5-2005

Design and performance evaluation of switching architectures for high-speed Internet

Alvaro Munoz
University of Texas-Pan American

DESIGN AND PERFORMANCE EVALUATION OF SWITCHING ARCHITECTURES FOR HIGH-SPEED INTERNET

A Thesis
by
ALVARO MUNOZ

Submitted to the Graduate School of the
University of Texas-Pan American
In partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

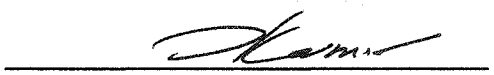
May 2005

Major Subject: Electrical Engineering

DESIGN AND PERFORMANCE EVALUATION OF SWITCHING ARCHITECTURES FOR HIGH-SPEED INTERNET

A Thesis
by
ALVARO MUNOZ

Approved as to style and content by:



Dr. Sanjeev Kumar
Chair of Committee

Dr. Heinrich Foltz
Committee Member

Dr. Zhixiang Chen
Committee Member

May 2005

ABSTRACT

Munoz, Alvaro, Design and Performance Evaluation of Switching Architectures for High-Speed Internet, Master of Science (MS), May 2005, 119 pp., 64 figures, 4 tables, references, 57 titles.

The motivation for this thesis is the desire to build faster and scalable routers that efficiently handle the exponential traffic growth in the Internet. The Internet forwards information through a mesh of routers and switches, which has to keep up with the increasing demands of traffic. Shared-memory based switches are known to provide the best throughput-delay performance for a given memory size. In this thesis performance of commonly used memory-sharing schemes for the shared memory switches are evaluated under balanced and unbalanced bursty traffic. The scalability of shared-memory switches has been a research issue for quite sometime. One approach is to employ multiple memory modules and use them in parallel to enhance the capacity. The two well-known architectures in this category are (i) shared-multibuffer (SMB) switch architecture invented by Yamanaka et al of Mitsubishi Electric Corporation, Japan; and (ii) the sliding-window (SW) switch architecture invented by Dr. Kumar of UTPA, Texas, USA. In this thesis, performance of these two architectures are evaluated and compared. Furthermore, in this thesis, the SW switch architecture is extended to enable priority switching to provide differentiated Quality of Service (QoS) for different traffic classes.

ACKNOWLEDGMENTS

This thesis has been possible with the help of many people. First and foremost, among these are my loving parents who I would like to thank for all their support to my education. I am deeply thankful for my advisor Dr. Sanjeev Kumar, for his continued support and guidance in both academic and personal matters, his help in defining the research problem, the frequent constructive discussions, his never-ending encouragement for me to excel in research, and the training in writing papers. It was a great opportunity and I was really excited to work on a new and an exceptional switching architecture invented by Dr. Kumar, namely the Sliding-Window Switch Architecture that has received global attention in this field. I thank Dr. Heinrich Foltz of Electrical Engineering and Dr. Zhixiang Chen of Computer Science for their willingness to serve as the committee members. I am also thankful for the research assistantship support and travel support from the Department of Electrical Engineering, CITeC, OBRR/NIH grant, and the Dean's office of the College of Science and Engineering at the UTPA.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

The Internet has had a tremendous success in the last several decades, evolving from a small research network to a scalable and distributed network with a rapid development and deployment of applications and services. The exponential growth of the Internet is demonstrated by the tremendous increase in both the number of users [1] and the traffic growth that has already made the Internet carry more traffic than the phone network [2][3]. Advances in transmission technologies have provided abundant transmission bandwidth. Progress in optical transmission technologies [4], such as dense wave division multiplexing (DWDM), optical add-drop multiplexers, ultralong-haul lasers, and optical amplifiers, has a large impact on lowering costs of digital transmission. The advent of DWDM in 1996 has provided doubling of the capacity of fiber optics every 7 to 8 months [5]. However, the growth rate is expected to decrease to doubling every year as we start approaching the maximum capacity per fiber of 100 Tbit/s [6]. Historically, router capacity has increased slightly faster than Moore's law, multiplying

by 2.2 every 1.5 to 2 years. This has been due to advances in router architecture and packet processing [7].

