive time units are

modeled as independent and identically distributed random variables with a general probability distribution, characterized by a probability generating function.

4.1.1 Logical Queue Length

To model the queue length process, let s_k denote the queue size (in cells) at the beginning of time unit k, and let $S_k(z)$ denote the probability generating function for s_k. Also let s and $S(z)$ denote the steady-state version (i.e., in the limit as $k \to \infty$) of s_k and $S_k(z)$ respectively.

$$s = \lim_{k \to \infty} s_k$$

Let $U(s_k)$ be the unit step function,

$$U(s_k) = \begin{cases} 1 & s_k \geq c \\ 0 & s_k < c \end{cases}$$

It can be easily seen that the buffer contents evolve according to the following system equation where s_k is the queue size and a_k is the total number of cell arrivals during time unit k

$$s_{k+1} = (s_k - c)^* + a_k,$$

$a = \lim_{k \to \infty} a_k$, and c is capacity of the queue (i.e., the number of cells served per time unit), and $(\ldots)^*$ denotes max(0,...). Using standard z-transformation techniques [33], it is possible to

derive the following expression for $S(z)$:

$$s_{k+1} = (s_k - c)U(s_k) + a_k$$

With $Z\{\ \}$ for the z-transformation, we get

$$S(z) = Z\{z^s\} = \sum_{k=0}^{\infty} \text{Prob}\{s = k\} z^k$$

$$= Z\{z^{(s-c)U(s-c)+a}\} = Z\{z^{(s-c)U(s-c)}\} Z\{z^a\}$$

$$= A(z)\left[A(z)\text{Prob}(s < c) + \sum_{k=0}^{\infty} \text{Prob}\{s = k\} z^{k-c}\right]$$

$$= A(z)\left[\sum_{i=0}^{c-1} P_i + S(z)z^{-c} - \sum_{i=0}^{c-1} P_i z^{i-c}\right]$$

where $A(z)) = Z\{z^a\}$ is the generating function of arrival.

$$S(z) = A(z)\left[S(z)z^{-c} + \sum_{i=0}^{c-1} P\ (1 - z^{i-c})\right]$$

$$= \frac{A(z) \sum_{i=0}^{c-1} (z^c - z^i)\text{Prob}(s = i)}{z^c - A(z)}$$

The above generating function for $S(z)$ contains c unknown constants, namely $\text{Prob}\{s = i\}$, for $i = 0, 1, 2, \ldots, c-1$. These can be determined by invoking the analyticity of

the generating function $S(z)$ inside the unit circle of the complex plane, and by using Rouche's theorem in [32][33]. The terms z^c and $A(z)$ are analytical, and z^c has c zeros inside the unit circle of the complex plane. Therefore $z^c - A(z)$ is analytical and has exactly c zeros in the unit circle of a complex plane.

The function $S(z)$ will at least converge inside the unit circle and it follows that the zeros of the denominator are also zeros of the numerator. Next we derive c equations to determine the c unknown constants P_0, P_i, ..., P_{c-1}, i.e.,

$$P_0 = \text{Prob}(s = 0), P_1 = \text{Prob}(s = 1), \ldots, P_{c-1} = \text{Prob}(s = c - 1),$$

and one of the equations is $S(1) = 1$. We get

$$\sum_{i=0}^{c-1} (z^c - z^i)\text{Prob}(s = i) = (c - A'(1))(z - 1)\prod_{i=1}^{c-1} \frac{z - z_i}{1 - z_i}$$

and the resulting equation for $S(z)$ is

$$S(z) = (c - A'(1))\frac{(z-1)A(z)}{z^c - A(z)}\prod_{i=1}^{c-1} \frac{z - z_i}{1 - z_i} \qquad (1)$$

The quantities z_i are the complex zeros of $z^c - A(z)$ strictly inside the unit circle of the complex z-plane, which can be computed numerically by means of the Newton-Raphson scheme.

The mean buffer content or mean queue length in steady state can be found by evaluating the first derivative of $S(z)$ at $z = 1$, yielding

$$E(s) = S'(1) = \frac{1}{2}\sum_{i=1}^{c-1}\frac{1+z_i}{1-z_i} + A'(1) \quad + \frac{A''(1)-(c-1)A'(1)}{2(c-A'(1))}$$

4.1.2 Cell Delay

In this section, we derive an expression for the probability generating function of the delay a cell experiences in the buffer, under a first-come-first-served (FIFO) queueing discipline [31]. This will permit the derivation of semi-closed-form expressions for several delay characteristics, such as the mean value of the cell delay and its variance.

The delay of a tagged cell is defined as the number of time units between its arrival and departure time. We assume that cell arrivals to the logical queue in the Ith time unit are transmitted over the output link in a random order. However, cells arriving in earlier time units must be transmitted first. Therefore, a cell delay has two components. First a cell must wait while cells have arrived in earlier time units are transmitted. Second, the cell must wait additional time units until it is randomly selected for transmission among the cell arrivals in its batch.

One must be careful when we compute the delay due to the transmission of other cell arrivals in the Ith time unit. Reference [3] points out that much of the standard work on queueing theory are in error when they compute the delay of a single server or multi severs queues with batch input.

Let I denote the time unit during which the tagged cell arrival occurs and, and let a_I be the number of cell arrivals during I. Owing to the fact that we consider an arbitrary arrival bulk process, we have

$$\text{Prob}[a_I = m] = \frac{m \cdot a(m)}{E[Arr]}, \quad m = 1, 2, 3, \ldots$$

where $E[Arr]$ denotes the mean number of cell arrivals per time unit, the probability of k cell arrivals during any time unit is $a(m)) = Prob[a = m]$. The probability that I contains m cells is proportional to the relative occurrence of time units with m cell arrivals and the number m itself. This is because our random arrival bulk size can be any of the m cells in a time unit during which the m cell arrivals occur.

Let the random variable d indicate the delay of an arbitrary cell in the steady state, and let $D(z)$ denote the corresponding probability generating function. Furthermore, let the random variable f describe the number of cells entering the buffer during the same slot as a tagged cell, but to be transmitted before this cell. Then

$$f(n) = \text{Prob}[a_I = m] \cdot g[n+1|m], \quad n = 0, 1, 2, 3, \ldots$$

Where $g[n+1|m]$ denotes the probability that our randomly chosen tagged cell is the $(n+1)th$ cell to be served during time unit I, provided that I contains m cells. Thus

$$g[n+1|m] = \frac{1}{m}$$

$$f(n) = \frac{\sum_{m=n+1}^{\infty} a(m)}{E[Arr]}$$

$$F(z) = \frac{A(z)-1}{(z-1)A'(1)}$$

The delay of a tagged cell follows a function of the buffer contents at the beginning of the arrival time unit of the tagged cell and the random variable f associated with this cell. Therefore, the delay d is given by

$$d = \frac{f + (s-c)^*}{c}$$

where $f + (s-c)^*$ represents the number of cells to be served before the tagged cell at the end of its arrival time unit. Also c is the output link capacity (number of cells served per time unit) and s is the queue length. Defining

$$r = f + (s-c)^*, \qquad d = \frac{r}{c},$$

$$r = (d-1)c$$

We have

$$\text{Prob}[d = i] = \sum_{j=0}^{c-1} \text{Prob}[r = (i-1)c + j], \quad i > 1.$$

The probability generating function for the above expression is

$$D(z) = \sum_{i=1}^{\infty} z^i \sum_{j=0}^{c-1} \text{Prob}[r = (i-1)c + j], \quad i > 1.$$

and it follows that

$$D(z^c) = z^c \sum_{j=0}^{c-1} z^{-j} \sum_{i=0}^{\infty} z^{ic+j} \text{Prob}[r = ic + j]$$

Using the Kronecker delta function

$$\delta(n) = \begin{cases} 1, n = 0 \\ 0, n \neq 0 \end{cases}$$

We obtain

$$D(z^c) = z^c \sum_{j=0}^{c-1} z^{-j} \sum_{i=0}^{c-1} \sum_{k=0}^{\infty} z^k \text{Prob}[r = k] \delta(k - ic - j)$$

Now, note the identity

$$\frac{1}{c} \sum_{n=0}^{c-1} g^{mn} = \sum_{i=-\infty}^{\infty} \delta(m - ci), h = e^{\left(\frac{2\pi i}{c}\right)},$$

(where i is the imaginary unit) that can eliminate the Kronecker delta function in the above expression. This results in

$$D(z^c) = \frac{z^c}{c} \sum_{j=0}^{c-1} \sum_{n=0}^{c-1} \left\{ \sum_{k=0}^{\infty} h^{n(k-j)} z^{k-j} \text{Prob}[r = k] \right\}$$

Defining $R(z)$ as the probability generating function of the random variable r, leads to

$$D(z^c) = \frac{z^c}{c} \sum_{j=0}^{c-1} \sum_{n=0}^{c-1} (h^n z)^{-j} R(h^n z)$$

Which, after eliminating the summation over j, becomes

$$D(z^c) = \frac{z^c}{c} \sum_{n=0}^{c-1} \frac{z^c - 1}{1 - (h^n z)^{-1}} R(h^n z),$$

where we have used the property $h^{nc} = 1$.

On the other hand, $R(z)$ can be expressed in terms of $S(z)$ and the quantities $\text{Prob}(s = j)$, as follows

$$R(z) = F(z)\left[S(z)z^{-c} + \sum_{j=0}^{c-1} P_j(1 - z^{j-c})\right]$$

Which results in

$$D(z^c) = \frac{c - A'(1)}{cA'(1)} \sum_{n=0}^{c-1} \frac{z^c - 1}{1 - (h^n z)^{-1}} \frac{A(h^n z) - 1}{z^c - A(h^n z)} \prod_{j=1}^{c-1} \frac{h^n z - z_j}{1 - z_j} \qquad (2)$$

This equation provides us with the desired explicit formula for the probability generating function of the cell delay, albeit for the argument z^c instead of z. The function is expressed in terms of the generating function $A(z)$ of the arrival process, the $c-1$ complex zeros of $z^c - A(h^n z)$ in the unit circle, and the c complex roots of the equation $z^c - 1$.

Thus, the average delay of a cell can be obtained by the first order derivative.

$$E(d) = D'(1) = \left.\frac{d(D(z^c))}{cdz}\right|_{z=1}$$

$$= \frac{1}{A'(1)}\left[\frac{1}{2}\sum_{j=1}^{c-1}\frac{1+z_j}{1-z_j} + E'(1) + \frac{A''(1)-(c-1)A'(1)}{2(c-A'(1))A'(1)}\right]$$

4.2 Performance Evaluation of Hybrid Buffer Switch

The above queueing model can be used to assess the performance of the proposed hybrid buffer switch. More specifically, consider a switch with N input links and N output links, which receives cells for different destinations with equal probabilities. The arrival process at each of the input links is assumed to be of Bernoulli type independent, identical distribution process, i.e., the probability of having a cell arrival during any time unit is p, the probability that there is no arrival is $1-p$, and arrival during consecutive time units are independent. Arriving cells are routed uniformly to one of the logical queues, i.e., the probability of having an arrival in a tagged logical queue coming from any given input link during any given time unit is equal to p/N. In these circumstances, each logical queue can be modeled as a discrete-time queueing system of the type discussed above. The delay and queue size of the logical queues of the switch can be investigated by concentrating on one tagged queue, since cells with different destinations do not influence each other's delay and since all logical queues exhibit the same statistical behavior.

Specifically, the queueing model used here is the Geo/D/c discrete-time queueing system, in which the cell arrival generating function is given by

$$A(z) = \left(1 + \frac{p}{N}(z-1)\right)^N$$

which is the generating function of a binomial distribution with parameters N and mean p. As N approaches infinity, the arrival process $A(z)$ becomes a Possion distributed random variable with mean p and a generating function $A(z) = e^{p(z-1)}$.

4.2.1 Cell Delay

Under normal operation, the cells in the hybrid buffering switch are routed out of the output buffer and cells in the shared-buffer are transmitted to the output buffers corresponding to their respective logical queues. The cell delay in the switch is determined by the cell delay in the logical queue, which is given by Eq. (2)

$$D(z^c) = \frac{c - A'(1)}{cA'(1)} \sum_{n=0}^{c-1} \frac{z^c - 1}{1 - (h^n z)^{-1}} \frac{A(h^n z) - 1}{z^c - A(h^n z)} \prod_{j=1}^{c-1} \frac{h^n z - z_j}{1 - z_j}$$

4.2.2 Queue Length in the shared-Buffer

In the shared-buffer of the hybrid switch, every logical queue tail holds cells destined to a particular output port and the total queue length in the shared-buffer is the sum of all logical queue

tail lengths. The logical queue tail length is a random variable q_i, with a given distribution. Therefore, the queue length of the shared-buffer is also a random variable q_s. The joint distribution of these logical queue tail length variables is same as the distribution of the queue length variable in the shared-buffer. Thus,

$$q_s = \sum_i q_i$$

With distribution $P_{it} = \text{Prob}[q_i = t]$, and $P_{st} = \text{Prob}[q_s = t]$, t is queue length.

Because the input traffic is assumed to be uniformly distributed with independent arrivals, the logical queue length variables are i.i.d. variables. The joint distribution of these i.i.d. variables is the distribution of the queue length variable in the shared-buffer. Applying convolution to this i.i.d. process, the joint distribution can be obtained from

$$P_{st} = P_{1t} \otimes P_{2t} \otimes P_{3t} \cdots P_{(n-1)t} \otimes P_{nt}$$

where n is the number of logical queues in the shared-buffer or switch size and $\otimes$ is the convolution operator.