1.2 Motivation

The Internet originally provided best effort service but as the Internet becomes widely used, and more users have broadband access service such as asymmetric digital subscriber line (ADSL) and fiber to the home (FTTH), new application types uniting voice, video, and data traffic need to be delivered on the network infrastructure. Demands for systems that provide quality-of-service (QoS) to real-time applications and mission-critical financial data are increasing. The net effect of these driving forces is a set of new requirements that are placed on the routing and switching functions. These new requirements have put increasing demands on Internet routers and switches for supporting higher bandwidth, greater switching capacity, and efficient support for quality-of-service (QoS) in the network. This thesis focus on the design and evaluation of efficient Internet switching architectures that can face the challenges imposed on the next generation Internet by the scalability issues and QoS requirements of present and future Internet applications.

1.3 Packet Switching

There are two fundamental types of underlying network infrastructures based on how traffic is multiplexed and switched inside a network: circuit-switching and packet-switching. A circuit-switched network provides a fixed connection to its hosts. In circuit-switching, a guaranteed amount of bandwidth is allocated to each connection and

available to the connection all the time. The most common example of a circuit-switched network is the public-switched telephone network (PSTN). Circuit-switching can be rather inefficient. An amount of bandwidth is dedicated for the duration of the connection, even if no data are being transferred. For a voice connection, utilization may be rather high but for bursty-traffic the utilization of networks resources is not efficient. An example of bursty-traffic is Web browsing. When a user clicks on a hyperlink, a new page needs to be downloaded as soon as possible from a server. When a user is looking at a recently downloaded page, there is almost no traffic. Thus a bursty stream in Internet requires a lot of bandwidth from the network whenever it is active and very little bandwidth when it is not active.

In contrast packet switching deals with the problem of transporting bursty-traffic efficiently. In packet-switched networks (Fig. 1.1), the data stream is broken up into small packets of data. Each packet contains a portion of the user's data plus some control information. The control information includes the information that the network requires to be able to route the packet through the network and deliver it to the intended destination. Theses packets are statistically multiplexed together with packets from other data streams inside the network. Therefore statistical multiplexing improves the bandwidth utilization of the network.

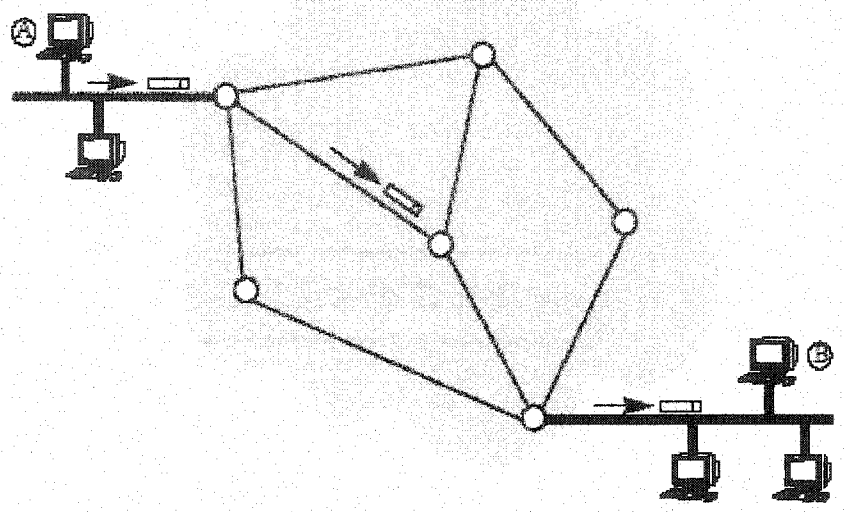


Fig. 1.1 The nodes in the network cloud can be switches or routers. As they arrive at each node, packets sent from host A to host B get switched onto a path which leads them to host B [13]

In a packet switched network, data-packets travel through a mesh of routers or switches over links towards their final destinations. The delivery of packets from host A to host B, as illustrated in Fig. 1.1, requires packets to pass through multiple routers and switches (nodes). As soon as a packet arrives at a node, the router decides to which output link the packet is to be switched, based on its IP address from packet's header, and transmits it to the corresponding output link. Contention occurs among packets for network resources when more than one packet is destined for the same output link at the same time. This is solved by queueing the packet(s) that could not access the destined output link and forwarding the packet(s) later from buffer when the output link becomes available. Buffers cause some important effects; the delay experience by packets at each node depends on how many packets are queued up ahead of it. This causes delay variations (jitter). On occasion, the traffic in some output links may be so high that it

causes the buffers to overflow. When this happens, some of the packets must be dropped from the network.

1.4 Router and Switches

A router receives traffic through its input-ports and sends it shortly afterwards through the appropriate output-ports. Information is sent either through fiber optics for the long haul and high speeds, or through copper cables for the short haul and mid-to-low speeds. Routers need to look up the destination address in a routing table to decide where to send a packet next, or in which queue it should be buffered. Packets also need to be scheduled to use the switch fabric, so that they go from the input-ports to the output-ports without contention. Conflicts are resolved by deferring the transmission of all but one of the conflicting packets until some later time when the contention has been cleared.

Fig. 1.2 shows the functional blocks of a router, also called a switch. When traffic arrives at the ingress linecard, the framing module extracts the incoming packet from the link-level frame. IP address is used to look up the routing table. A so-called longest-prefix matching method is used to find the output-port. In some applications, packets are classified based on the combined information of IP source/destination addresses, transport layer, port numbers, and type of protocol. Based on the result of classification, packets may be either discarded or handled at different priority levels. Then, any required operations on the packet's header are performed, such as decrementing the Time-To-Live (TTL) field, updating the packet checksum, and processing any IP options. After these operations, the packet is sent to the egress port through a switch fabric, which is rescheduled every time slot. Several packets destined to the same egress port could arrive

at the same time. Thus, any conflicting packets have to be queued in the input-port, the output-port, or a centralized shared-memory location.

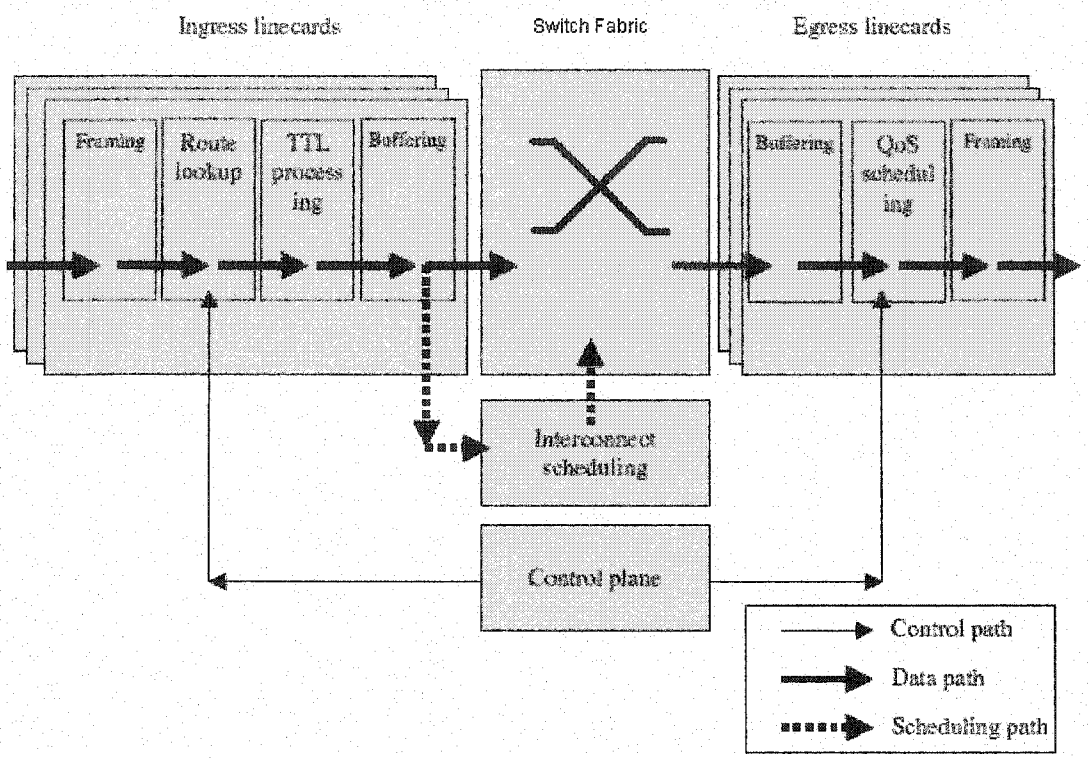


Fig. 1.2 Functionality of a router or switch [5]