Note that a logical queue tail length m in the shared-buffer is related to the logical queue length by the formula

$$m = s - k,$$

where s is the total logical queue length of a particular output port and k is the output buffer size.

From Eq. 1 and by applying the approach outlined in Appendix A. and [34] for the case

$q = t$, we get

$$P_{it} = \text{Prob}[q = t] \cong -\left(\frac{b_q}{z_0}\right) z_0^{-t}$$

where z_0 is the smallest pole outside unit circle, and t is the queue length, and

$$b_q = \frac{T(z_0)}{N'(z_0)} = (c - A'(1)) \frac{z_0 - 1}{c z_0^{c-1} - A'(z_0)} \prod_{j=1}^{c-1} \frac{z_0 - z_j}{1 - z_j}$$

z_j is a root of $z^c - A(z)$.

The above expression for q_{it} gives the overall distribution of the logical queue length variable. By applying convolution on the distribution of the logical queue tail length variable in the shared-buffer, we can derive the queue length distribution for the shared-buffer. From the joint distribution, we can obtain any measures of the queue length variable in the shared-buffer, including the mean queue length, all orders of moments, and the cell loss probability in a finite size shared-buffer.

Define k as the output buffer size and m as the logical queue tail length in the shared-buffer, then we obtain

$$P'_{im} = P_{i(k+m)} = \text{Prob}[q_i = k + m] \cong -\left(\frac{b_q}{z_0}\right) z_0^{-(m+k)} = -\left(\frac{b_q z_0^{-k}}{z_0}\right) z_0^{-m}$$

When the switch has two output port only, i.e., two logical queue tails in the shared-buffer,

the probability of having a shared-buffer queue length of m is derived by

$$P_{s(2m)} = \text{Prob}[q_s = m] = P_{1m} \otimes P_{2m}$$

where $P_{1m} = P_{2m}$.

$$q_{s2(m)} = \sum_{j=0}^{m} P_{1(m-j)} P_{1j} = \sum_{j=0}^{m} \left[-\left(\frac{b_q z_0^{-k}}{z_0} \right) z_0^{-(m-j)} \right] \cdot \left[-\left(\frac{b_q z_0^{-k}}{z_0} \right) z_0^{-j} \right]$$

$$= \left(\frac{b_q z_0^{-k}}{z_0} \right)^2 \sum_{j=0}^{m} z_0^{-m} = (1+m) \left(\frac{b_q z_0^{-k}}{z_0} \right)^2 z_0^{-m}$$

In the switch with four ports, the above formula becomes

$$P_{s4(m)} = P_{1m} \otimes P_{2m} \otimes P_{3m} \otimes P_{4m} = \sum_{j=0}^{m} P_{s2(m-j)} P_{s2(j)}$$

$$= \sum_{j=0}^{m} \left[(m-j+1) \left(\frac{b_q z_0^{-k}}{z_0} \right)^2 z_0^{-(m-j)} \right] \cdot \left[(j+1) \left(\frac{b_q z_0^{-k}}{z_0} \right)^2 z_0^{-j} \right]$$

$$= \left(\frac{b_q z_0^{-k}}{z_0} \right)^4 z_0^{-m} \sum_{j=0}^{m} [(m+1-j)(j+1)]$$

Which can be rewritten in the form:

$$P_{s4(m)} = \left(\frac{b_q z_0^{-k}}{z_0}\right)^4 z_0^{-m}[M \otimes M]$$

where M is an $m+1$ dimensional vector of $M = [1, 2, 3, \ldots, m+1]$

For the switch with eight ports, the queue length distribution is given by

$$P_{s8(m)} = \sum_{j=0}^{m} P_{s4(m-j)} P_{s4(j)} = \left(\frac{b_q z_0^{-k}}{z_0}\right)^8 z_0^{-m}(M \otimes M \otimes M \otimes M)$$

which can be generalized for any switch size of w, $w = 2, 4, 8, \ldots 2^i$ as follows

$$P_{sw(m)} = \left(\frac{b_q z_0^{-k}}{z_0}\right)^w z_0^{-m}(\underbrace{M \otimes M \otimes M \ldots M \otimes M}_{w/2}) \qquad (3)$$

This equation provides the required semi-closed-form expression for the queue length distribution in the shared-buffer.

4.3 Tail Distribution in Hybrid Buffering Switch Model

4.3.1 Cell Delay Jitter

Under normal operation, the cells in the hybrid buffering switch are routed out of the output buffer and cells in the shared-buffer are transmitted to the output buffers corresponding to their respective logical queues. The cell routing is supposed to operate within the delay of one

time unit. Becasue of buffering, a cell may may be switched over the delay of the referenced one time unit. The cell delay variation (CDV) or delay jitter is the difference between the actual cell delay and the reference delay. The negative delay jitter (or CDV) is that the actual cell delay is larger than the reference delay. We derive the (negative) delay jitter of the switch by the knowing quantity D (refence delay) of the tail distribution of delay and is given by

$$\text{Prob}[d \geq D]$$

We have obtained the generating function of the delay in

$$D(z^c) = \frac{c - A'(1)}{cA'(1)} \sum_{n=0}^{c-1} \frac{z^c - 1}{1 - (h^n z)^{-1}} \frac{A(h^n z) - 1}{z^c - A(h^n z)} \prod_{j=1}^{c-1} \frac{h^n z - z_j}{1 - z_j}$$

where $h = e^{2\pi i/c}$ and z_j are the $c-1$ roots of $z^c - A(z)$ complex function inside the unit circle.

According to Appendix A, we note that the probability generating function does not satisfy the condition that there is only one pole for which the modulus is minimal. Indeed, if z_0 is the zero of $z^c - A(z)$ outside the unit circle with the smallest modulus, then $z^c - A(z)$

has n roots with the same absolute value, i.e., the complex quantity $h^{-n} z_0$ for all n between 0 and $c-1$. This implies that the method used in Appendix A to obtain the approximation for the tail distribution of a random variable, can not be applied directly. Nevertheless, we are still able to derive an expression for the tail distribution of the delay, by taking into account all the poles of

$D(z^c)$ with the minimal modulus. Thus, we get

$$\text{Prob}[d = k] \cong -\sum_{n=0}^{c-1} \frac{b(n)h^n}{z_0}(z_0 h^{-n})^{-kc}$$

where $b(n) = \dfrac{T(z_0 a^{-n})}{N'(z_0 a^{-n})}$. and $T(z)$ and $N(z)$ are the numerator and the denominator, respectively, of $D(z^c)$. The above equation can be rewritten as

$$\text{Prob}[d = k] \cong -b_d \sum_{n=0}^{c-1} \left(\frac{h^n}{z_0}\right)^{kc}$$

where

$$b_d = \frac{c - A'(1)}{cA'(1)} \frac{z_0{}^c - 1}{z_0 - 1} \frac{A(z_0) - 1}{cz_0{}^{c-1} - A'(z_0)} \prod_{j=1}^{c-1} \frac{z_0 - z_j}{1 - z_j}$$

Using the property that $h = e^{2\pi i/c}$ and $h^{nc} = 1$, yields

$$\text{Prob}[d = k] \cong -cb_d z_0{}^{-kc}$$

From this equation, we can easily derive an expression for the probability that the cell delay exceeds a given bound, i.e., the delay jitter as follows

$$\text{Prob}[d > D] \cong -cb_d \frac{z_0{}^{-Dc}}{z_0{}^c - 1}$$

Substituting for b_d yields

$$\text{Prob}[d > D] \cong -\frac{c - A'(1)}{cA'(1)} \frac{z_0^{-Dc}}{z_0 - 1} \frac{A(z_0) - 1}{cz_0^{c-1} - A'(z_0)} \prod_{j=1}^{c-1} \frac{z_0 - z_j}{1 - z_j} \qquad (4)$$

It should be noted that in order to be able to actually calculate the this quantity, we must first determine the roots inside the unit circle and the root with the smallest modulus outside the unit circle of $z^c - A(z)$. This can be done using the Newton-Raphson algorithm.

4.3.2 Cell Loss Rate

The cell loss ratio can be derived from the tail distribution of queue length represented by

$$\text{Prob}[q \geq Q]$$

Incoming cells are attached to the tail of logical queues corresponding to their output ports. An incoming cell destined to a full output buffer will be stored in the shared-buffer. Cells always proceed from the head of queue tails in the shared-buffer into output buffers. No cell will be moved out of the shared-buffer to a full output buffer. Therefore cell loss can occur only in the shared-buffer. Consequently, the queue length tail distribution of the share buffer determines the cell loss rate of the switch.

For switch with size of w, $w = 2, 4, 8, \ldots 2^i$, from Eq. 3, we have

$$Q_{wm} = \left(\frac{b_q z_0^{-k}}{z_0}\right)^w z_0^{-m}(\underbrace{M \otimes M \otimes M \ldots M \otimes M}_{w/2})$$

where M is an $m+1$ dimensional vector of $M = [1, 2, 3, \ldots, m+1]$ .For a queue length larger than the threshold Q , the above probability is the cell loss rate

$$Q_m(m > Q) = \sum_{m=Q}^{\infty} \left(\frac{b_q z_0^{-k}}{z_0}\right)^w z_0^{-m}(\underbrace{M \otimes M \otimes M \ldots M \otimes M}_{w/2})$$

$$Q_m(m > Q)' = 1 - \sum_{m=0}^{Q} \left(\frac{b_q z_0^{-k}}{z_0}\right)^w z_0^{-m}(\underbrace{M \otimes M \otimes M \ldots M \otimes M}_{w/2}) \qquad (5)$$

The above expression, provides the required simple equation for the cell loss rate of the shared-buffer and also of the switch. In Chapter 5, the numerical results of the model are presented and verified with simulation.

4.3.3 Performance Evaluation

The queuing model developed in this Chapter provides a set of equations, in a semi-closed or semi-semi-closed form, for the delay jitter and cell loss performance of the proposed switch under uniform independent traffic. The equations can be solved easily by numerical techniques such as Newton-Raphson method.

Figure 4.1 plots the cell delay jitter of the switch under different input loads for a switch with 16 ports, and where output buffer size is 40 cells (per port) and shared-buffer size is 10 cells (overall), respectively. The output buffer capacity is taken to be one (i.e., $c = 1$). Clearly, the delay jitter increases as the load becomes heavier. The switch performance has been also verified with simulation and it is found out that the numerical results are very close to the simulation at lighter input traffic loads. At higher loads, the analytical model provides an upper-bound of the measure compared to simulation. The difference is only in the sixth decimal digit.

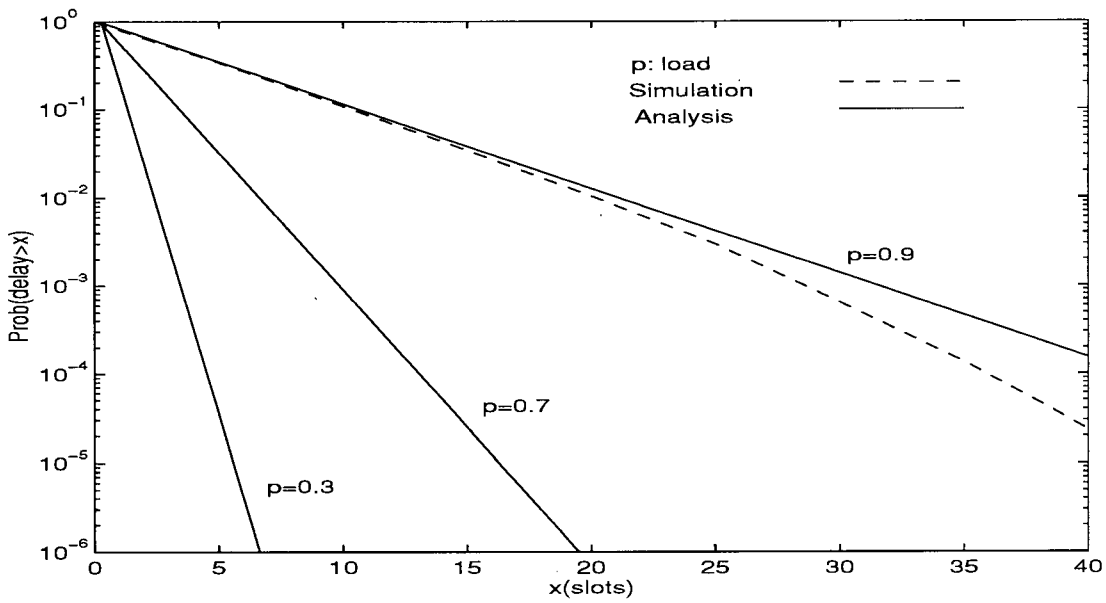

Figure 4.1 Delay jitter with different loads (16 x16 switch)

Figure 4.2 shows the cell loss probability as a function of the shared-buffer size under a load $p = 0.9$, switch size of 2 ports, an output buffer size of 5 cells (per Port) and a shared-buffer size of 10 cells (overall). With larger output buffer size, the cell loss is smaller. When the shared-

buffer size increases, the cell loss is also reduced. The switch performance has been also verified with simulation and it is found out that the numerical results are very close to the simulation. Figure 4.3 reveals the cell loss probability as a function of shared-buffer size under load $p = 0.9$, switch size of 4 ports, with an output buffer of size 5 cells. When the shared-buffer size increases, the cell loss is reduced. The switch performance has been also verified by simulation and it is found out that the numerical results provide an the upper-bound to the simulation.

In the approximate model, the buffer size is assumed to be infinite and all incoming cells are buffered and routed eventually (i.e. no cells are dropped from the buffer). In the simulation, however, the buffer size is finite and cells will be dropped when the buffer is full. Therefore, the queue length in the simulation is shorter than in the analytical model and, consequently, delay jitter and cell loss are smaller in simulation. The delay model can be applied to larger switch sizes with small error. Unfortunately, the cell loss model can not be applied to large switch sizes because the error is large.

Figure 4.4 compares the cell loss probability for several classes of switches ranging from pure-output buffered switches (i.e., shared-buffer size is zero), to completely shared-buffer switches (i.e., output buffer size is zero), and including different levels of hybrid buffering. Dedicated output buffering gives the largest cell loss rate while the completely shared-buffering gives the smallest. The hybrid buffering has cell loss rate in between the other two buffering schemes.

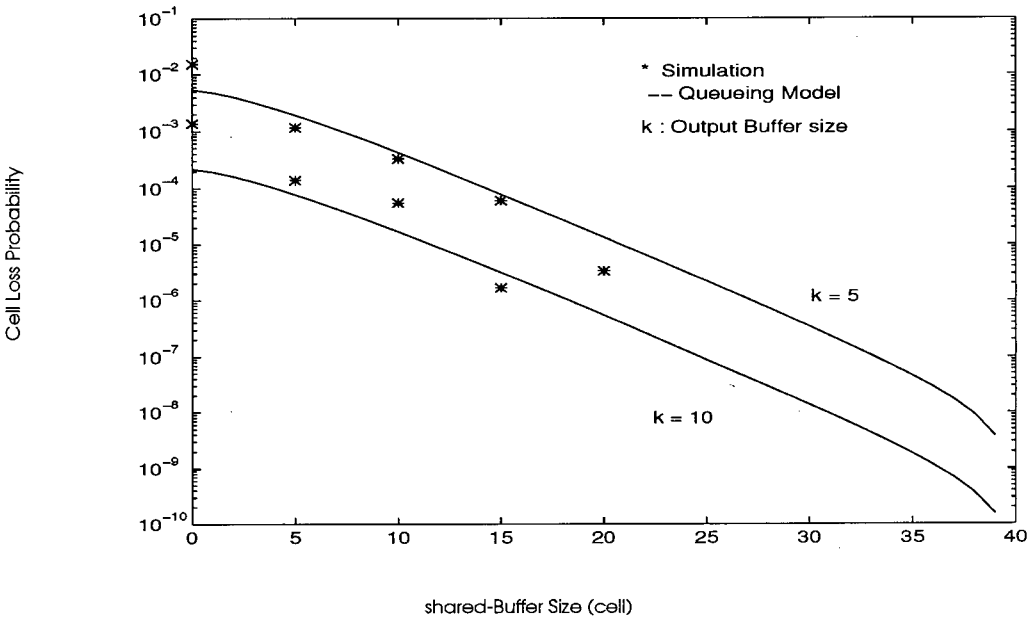

Figure 4.2 Cell loss rate with different output buffer size (load p=0.9, 2 x 2 switch)

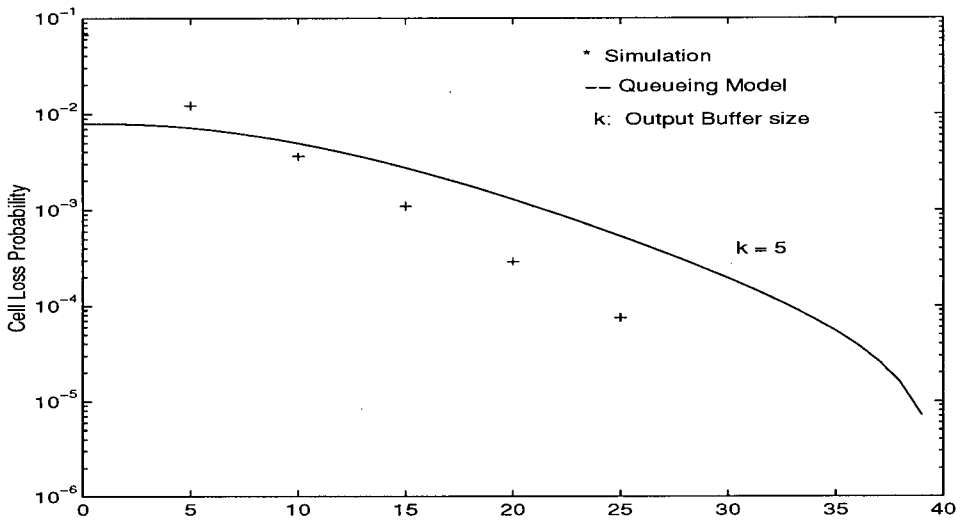

Figure 4.3 Cell loss rate (load 0.9, 4 x 4 switch)

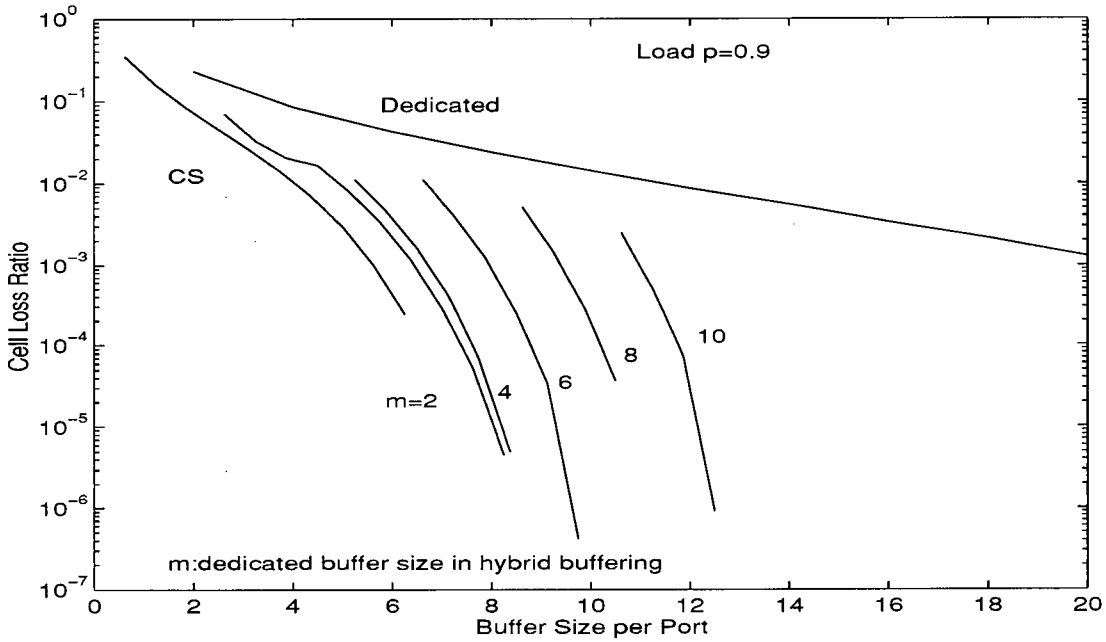

Figure 4.4 Cell loss rate comparison (16 x 16 switch)

Chapter 5 Simulation and Results

In this chapter, the simulation design is described. The switch model, traffic model, simulation parameters, numerical and simulation results are presented.

5.1 Traffic Model

5.1.1 Single-Class Traffic

To verify the queueing analysis, the single-class traffic model is used in the simulation [3]. The arrival process at each of the input links is assumed to be of Bernoulli type (i.i.d), i.e., the probability of having a cell arrival during any time unit is p, the probability that there is no arrival is $1-p$, and arrivals during consecutive time units are independent. The input cells have N destination addresses uniformly distributed. The probability of k cells arriving at a particular output port is given by:

$$a_k = \text{Prob}[a = k] = \binom{N}{k}\left(\frac{p}{N}\right)^k\left(1-\frac{p}{N}\right)^{N-k}, \quad k = 0, 1, 2, \ldots, N$$

5.1.2 Two-class Traffic

The simulation is designed to compare the performance of the hybrid buffer switch with that of output buffer switch [15] which uses priority queueing in output buffers. Two classes of

traffic are of concern, the high priority and the low priority. The high priority class traffic is delay-sensitive but can tolerate given cell loss, such as voice. Voice traffic requires short transmission delay because human hearing is sensitive to delay in communication. But certain amount of information loss in voice communication does not degrade the voice quality of understandability. The low priority class traffic is loss-sensitive but can tolerate given delay, such as data. Data traffic between computers can tolerate given delay, but any information loss in data traffic will generate obvious error.

The input cell streams consist of two types of cells: delay-sensitive (DS) cells and loss-sensitive (LS) cells. The DS cells have high priority in routing and tight delay jitter requirement, so the switch will route those cells out of buffers with higher priority than LS cells. If a cell delay is lager than the deadline, or buffers are full and congestion happens inside the network, the DS cell can be dropped from the buffers to release the buffer space and bandwidth, accordingly, or to alleviate the congestion.

However, the LS cells can sustain higher delay but have tight cell loss rate requirement, so the switch will route the cells in buffers with lower priority than DS cells. If buffers are full and congestion happens, the cells have to be stored in buffers and transmitted later when bandwidth is resumed resulting from the dropping of DS cells.

According to the simple traffic model in [15], the arrival process at each of the input links is assumed to be of Bernoulli type (i.i.d), i.e., the probability of having a cell arrival during any time unit is p, the probability that there is no arrival is $1-p$, and arrivals during consecutive time units are independent. The total number of cells destined for one output port in a certain time slot is a. The input cells have N distinct destination addresses uniformly distributed. The

probability of k cells arriving at a particular output port is given by:

$$a_k = \text{Prob}[a = k] = \binom{N}{k}\left(\frac{p}{N}\right)^k\left(1-\frac{p}{N}\right)^{N-k}, \quad k = 0, 1, 2, \ldots, N$$

Let P_v be the probability that an incoming cell is a high priority one, let a_d and a_v be the number of high priority and low priority cells, respectively, reaching a particular output in a given time slot, then the probability of k cells with high priority arriving at a particular output port is given by

$$v_k = \text{Prob}[a_v = k] = \sum_{m=k}^{N} a_m\binom{m}{k}p_v{}^k(1-P_v)^{m-k}, \quad k = 0, 1, 2, \ldots, N$$

The probability of k cells with low priority arriving at a particular output port is given by

$$d_k = \text{Prob}[a_d = k] = \sum_{m=k}^{N} a_m\binom{m}{k}p_v{}^{m-k}(1-P_v)^k, \quad k = 0, 1, 2, \ldots, N$$

The above described two priority traffic is applied to the hybrid switch model introduced in Chapter 3 (Figure 3.2). The switch employs scheduling policy to manage the transmission of the cells from the two priority traffic classes. Additionally, the switch employs a buffer management policy in the shared-buffer to manage cell loss for LS class. In the following, we describe two alternative schemes that combine cell scheduling with buffer management in the hybrid switch, and evaluate their performance through extensive simulation.

5.1.3 Policy 1: Favor Loss-sensitive Cells

The shared-buffer contains a number of logical queue tails, with the cells in each queue tail served in FIFO order, i.e., the head cell of queue tail in the shared-buffer is forwarded to the corresponding output buffer. Each output buffer implement a priority queueing policy that transmits DS cells before LS cells, i.e., an LS is transmitted out of the buffer only when there are no DS cells in the buffer. Policy 1 works as follows:

1. An incoming DS cell is forwarded directly to its output buffer where it will be stored if the buffer is not full. Otherwise, the DS cell is discarded.

2. An incoming LS cell will be also stored directly in its output buffer if there is space available in the buffer. Otherwise, the LS cell is stored in the shared-buffer in the appropriate queue tail. The shared-buffer implement a "push-out" buffer management policy, in which the LS cell at the end of the longest queue tail is discarded when the shared-buffer is full to make room for the incoming LS cell.

3. Contention may arise between an LS cell at the head of a logical queue tail (in SMB) and between an incoming DS cell destined to the same output buffer. In this case, contention is resolved by forwarding the LS cell into the buffer and discarding the DS cell.

This policy is useful when the LS class corresponds to a non-real-time VBR service with very low cell loss requirement, while the DS service corresponds to a real-time CBR or VBR service with less strict cell loss requirements.

5.1.4 Policy 2: Favor Delay-sensitive Cells

As with Policy 1, the shared-buffer has logical queue tails for LS cells only, and the output buffer implements a priority queueing policy that transmits DS cells before LS. Also, the forwarding of LS and DS cells is the same as in Policy 1 when there is no contention for an output buffer.

When contention arises between an LS cell at the head of a logical queue tail (in the SMB) and an incoming DS cell destined for the same output buffer, the DS is pushed into the output buffer and the LS cells waits in its queue tail. since the LS cell is not discarded during contention, this means that Policy 2 implements a back pressure policy with respect to the LS traffic class, and a cell discarding policy for the DS class when the output buffer is full.

The policy is effective when the LS class corresponds to an ABR service with very low cell loss requirements, while the DS class corresponds to a real time VBR or CBR service also with low cell loss requirements.

5.2 Numerical Results

5.2.1 Two-class Traffic Management with Policy 1

Figures 5.1 to 5.6 show the performance measures of cell loss rate and cell delay jitter for both classes of traffic using Policy 1 for a hybrid switch with 16 ports, p= 0.9 loading, where 10-50% of that load is dedicated to the DS class. The results were obtained through extensive simulation. Figure 5.1 indicates the delay jitter of the LS cells with shared-buffer sizes of 10 cells (overall) and output buffer size of 5 and 20 cells (per port), respectively. When the DOB size is 5

cells, the probability of a delay jitter of 10 slots (with 90% LS traffic) is 4.227356e-04; and 8.499796e-02 DOB size is 20 cells. Note that as the output buffer space grows, the delay jitter is also increasing because more cells can enter the output buffer. When the percentage of LS cells decreases, their delay jitter increases because the output buffer gives higher priority to a large number of DS cells.