In the output linecard, some routers perform additional scheduling that is used to police or shape traffic, so that quality-of-service (QoS) guarantees are not violated. Finally, the packet is placed in a link frame and sent to the next hop. In addition to the datapath, routers have a control path that includes the system configuration, management, and exchange of routing table information. These are performed relatively infrequently. The router controller exchanges the topology information with other routers and constructs a routing table based on a routing protocol.

1.5 Buffering Strategies

One of the central concerns in designing a high-capacity switch is limited memory-bandwidth [8][9]. For switches operating at high speed, the speed of the memory, which is a basic building block of queues, becomes a limiting factor. Switch speed is often limited by the rate at which the memory can operate. Depending on switch architecture, queueing can take place at different parts of the switch as shows in Fig. 1.3: at the inputs, at the outputs, at both inputs and outputs, or at a centralized location. The following subsections explain different buffering strategies and the advantages and disadvantages of theses approaches.

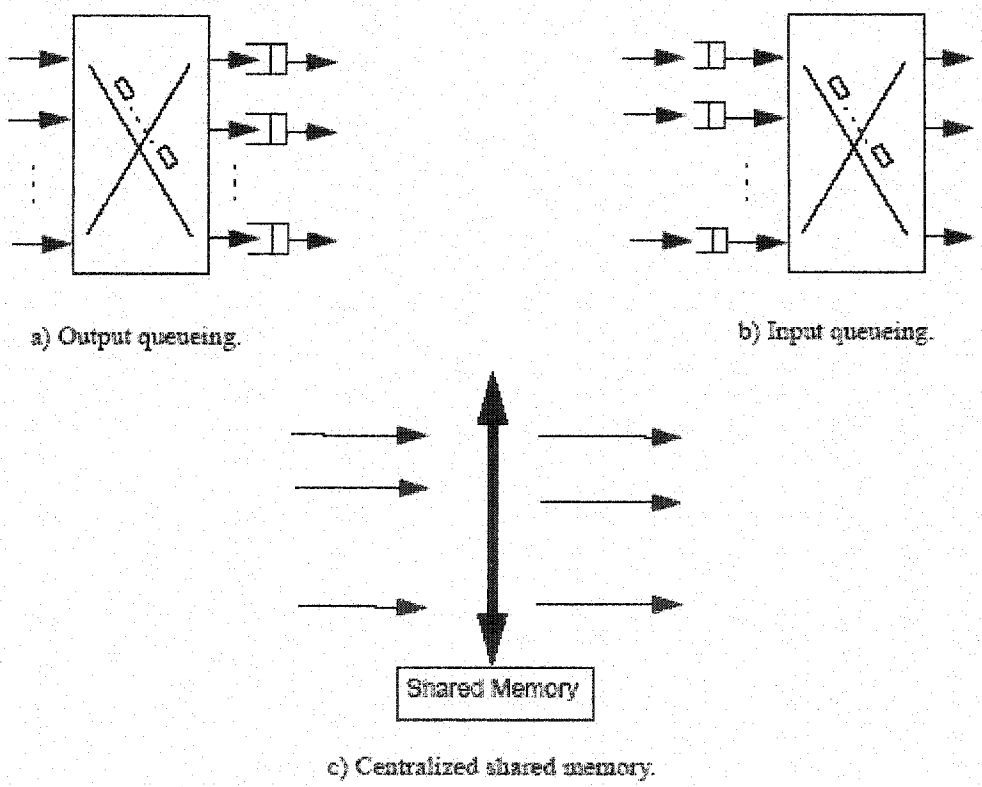


Fig. 1.3 Various buffering strategies on switching architectures [13]

1.5.1 Input Buffered Switches

Input-buffering is characterized by the memory blocks located at the input ports as shown in Fig. 1.3 (b). At most one packet can arrive at and depart from each input in one time slot. Thus, the memory is only required to operate at twice the line rate. Unfortunately input buffering suffers from head-of-line (HOL) blocking and limits the throughput to 58.6% for uniform traffic [10]. HOL blocking occurs because packets for different outputs share the same first-in first-out (FIFO) queue.