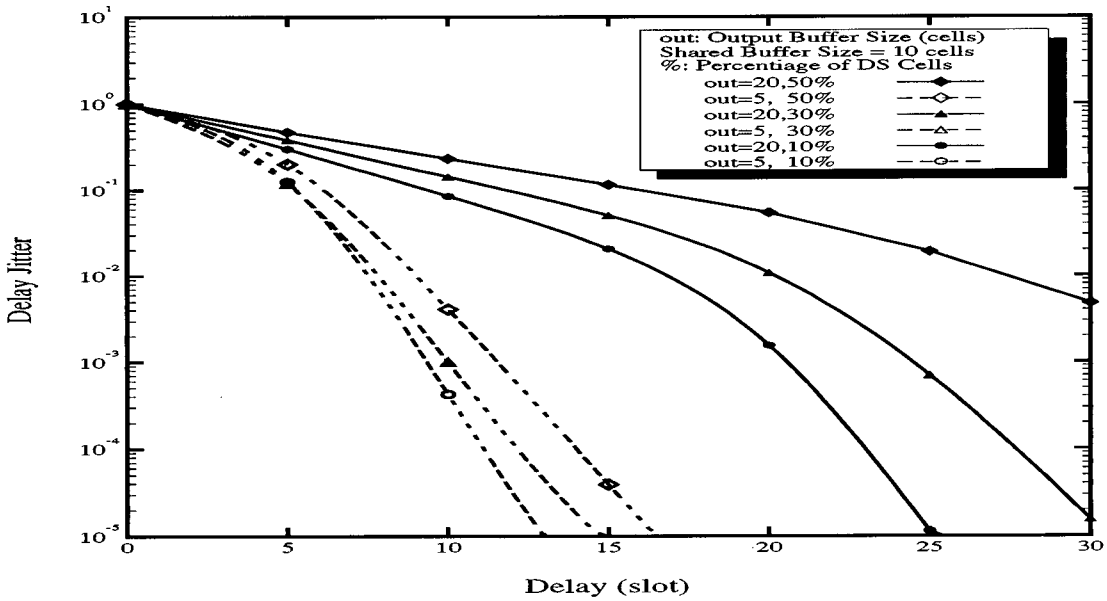

Figure 5.1 Delay jitter of loss-sensitive (LS) cells under Policy 1 (16 x 16 switch, load $p = 0.9$)

Figure 5.2 shows the DS cell delay jitter associated with output buffer sizes of 5 and 20 cells (per port), respectively. With an output buffer size of 5 cells and 50% DS cells, the probability that the delay jitter exceeds 3 slots is 4.983869e-03. With an output buffer size of 20 cells, the probability is 1.535697e-02. Note that as the size of output buffer decreases, delay jitter is reduced for DS cells because fewer cells can stay in the buffer. With higher percentage of DS

cells, the DS cell jitter increases because more DS cells enter the output buffer.

Figure 5.3 reveals the behavior of delay jitter for both traffic classes as a function of the load p, for an output buffer size of 5 cells (per port) and a shared-buffer size of 10 cells (overall). For a better observation, the plots are focused on a range of delay jitter (i.e., >3-5 slots). Note that the LS delay jitter is much larger than that of DS cells. The delay jitter difference between the two traffic classes is high when the percentage of DS cells is smaller. But with equal loading from DS and LS cells, the difference in delay jitter diminishes as the overall load is increased.

Figure 5.4 shows that the cell loss probability for the LS traffic class decreases rapidly as the overall buffer size per port increases (the overall buffer size per port is calculated by averaging the shared-buffer size over all 16 ports plus the output buffer size). When the DOB size is 5 cells and the overall buffer size per port is 5 cells, (i.e., shared-buffer size is zero). This situation corresponds to a completely dedicated output buffering scheme, the cell loss probability is 5.034447e-02. When buffer size per port is 6.25 cells, (i.e., shared-buffer size is 20 cells), the cell loss probability (with 90% LS cells) is 3.697112e-04. Note that increasting shared-buffer size can efficiently reduce the cell loss probability for LS traffic. In comparison, increasing the output buffer size to 20 cells (per port), i.e. with shared-buffer size equal to zero, improves the cell loss probability of LS traffic slightly. Under independent uniform traffic, fairness buffering and cell loss improvement of the output buffers are not obvious. The percentage of LS cells in the overall input traffic will change the cell loss probability but only slightly. As the percentage of LS cell rises, the cell loss probability increases due to more LS cells entering the shared-buffer and output buffers.

Figure 5.5 plots the cell loss ratio with different sizes of output buffer for DS traffic.

Because DS cells can not enter the shared-buffer, the shared-buffer size does not have a significant effect except when shared-buffer size is very small (or zero). With 10% DS cells, a shared-buffer size of 10 cells (per port), and an output buffer size of 5 to 10 cells, the DS cell loss probability slides down to 6.218715e-02. With a shared-buffer size of 0 cells (completely dedicated output buffer), it goes down from 4.983689e-02 to 1.264333e-02. The cell loss probability with no shared-buffer is smaller than that with a shared-buffer size of 10 cells because there are no LS cells in the shared-buffer to compete with DS cells entering the output buffer. The dashed curves in Figure 5.5 also show that with no shared-buffer, the percentage of DS cell traffic does not have a significant impact on the cell loss probability. This follows from the fact that, in each slot, only LS cells arrivals will knock out the DS cells at the output buffer input. However, with a shared-buffer size of 10 cells, a higher percentage of DS cells results in lower DS cell loss rate because fewer LS cells enter the output buffer.

Figure 5.6 shows the cell loss probability for both traffic classes as a function of the load p, for an output buffer size of 5 cells (per port) and a shared-buffer size of 10 cells (overall). The cell loss probabilities increase with the rising load. However, the loss ratio of LS cells is much smaller than that of DS cells. The higher the percentage of LS traffic, the lower the LS cell loss ratio rate For DS traffic, when the load is larger than 0.8, the cell loss probability decreases as the percentage of DS cells increases. When the load smaller than 0.8, the cell loss probability increases as the percentage of DS cells increases. The total load and DS relative load differ the performance.

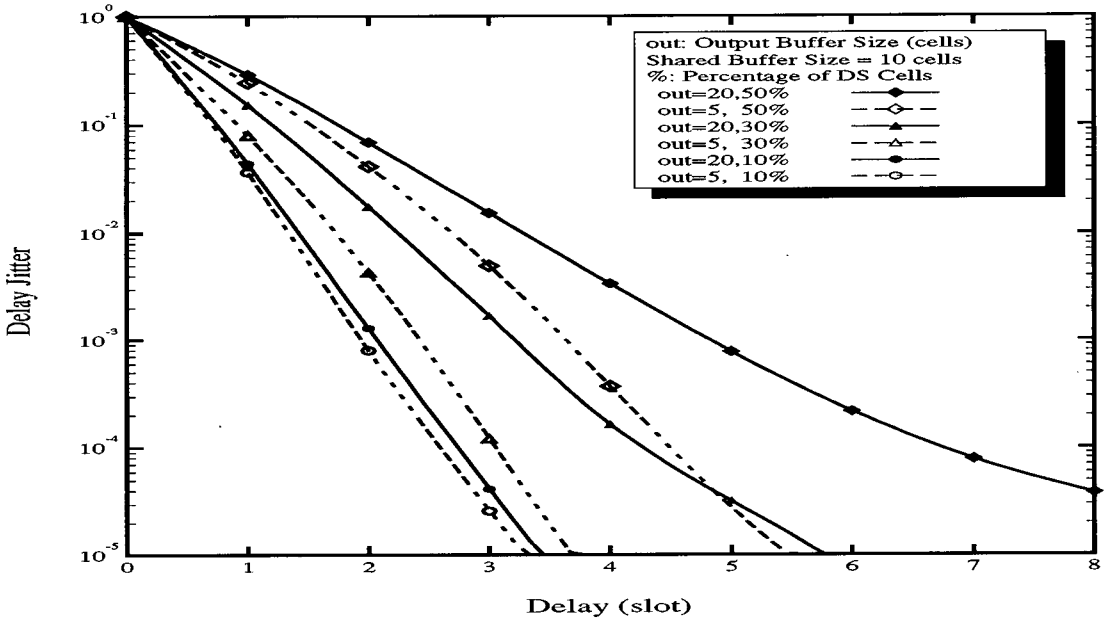

Figure 5.2 Delay jitter of delay-sensitive (DS) cells under Policy 1 (16 x 16 switch, load p = 0.9)

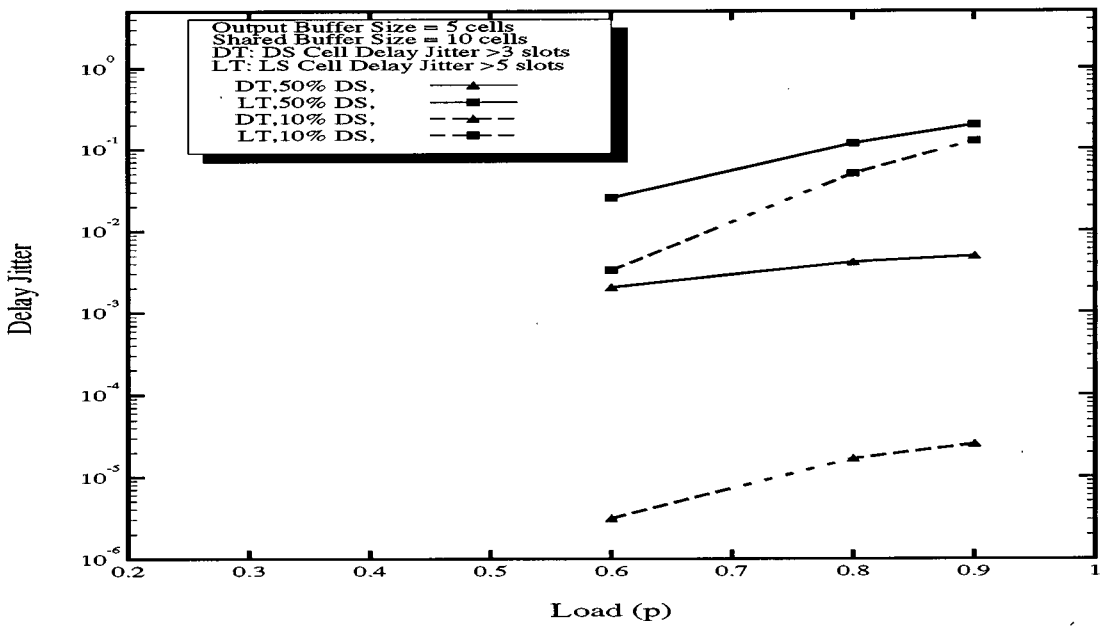

Figure 5.3 Delay jitter comparison of two-class traffic under Policy 1 (16 x 16 switch, load p = 0.9)

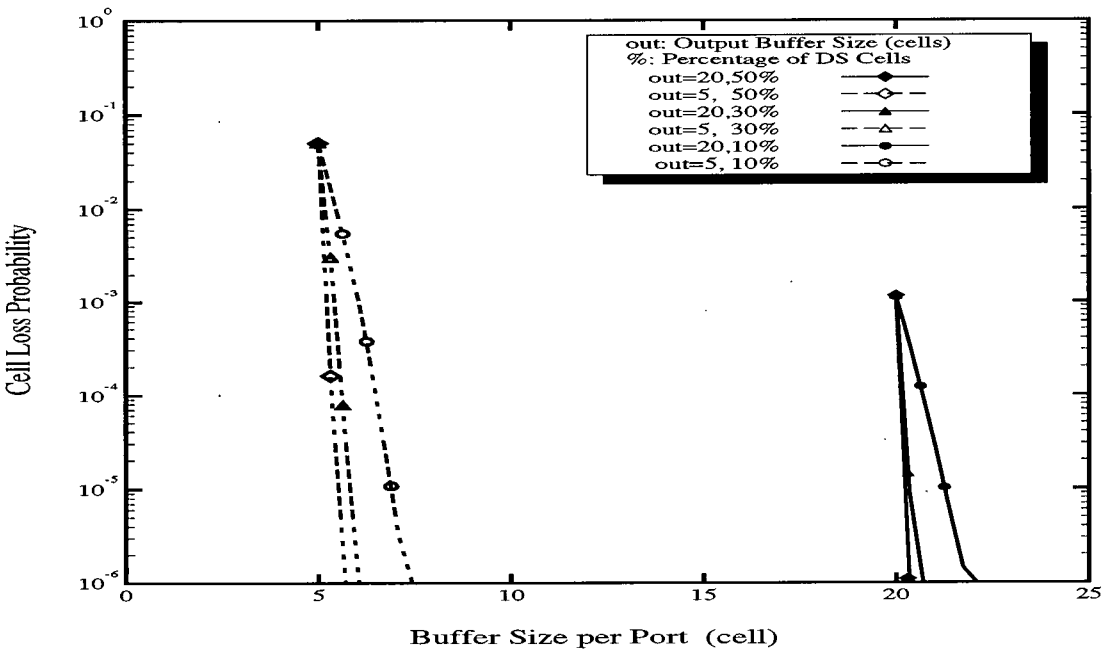

Figure 5.4 Cell loss rate of loss-sensitive (LS) cells under Policy 1 (16 x 16 switch, load $p = 0.9$)

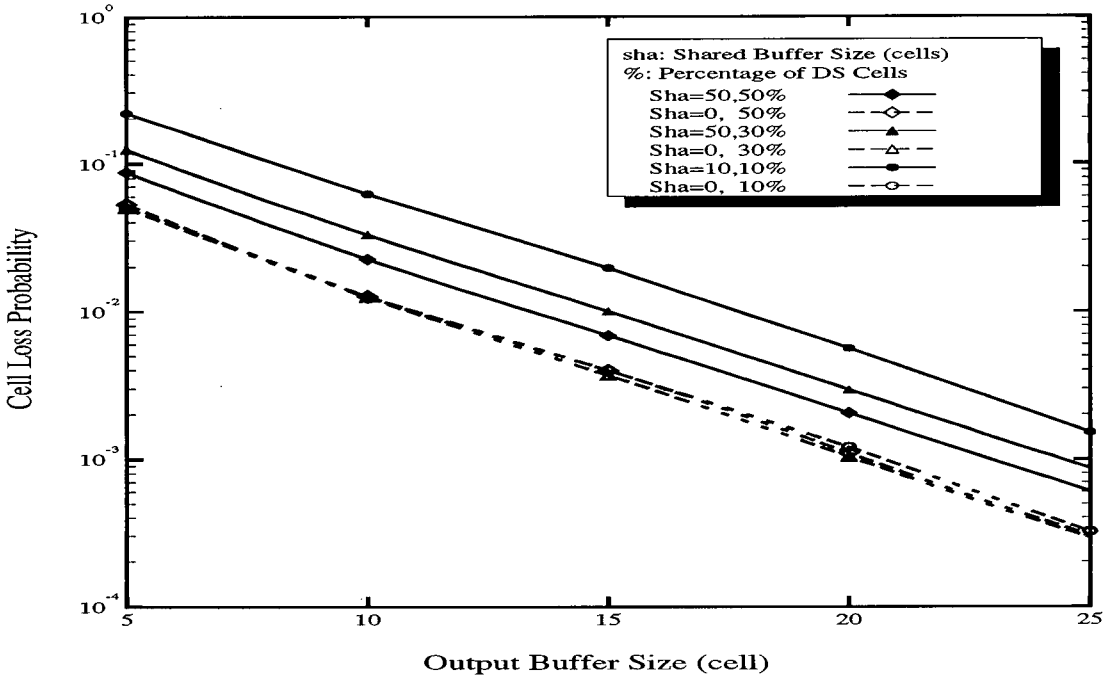

Figure 5.5 Cell loss rate of delay-sensitive (DS) cells under Policy 1 (16 x 16 switch, load p = 0.9)

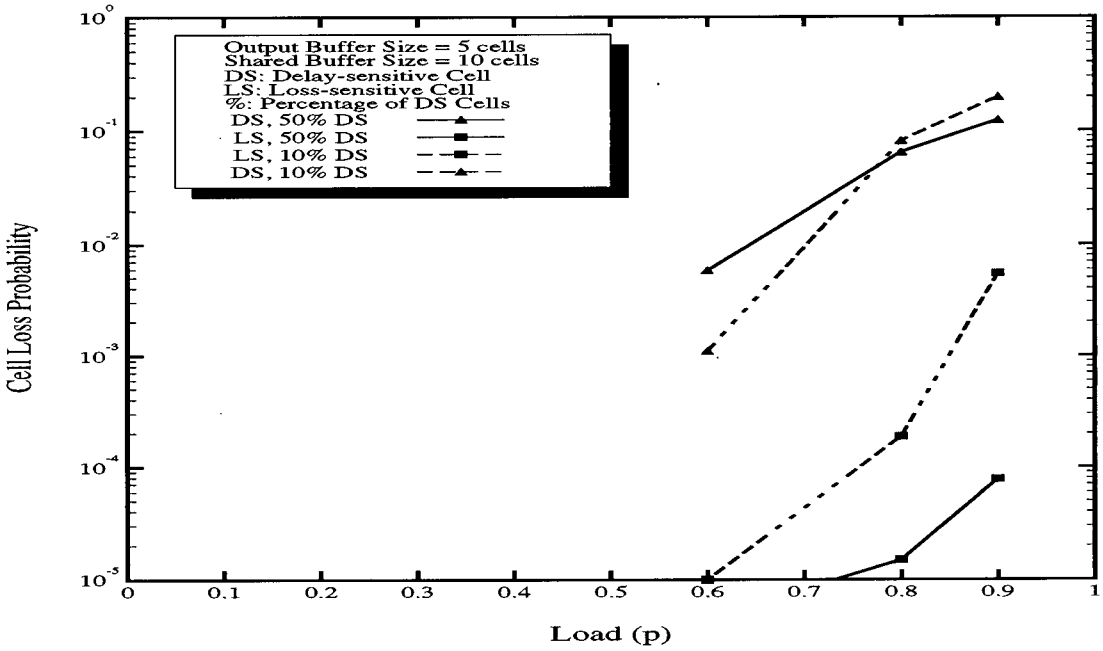

Figure 5.6 Cell loss rate comparison of two-class traffic under Policy 1 (16 x 16 switch, load p = 0.9)

5.2.2 Two-class Traffic with Policy 2

Figure 5.7 to 5.12 show the performance measures of cell loss probability and cell delay jitter for both types of traffic with Policy 2 for a hybrid switch with 16 ports, $p = 0.9$ loading, with 10-50% of that load dedicated to the DS classes. Figure 5.7, indicates the delay jitter of LS cells with a shared-buffer size of 10 cells (overall) and an output buffer size of 5 and 20 cells (per port) respectively. When the output buffer size 5 cells, probability of a delay jitter of 10 slots (with 90% LS traffic) is 3.487874e-02; and when the output buffer size is 20, this probability is 1.317911e-01. Note that as the DOB size grows, delay jitter increases because more cells can enter the output

buffer. When the percentage of LS cells decreases, their delay jitter increases because the output buffer gives higher priority to a larger number of DS cells.

Figure 5.8 shows the DS cell delay jitter associated with a DOB size of 5 and 20 cells (per port). With an output buffer size of 5 cells and 50% DS cells, the probability of delay jitter exceeding 3 cell slots is 4.868233e-03. With an DOB size of 20 cells, the probability is 1.535697e-02. Note that as size of the output buffer decreases, delay jitter is reduced for DS cells because fewer cells stay in the output buffer. A higher percentage of DS cells increases the jitter because a DS cell competes with more DS cells in its output buffer.

Figure 5.9 reveals the behavior of delay jitters for both traffic classes as a function of the load, for an DOB size of 5 cells (per port) and a shared-buffer size of 10 cells (overall). Note that the LS cell delay jitter is much larger than that for DS cells. The jitter difference between the two traffic classes is larger when the percentage of DS cells is smaller, but with equal loading from DS and LS cells, the difference in jitter diminishes as the overall load is increased.

Figure 5.10 shows that the cell loss probability for LS traffic decreases rapidly when the overall buffer size per port increases. When the DOB size is 5 cells, and the overall buffer size per port is 5 cells, (i.e., shared-buffer size is zero), the cell loss probability is 5.019813e-02. By increasing the overall buffer size per port from 5 cells to 6.25 cells, (i.e., the shared-buffer size is 20 cells), the LS cell loss probability (with 90% LS cells) goes down to 3.309488e-03. The percentage of LS cells will change slightly the cell loss probability. As the percentage of LS cell rises, the cell loss probability increases, since more LS cells try to enter the shared-and output buffer.

Figure 5.11 plots the cell loss ratio against the output buffer size for DS traffic. Because DS cells can not access the shared-buffer, the shared-buffer size does not strongly affect this measure except when it is size zero. With 10% DS cells, a shared-buffer size of 10 cells, and an output buffer size of 5 to 10 cells, the DS cell loss probability is 3.471906e-02. With no shared-buffer size 0 cell (completely dedicated output buffer), the DS cell loss decreases to 1.280243e-02. The DS cell loss is smaller with no shared-buffer because there are no LS cells coming from shared-buffer.

Figure 5.12 shows a profile of cell loss probability versus input load for both traffic with output buffer size of 5 cells (per port) and a shared-buffer size of 10 cells (overall). The cell loss probabilities generally increase with the rising load. However, cell loss for LS cells is much smaller than that of DS cells. For DS traffic, the cell loss probability decreases dramatically as the proportion of DS cell increases in the traffic mix, mainly because DS cells have higher priority access to output buffer and there are less LS cells entering the switch. Also, the loss probability of LS cells will decreases as their percentage in the traffic mix increases.

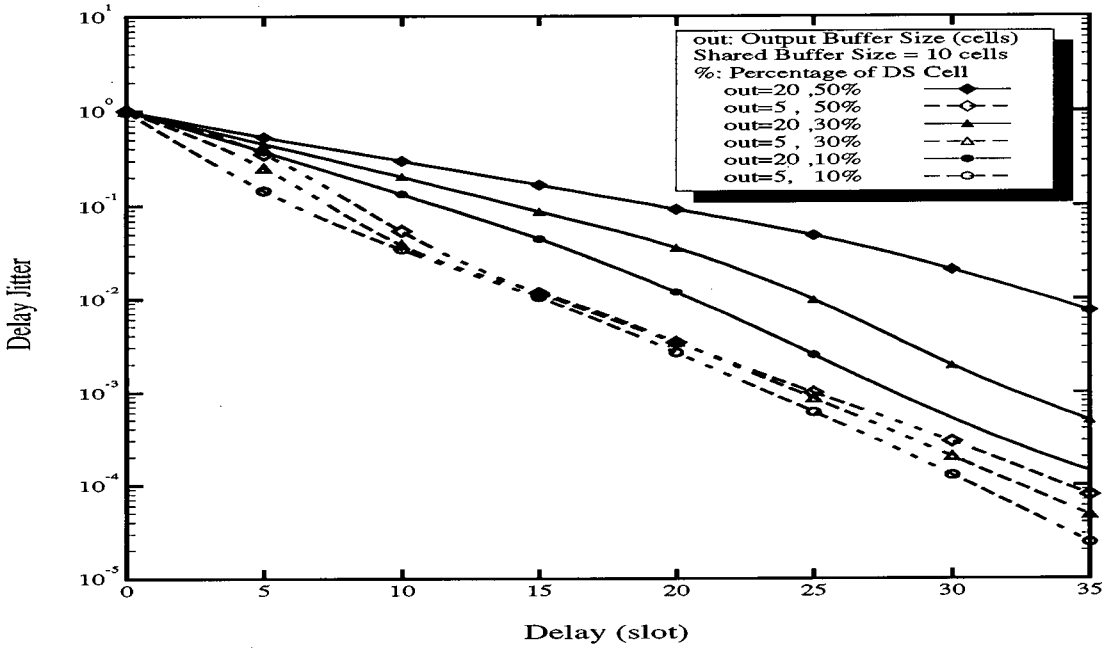

Figure 5.7 Delay jitter of loss-sensitive (LS) cells under Policy 2 (16 x 16 switch, load $p = 0.9$)

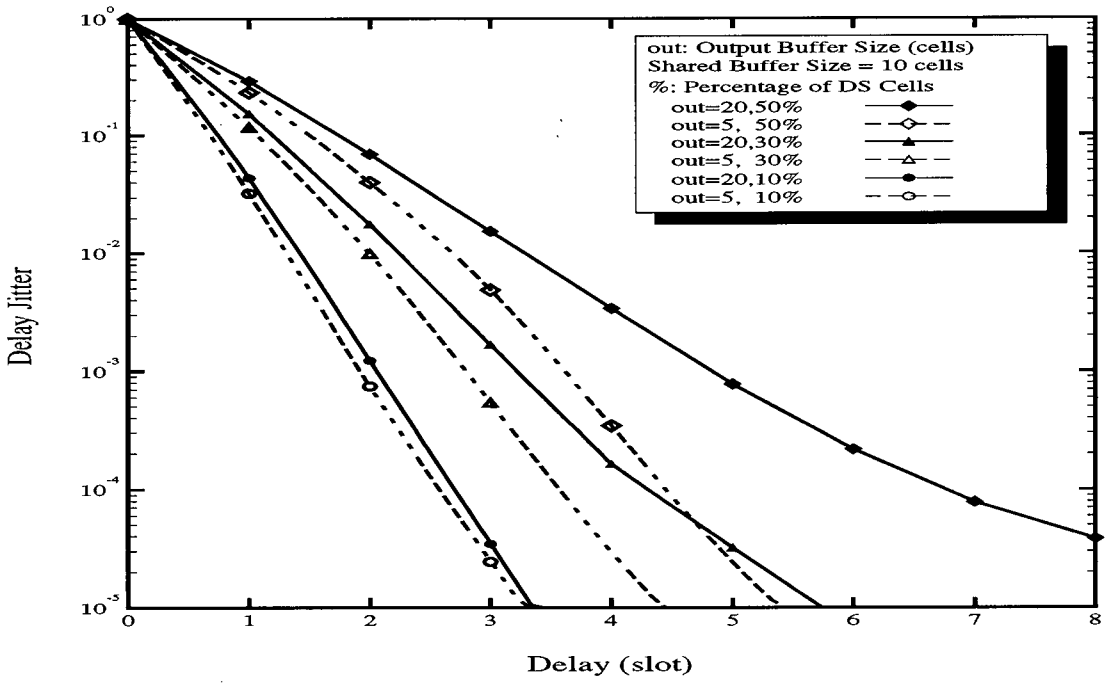

Figure 5.8 Delay jitter of delay-sensitive (DS) cells under Policy 2 (16 x 16 switch, load $p = 0.9$)