When packets for different outputs share a FIFO queue, a packet, which may be destined to a free output, can be blocked by a packet in front of it, which may be destined to a busy output-port. Such a packet is called a HOL packet. The HOL packet may be destined to a different output but has to remain in the queue because its output is busy. As a result, some inputs and outputs are unnecessarily left idle and thus degrading the throughput performance. There are various techniques to reduce HOL blocking [11][12][13]. A technique called virtual output queueing (VOQ) aims to eliminate HOL blocking [14][15] but the scheduling problem in VOQ switches limit the scalability of this switch architecture. An *NxN* size VOQ switch need to manage N^2 logical queues in all memory blocks and existing scheduling algorithms are either complex to run at high speed or only perform well under restricted conditions.

1.5.2 Output Buffered Switches

Output-buffering is referred to as a technique in which all memory blocks are placed at the outputs [16][17][18] as shown in Fig. 1.3 (a). Memory independence among

output-ports enables output-buffered switches to guarantee a minimum-bandwidth between individual connections [19]. Nonblocking fabrics help to ensure that the adequacy of service delivered to a particular user should not depend on the detailed behavior of other users. Deterministic bounds on end-to-end delay and end-to-end buffer sizing results for lossless transmission are available under output-buffering; these results can be extended to arbitrary virtual path connection (VPC) structures [20]. In addition, because output-buffering do not suffer from HOL blocking at the input-ports, they can achieve 100% throughput for each output link. On the other hand output-buffering requires that all arriving packets must be immediately delivered to their outputs. An output-buffered switch of size NxN would require in the worst-case scenario $N + 1$ write/read operations to be done in a memory block which limit the scalability of this buffering technique.

1.5.3 Input and Output Buffered Switches

Input and output buffered switches are intended to combine the advantages of input buffering and output buffering [21][22][23]. In input buffering, the input buffer speed is comparable to the input line rate. In output buffering, there are up to L ($1<L<N$) packets that each output-port can accept at each time slot. If there are more than L packets destined for the same output-port, excess packets are stored in the input buffer. Since the output buffer memory only needs to operate at L times the line rate, a large-scale switch can be achieved by using input and output buffering. However, this type of switch requires a complicated arbitration mechanism to determine which of L packets among the N HOL packets may go to the output-port.

1.5.4 Shared-Memory Switches

In shared-memory switch [24][25][26] incoming packets are time-division multiplexed into a single data stream and sequentially written to a shared-memory. Also packets are removed from shared-memory in a single output data stream and demultiplexed in several egress lines. The buffer-space is a centralized memory block (Fig. 1.3 c) used by all ports to write or read packets. Logically the buffer-space is partitioned in multiple logical queues, one for each output-port or even further for each traffic flow to provide quality-of-service (QoS). Shared-memory switches provide the best memory utilization. All input/output have access to the shared-memory and buffer space that is unused by other outputs can be employed by very active output-ports. A subject that should be taken into account in shared-memory switches is the efficient access of output-ports to the buffer-space. The problem is that a single port or group of ports can take over most of the buffer, preventing packets destined for less utilized ports from gaining access. This causes degradation in throughput performance of the switch. Optimal ways to share the buffer-space among all the input and output-ports by controlling the queue build-up inside the shared-memory switch for various traffic conditions have been discussed in [27][28][29][30][31][32][33].

A drawback, in the shared-memory switch is the speed at which the memory has to operate. Scaling a shared-memory switch to a larger size is constrained by an internal speedup requirement. For a switch of size NxN, the memory must be able to support N read accesses by all the N outputs and N write accesses by all the inputs in a time slot, i.e., the memory must operate $2 \cdot N$ times faster than the line rate. There are several

approaches to overcome traditional limitations in shared-memory switches. A method to scale a packet switch is to use multiple switches operating independently and then interconnect such switches in some fashion to build a large-scale switching system [34][35][36][37]. Parallel packet switch (PPS) architecture [34][35] is comprised of multiple identical lower speed packet switches in parallel. An incoming stream of packets is spread, packet by packet, by a demultiplexer across switches, and then recombined by a multiplexer at the output. However, PPS architecture requires coordination buffers in the multiplexers and de-multiplexers in order to employ a distributed scheduling algorithmic. Use of multistage interconnection networks (MINs) [36][37] to interconnect small-scale shared-memory switches to build large size shared-memory switch has been used. Although, it is shown in [42] that the optimal throughput performance of small-size shared-memory switches degrades significantly if the switch is grown to a larger-size switch by connecting these optimal-performance switches using MIN approach.