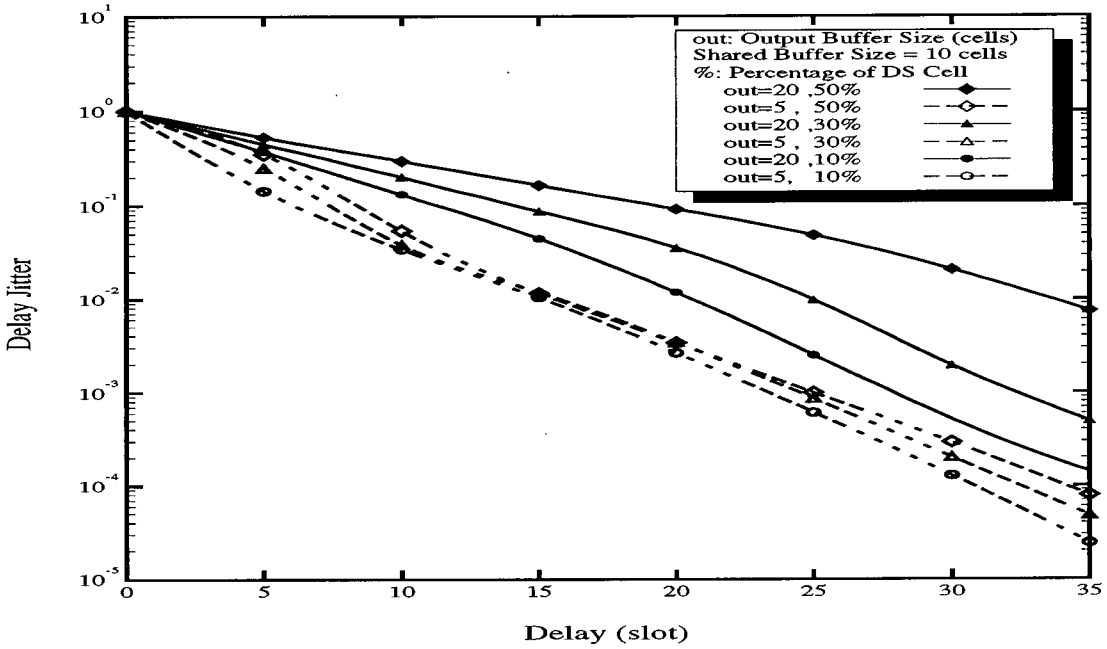

Figure 5.7 Delay jitter of loss-sensitive (LS) cells under Policy 2 (16 x 16 switch, load $p = 0.9$))

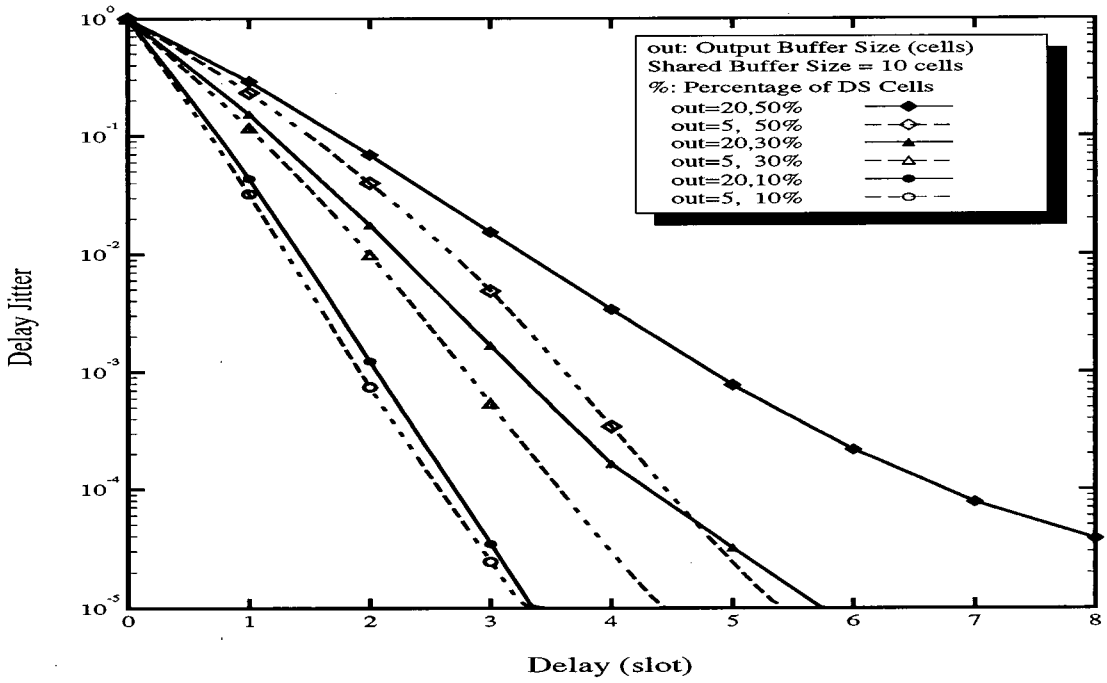

Figure 5.8 delay jitter of delay-sensitive (DS) cells under Policy 2 (16 x 16 switch, load $p = 0.9$)

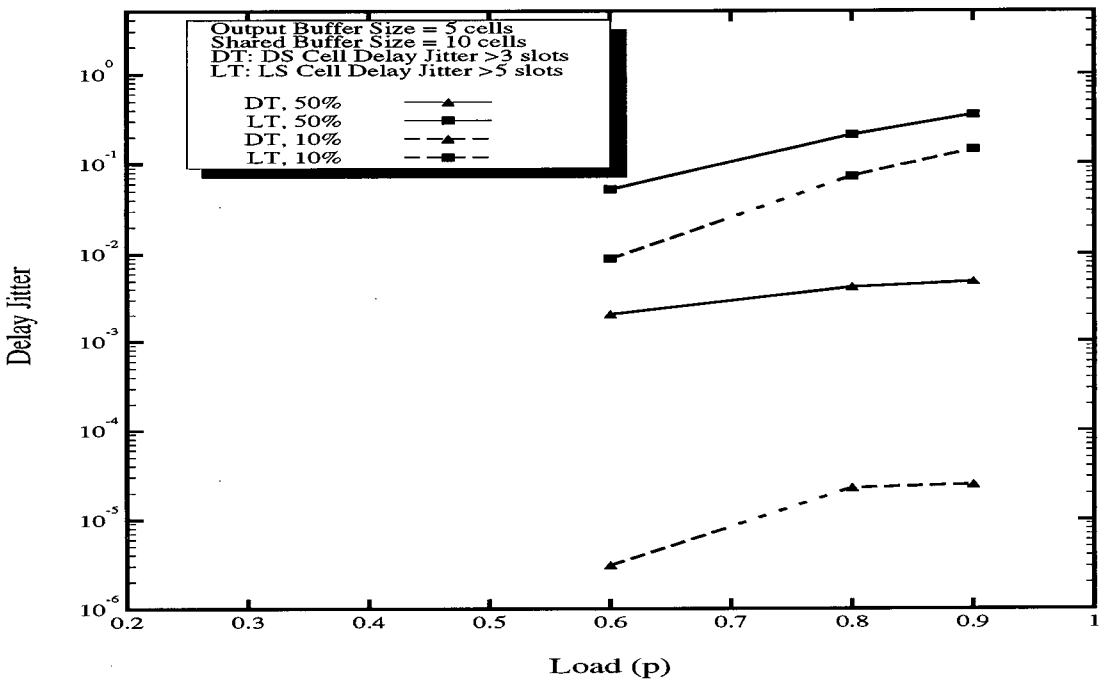

Figure 5.9 Delay jitter comparison of two-class traffic under Policy 2 (16 x 16 switch, load p = 0.9)

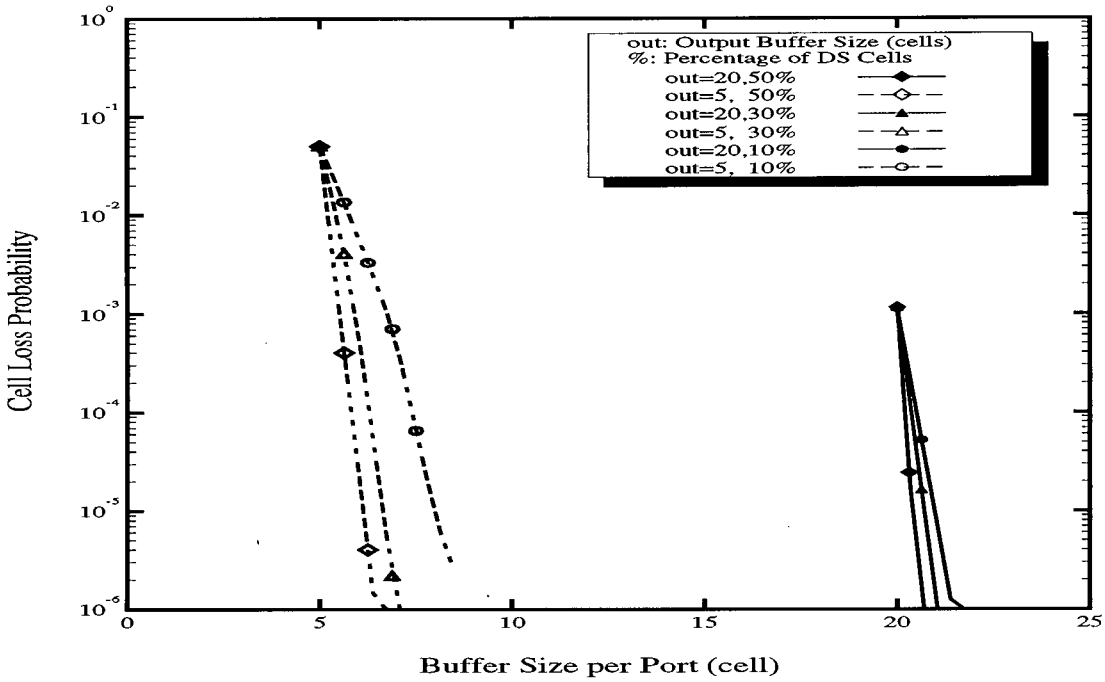

Figure 5.10 Cell loss rate of loss-sensitive (LS) cells under Policy 2 (16 x 16 switch, load p = 0.9)

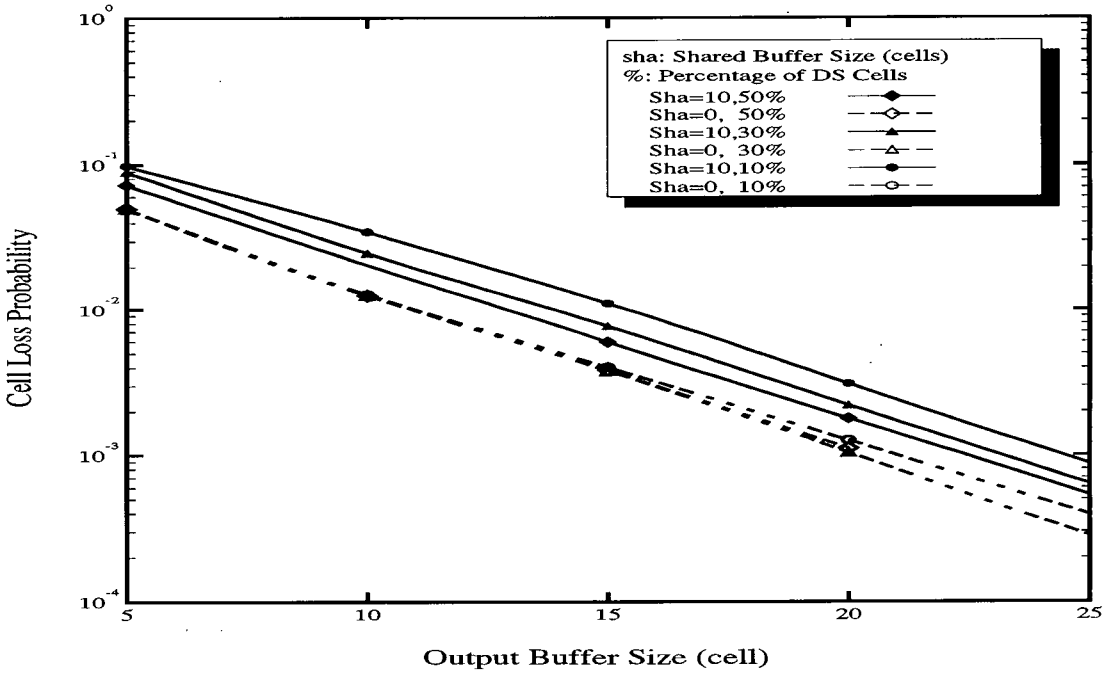

Figure 5.11 Cell loss rate of delay-sensitive (DS) cells under Policy 2 (16 x 16 switch, load p = 0.9)

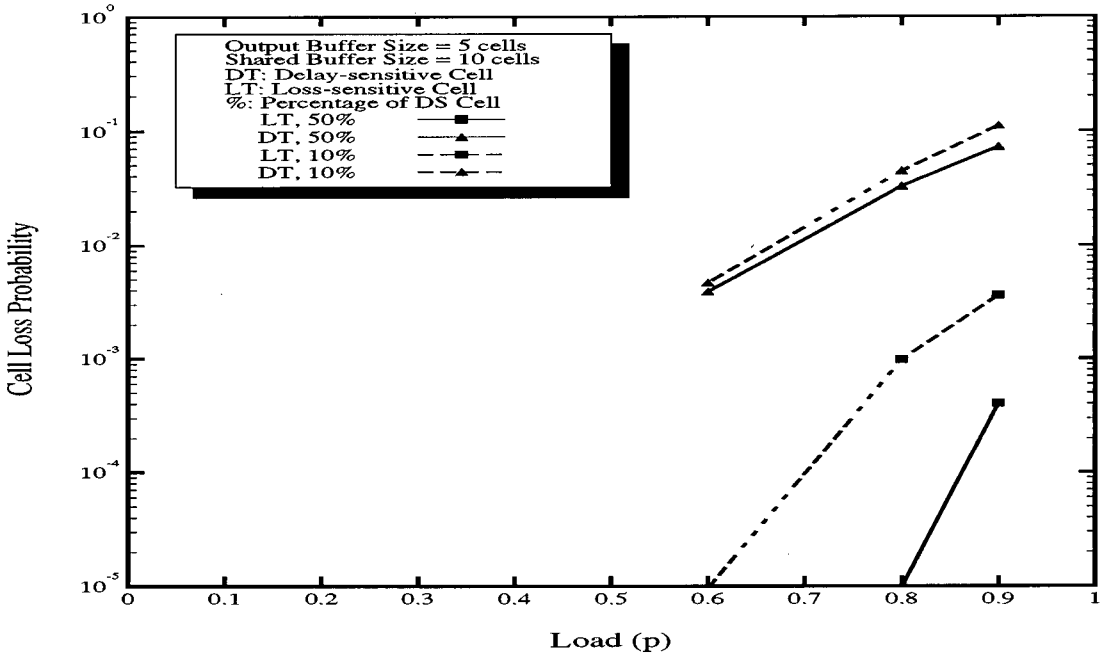

Figure 5.12 Cell loss rate comparison of two-class traffic under Policy 2 (16 x 16 switch, load p = 0.9)

5.2.3 Comparison between Policy 1 and Policy 2

Figures 5.13-5.16 compare the performance between the two management policies. Figure 5.13 shows the delay jitter performance for LS traffic under the two policies, with 10% and 50% DS relative load. Also, an output buffer of size 20 cells (per port) and a shared-buffer of size 10 cells (overall) are assumed. It can be observed that the delay jitter under Policy 2 is larger than that of Policy 1, because Policy 2 gives higher priority to DS cells, hence an output buffer, especially under higher DS load, spends more time servicing DS cells than LS cells.

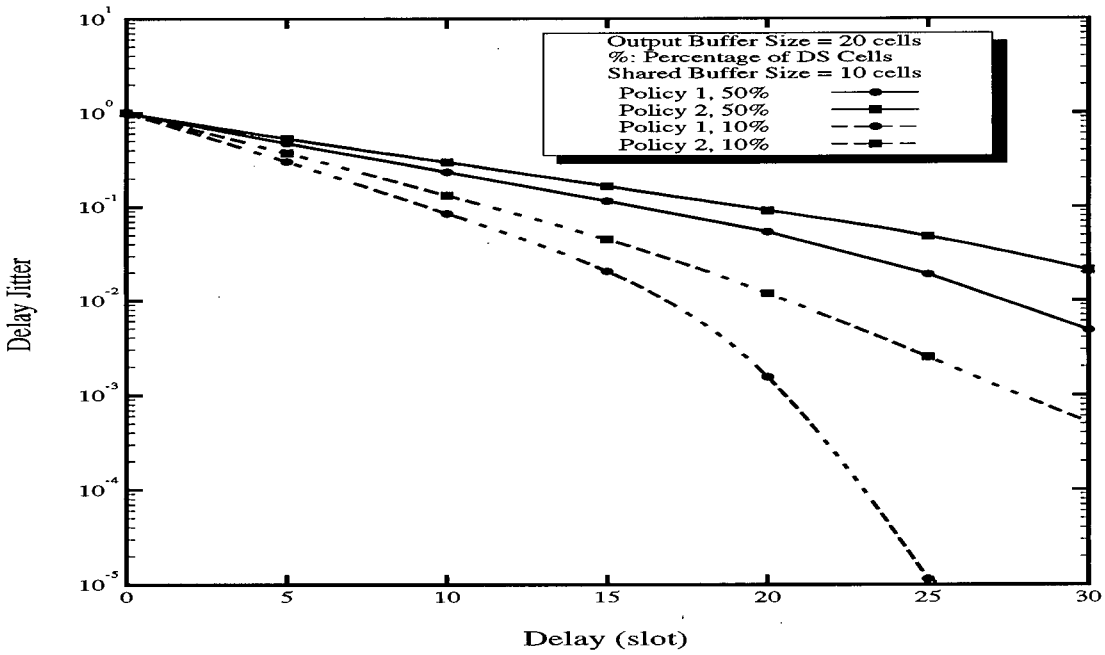

Figure 5.13 LS cell delay jitter comparison under two policies (16 x 16 switch, load $p = 0.9$)

Figure 5.14 indicates the delay jitter for DS traffic with under the two policies, using an output buffer of size 20 cells (per port), shared-buffer of size 10 cells (overall), and 10% and 50%

DS traffic relative load. With 10% of the load coming from DS traffic, the Policy 2 causes slightly higher delay jitter than Policy 1, but with 50% of the load coming from DS traffic, the delay jitter is the same under both policies. Figure 5.15 presents a similar comparison as Figure 5.13 but for cell loss under two policies. With a larger LS traffic load, the cell loss rate is larger for LS traffic. In general, Policy 1 has lower cell loss rate for LS traffic than Policy 2. This is due to the fact that LS cells can enter both shared-buffer and output buffers. Figure 5.16 reveals the DS cell loss rate for both policies with 10% and 50% of the load, coming from DS cell, and shared-buffer size 10 cells. Clearly, Policy 2 results in lower cell loss rate for DS traffic, especially for higher DS load.

In conclusion, the two policies exhibit close cell delay jitter performance for DS cells, mainly because only output buffers hold DS cells and the DS queue length never exceeds the output buffer size. However, the two policies perform differently with respect to the delay jitter of LS cells. The LS cell delay jitter under Policy 1 is smaller than under Policy 2 regardless of the traffic load mix. This is due to the fact that Policy 1 favors LS cells. DS cell loss rate of Policy 2 is smaller than under that of Policy 1 because Policy 2 favors DS cells into the output buffers. The LS cell loss rate of Policy 1 is smaller than that of Policy 2 because Policy 1 gives LS cells higher priority in accessing output buffers.

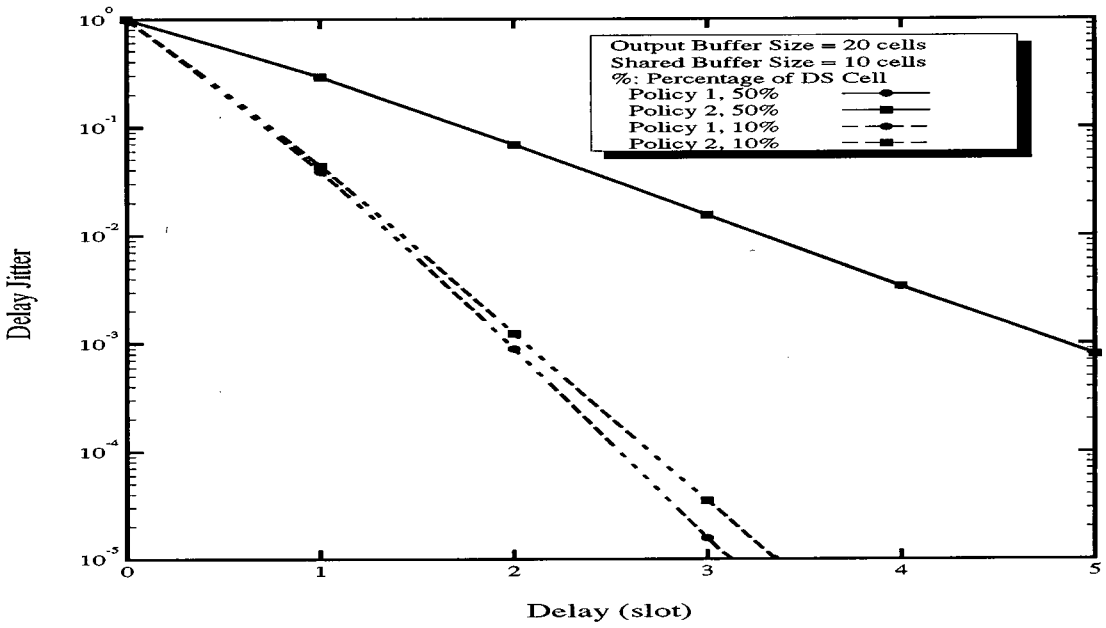

Figure 5.14 DS delay jitter comparison under two policies (16 x 16 switch, load $p = 0.9$)

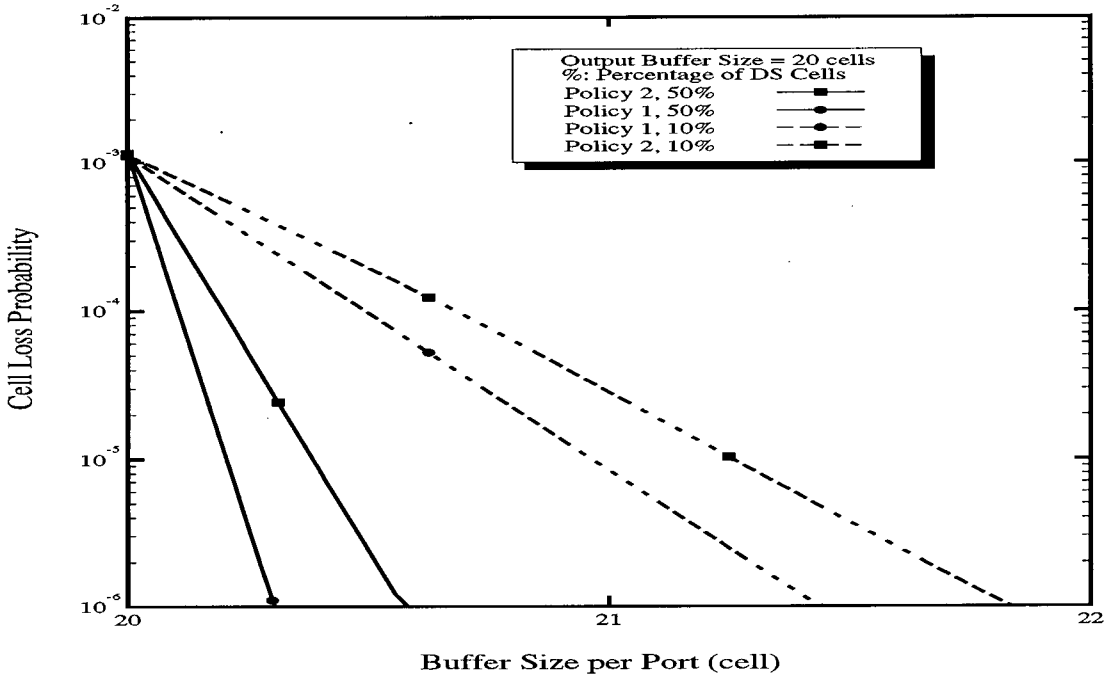

Figure 5.15 LS cell loss rate comparison under two policies (16 x 16 switch, load $p = 0.9$)

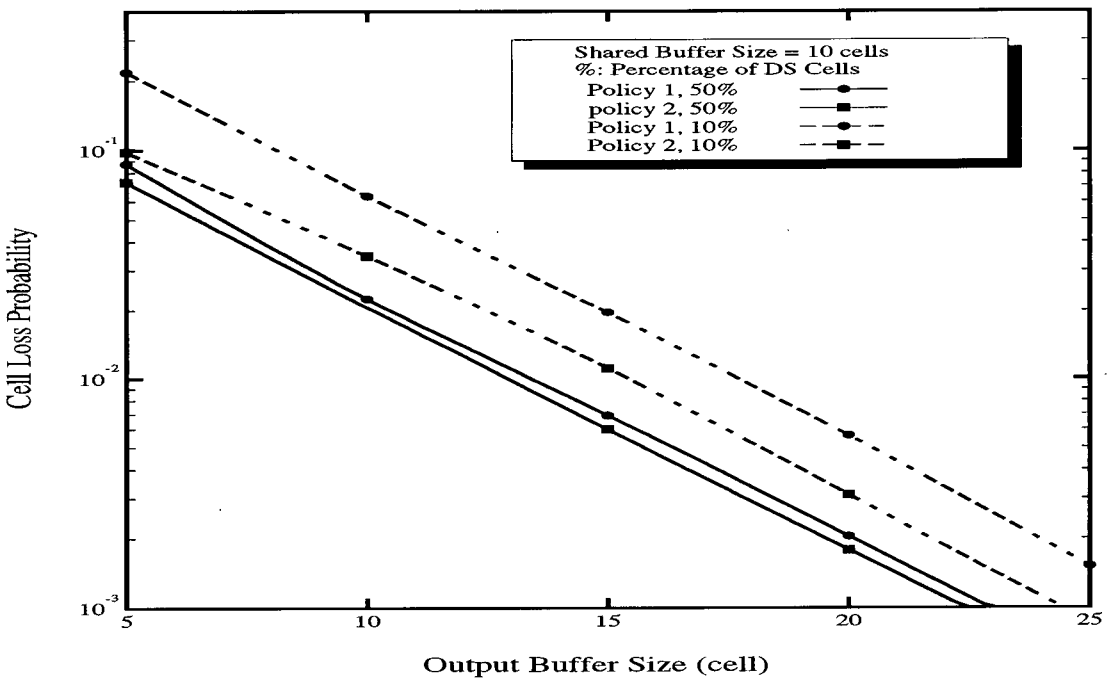

Figure 5.16 DS cell loss rate comparison under two policies (16 x 16 switch, load $p = 0.9$)

Chapter 6 Conclusion

6.1 Conclusion Remark

We have proposed a hybrid buffering ATM switch architecture, and developed a framework for the efficient analysis of the switch under independent uniform traffic with FIFO queues. We also developed a simulation package to evaluate the performance of two hybrid distributed scheduling policies for buffer management under two-class traffic models. The proposed switch combines the features of output-buffered and completely-shared-buffer switches with the purpose of improving fairness, and increasing throughput, speed and meeting the QoS requirement of each traffic class. The distributed buffer control feature can potentially support large size switches with a shared-buffer. This architecture is the first to investigate the new concepts of "queue tail management", "distributed buffer control" and combine it with "priority scheduling."

A queueing model has been developed to analyze the switch performance. The analytical approach assumes uniform independent cell arrivals. The computational complexity of the analysis requires to search only for the roots of a non-linear complex equation where the equation is of semi-closed form with a number of variables. The computational complexity of this approach is much smaller than the models used in [12]-[14].

In the queueing model, the buffer size is assumed to be infinite and all incoming cells are buffered and routed eventually (i.e. no cells are dropped from the buffer). The the probability of

queue length larger than a given value (e.g, buffer size) is an approximation of the cell loss rate of the finite size buffer. The queueing model is verified by simulation. In the simulation, however, the buffer size is finite and cells will be dropped when the buffer is full. Therefore, the queue length in the simulation is shorter than in the analytical model and, consequently, delay jitter and cell loss rate in simulation are smaller than in that in analysis. Thus the queueing model provides an upper-bound on the switch performance. The delay model can be applied to large size switches with small error, but the cell-loss model can only be applied to small size switches.

We have proposed two buffer management models with distributed cell scheduling policies for routing two-priority traffic through the proposed switch. The switch can provide very small cell delay jitter for delay-sensitive traffic and small cell loss rate for loss-sensitive traffic. The proposed buffer management and priority scheduling schemes are simple. However, the two policies perform differently with respect to the delay jitter of LS cells. The LS cell delay jitter under Policy 1 is smaller than under Policy 2 regardless of the traffic load mix. This is due to the fact that Policy 1 favors LS cells. DS cell loss rate of Policy 2 is smaller than under that of Policy 1 because Policy 2 favors DS cells into the output buffers. The LS cell loss rate of Policy 1 is smaller than that of Policy 2 because Policy 1 gives LS cells a higher priority in accessing output buffers.

The delay of two-class traffic and the cell loss rate of DS traffic in the hybrid switch and the output buffered switch are similar in [15] but the non-real time cell loss rate in the hybrid switch is 10% lower than that in the output buffered switch. The performance of the proposed hybrid switch can be improved by implementing more effective management policies in the shared-buffer. Also, the switch can be modified to handle more than two priority classes by using

more complex scheduling techniques in the output buffers, as well as more complex buffer management and priority routing in the shared-buffer.

6.2 Further Research Work

1. Investigating the scalability of switch architecture. One possible solution is to use multi copies of the hybrid buffering switch and interconnection them to construct large switches.

2. Investigating the performance of the switch under bursty, multi-priority, and unbalanced (or hot-spot) traffic.

3. Evaluating performance of the switch under different scheduling policies when the input traffic mix includes multicast traffic.

4. Improving the finite size queueing model for the switch.

5. Investigating the feasibility and cost of hardware implementation of the switch..

References

[1] M.De Prycker, Asynchronous Transfer mode: Solution for broadband ISDN, Chichester, England: Eliss Horwood, 1991

[2] F. Tobagi, "Fast cell Switch Architecture for Broadband Integrated Services Digital Network", Proc. of IEEE vol. 78, Jan 1990, pp. 133- 167

[3] M.J. Karol and M.G. Hluchyi, "Input versus Output Queueing on a Space-division cell Switch", IEEE Trans. commun., vol. COM-35, pp 1347-1356, Dec., 1987

[4] M.G. Hluchyi and M.J. Karol, "Queueing in high performance cell Switching", IEEE J. Select. Area in Commun, vol 6, Dec, 1988, pp 1587-1597

[5] S. Bhagwat, D. Tipper, K. Balakrishnan and A. Mahapatra, "Comparative evaluation of output buffer Management Schemes in ATM networks", ICC' 94 Conf. Rec., New Orleans, LO, May 1994, pp 1174-1178

[6] J.W. Causey, H.S. Kim, "Comparison of Buffer allocation schemes in ATM Switch: complete sharing, Partial Sharing, and Dedicated Sharing", ICC `94 Conf.Rec, New Orleans, LO, May 1994, pp 1164-1168

[7] M. Irland, "Buffer Managenet in a cell Switch",IEEE Trans. Commum, vol. COM-26, no. 3, pp. 328-337, March 1978.

[8] F. Kamoun and L. Kleinrock, "Analysis of shared-Buffer Finite Storage in a Computer Network Node Enviorment Under General Traffic Schemes", IEEE Trans. Commum, vol. COM-28, no. 7, pp 992-1003, July 1980.

[9] G. Latouche, "Exponential Servers Sharing a Finite Storage", IEEE Trans. Commum, vol. COM-28, no. 6, pp 910-915. June 1980.

[10] T.P. Yum and C. Dou, "Buffer Allocation Strategies with Blocking Requiremnets", Performance Evaluation, vol.4, pp 285-295, 1984

[11] A.B. Bondi, "An Analysis of finite Capacity Queues with Priority Scheduling and Common or Reserved Waiting Areas", Computers and Operations Research, vol. 16, no.3, pp. 217-233, 1989

[12] J. S. Turner, "queueing Analysis of Buffered Switching Networks", IEEE Trans. Commun, vol. 41, no. 2, pp. 412-420, Feb. 1993

[13] A. Monterosso and A. Pattavina, "Performance Analysis of Multistage Interconnection Networks with shared-Buffer Switching Elements for ATM Switching", Proceedings of IEEE INFOCOM 92, pp. 124-131, May 1992

[14] G. Bianchi and J.S. Turner, "Improved Queueing Analysis of shared-Buffer Switching Netowrks, IEEE/ACM Trans. on Networking, vol. 1, no.4, pp. 484-490, August 1993

[15] T.Y. Huang, J. L.C. Wu, "Performance Analysis of ATM Switches Using Priority Schemes", IEE Proc.-Commun., Vol. 141, No.4, august, 1994

[16] T. Kozaki,Y. Sakurai, O. Matsubara, M. Mizukami, M. Uchida, Y. Sato, and K Asano, "32 x 32 Shared-Buffer Type ATM Switch VLSI for B-ISDN", ICC' 91 Conf.,Rec. pp 711-715

[17] T. Kozaki et al., "32 x 32 shared-buffer type ATM switch VLSI's for B-ISDN's", IEEE J. Select. Area in Commun., vol 9, Oct. 1991, pp1238-1247

[18] S. Kumar and D.P. Agrawal, "A Shared-Buffer Direct-Access (SBDA) Switch Architecture for ATM based Networks", ICC' 94 Conf. Rec., New Orleans, LO, May 1994, pp 101-105

[19] K.J. Schultz and P.G. Gulak, "CAM-Based Single-chip shared-Buffer ATM Switch", ICC' 94 Conf. Rec., New Orleans, LO, May 1994, pp 1190-1195

[20] Kai Y Eng, Mark J. Karol, and Richard D. Gitlin, "Memory- and Channel sharing Techniques for congestion Control in ATM Networks", INFORCOM'93, Conf. Rec.,

[21] H. Yamanaka, etc, "622 Mb/s 8 x 8 shared-multibuffer Switch with Hierarchial Queueing and Multicast Functions", GLOBCOM' 93 Conf. Rec., Houston, TX, Nov./Dec. 1993, pp 1488-1495

[22] Abhijit K. Choudhury and Ellen L. Hahne, "Space Priority Management in a shared-Memory ATM Switch", GLOBCOM' 93, Conf. Rec., Vol.3, (Houston, Texas), Dec. 1993, pp 269-275

[23] Abhijit K. Choudhury and Ellen L. Hahne,"Buffer management in a Hierarchical shared-Memory Switch", INFORCOM' 94, Conf. Rec.

[24] Joan Garcic-Haro and Andrezej Jajszczyk, "ATM shared-Memory Switching Architectures", IEEE Networks, July/August, 1994, pp 18-26

[25] H. Kuwahara et al., "A shared-buffer memory switch for ATM exchange", ICC `89 Conf., Rec., Boston, MA, June 1989, pp 118-122

[26] Kai Y Eng, Mark J. Karol, and Y-S Yeh, "A growable packet (ATM) switch architecture: Design principles and application", IEEE Trans commun., vol 40, Feb. 1992, pp 423-430

[27] W. Fischer et al., "A scalable ATM switching system architecture", IEEE J. Select. Area in Commun., vol 9, Oct 1991, pp 1239-1307

[28] A. Jajszczyk and W. Kabacisinski, "A growable shared-buffer based ATM switch Fabric", in Globecom `93 conf. Rec., Houston, TX, Nov,/Dec., 1993 pp 29-33

[29] S. X. Wei and V. P. Kumar, "On the multiple shared-memory module approach to ATM Switching," in Proc. INFORCOM' 92, Florence, Italy, 1992, pp116-122

[30] M. D'Ambrosio, R. Melen, "Evaluating the Limited Behavior of the ATM Traffic Within a Network", *IEEE/ACM Trans. on Network*, vol.3, No 6, Dec., 1995, pp832-841

[31] B. Steyaert, H. Bruneel, "Delay characteristics of discret-time multiserver queues: an analytical approach, ICC 91.

[32] N.D. Georganas, "Buffer behavior with possion arrival and bulk geometric service", IEEE Trans. Comm., vol COM-24 (1976), pp. 938-940

[33] H.Bruneel, "Buffer with stochastic output interruptions", Electron. Lett, vol. 19 (1983), pp 735-737

[34] Y. He, H.Bruneel, "An exact analysis of the delay performance of ATM switch with input buffer", Technical report, Lab of Communication, Gent University, Belgium, 1993

[35] Yue He, H. Alnuweiri, M. Ito, "Queue-tail ATM Switch", *Technical Report*, Dept. of Elec. Eng., Univ. of British Columbia, Vancouver, Canada, 1995

[36] C. Lfelhocz, B. Lyles, C. Shenker, and L. Zhang, "Congestion Control for Best Effort Service: Why We Need a New Paradigm", *IEEE Network*, Vol. 10, No. 1, 1996, pp. 10-19.

[37] F. Bonomi, K.W. Fendick, "The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service", *IEEE Network*, March/April 1995, pp 25-39.

[38] A. K. Choudhury and E. L. Hahne, "Dynamic Queue Length Thresholds in a shared-Memory ATM Switch", *Proc. Infocom '96 Conference*, San Francisco, CA, 1996, pp. 679-687.

[39] Yue He, H. Alnuweiri, M. Ito, "Enhancing ATM Switch Design with shared-Queue Tail Memory ", *HiNet'96 Workshop, International Parallel Processing Symp'96*, Hawaii, USA, April, 1996

Appendix A.

As [25] we consider an integer-valued non-negative random variable g with a rational probability generating function $G(z)$,

$$G(z) = \frac{T(z)}{N(z)}$$

where $T(z)$ and $N(z)$ are polynomials of degree T and N, respectively. $G(z)$ has N complex poles, poles of which are the zeros of $N(z)$ $N(z)$ since $G(z)$ is analytical within the complex unit circle, its poles have an absolute value greater than 1. Unless indicated otherwise, it will assume that each pole has a distinct modulus unless it has a complex conjugate. Using a partial fraction expansion, the complex rational function $G(z)$ can be expeessed uniquly as

$$G(z) = \frac{T(z)}{N(z)} = \sum_{i=0}^{k} a_i z^i + \sum_{j=1}^{N} \frac{b_j}{z - z_j}$$

where the a_i and b_j are given constants. The first sum is the results of the division of the polynomial $T(z)$ by $N(z)$ or the case that $T \geq N$; we then have $k = T - N$. The second sum, is a weighted sum of N probability generating functions for geometrically distributed random variables, is obtained by comuting the partial fraction expansion of the remainder of the division. Using the residue theorem, one can prove that $b_j = \frac{T(z_j)}{N'(z_j)}$

Applying the inverse z-transform to $G(z)$, we obtain

$$\text{Prob}[g = n] = a_n + \sum_{j=1}^{N} -\left(\frac{b_j}{z_j}\right) z_j^{-n}$$

where $a_n = 0$ when $n \geq k$. It is clear that for sufficiently large values of n (or least $n > k$), the distribution of g is dominated by the contribution of the pole of $G(z)$ with the smallest absolute value. From the above assumptions, we can deduce that, this pole is real and strictly positive. Indeed, if the denominator $N(z)$ has two complex conjugate zeros or one negative real zero with the smallest modulus, this will lead to a number of negative quantities $\text{Prob}\{g = n\}$ for sufficiently large values of n. Now we can derive the following approximation for the tail distribution of the random variable g,

$$\text{Prob}[g = n] \cong -\left(\frac{b}{z_0}\right) z_0^{-n}$$

for sufficiently large n, where z_0 is the real pole of $G(z)$ with the smallest modulus, and

$$b = \frac{T(z_0)}{N'(z_0)}$$