Shared-memory switches have been scaled using collectively multiple physically-separate memory-modules which as a group form a large memory block that it is intended to buffer packets that lost contention for switch resources. Allowing sharing of the total buffer space from the entire number of memory-modules among all the input and output-ports [38][39][40][41][42][43]. A Shared-Multibuffer (SMB) switch design, proposed in [38][39][40], provides a complete sharing of all the memory-modules among all input and output-ports. The SMB-switch deploys a centralized controller to centrally control and manage all switching functions; such as their write/read operations for all incoming and outgoing packets, update idle and used memory addresses, provide instructions to input and output spatial (interconnection) switches on how to provide routing of data

packets corresponding to the input and output lines The main disadvantage of this approach is that the use of a centralized controller can become a performance bottleneck as the switch is grown to a larger size. The sliding-window (SW) switch architecture in [41][42] is a class of switching architecture, where physically separate multiple memory-modules are logically shared among all the ports of the switch, and the control is decentralized. Memory-modules are independent and they use their local memory controllers to perform write/read operations for data packets based only on the information available locally. Decentralized switching functions enable the SW-switch to operate in a pipeline fashion to enhance scalability and switching capacity.

CHAPTER 2

MEMORY-SHARING SCHEMES

2.1 Introduction

Switching systems employing shared-memory have been known to provide the highest throughput and incur the lowest packet-loss compared to that of packet switches employing input or output buffering strategies under conditions of identical memory size and bursty-traffic. High-capacity shared-memory based switches are becoming popular due to new approaches to obtain high memory-bandwidth with the use of multiple memory-modules in parallel that efficiently emulate a big-size memory-module which is logically shared among ports. [38][41][42]. Internet traffic is inherently bursty in nature. With the increase in Internet traffic, the Internet switches have to be efficient in handling bursty-traffic. Because of their ability to achieve high throughput for a given memory resource deployed, the shared-memory switches are increasingly being used in high-performance Internet core routers and switches. For efficient sharing of a common memory space by the packets of the output-ports in a shared-memory switch, it is important to deploy some type of memory-sharing scheme. These memory-sharing

schemes have a direct impact on the throughput performance and utilization of its output-ports [29][30][31][33]. In this chapter we present different types of memory-sharing schemes that can be possible for packets of different output-ports to share a common memory space. We also measure the performance of these memory-sharing schemes under bursty-traffic conditions. The bursty-traffic can be divided into two different types (i) Balanced bursty-traffic and (ii) unbalanced bursty-traffic. The balanced bursty-traffic has bursts of incoming packets uniformly distributed to the output-ports with each output-port having equal probability of receiving a burst of packets. In the unbalanced bursty-traffic, some ports of the switch have higher probability of receiving bursts of traffic than other ports. Some examples of applications that can produce unbalanced bursty-traffic could be due to fixed geo-spatial location of videoconferencing sessions (e.g. business area of downtown), or live digital broadcast for sporting events which may be more popular with subscribers in certain geographical area. In this chapter, the switch output-ports with higher probability of receiving bursts of packets are called very active ports and the ports with lower probability of receiving bursts of packets are called lightly active ports. To the best of our knowledge, no prior work has been done to compare all four different memory-sharing schemes under balanced and unbalanced bursty-traffic. In this chapter, we measure and compare the effect of balanced and unbalanced bursty-traffic on the performance of four memory-sharing schemes for the class of shared-memory packet switches used in Internet.

2.2 Background

A shared-memory switch, as shown by Fig. 2.1, allows multiple broadband lines to share a common buffer-space for storing packets bound for various output-ports of the switch. The access to the memory is time multiplexed via a shared bus. Packets arriving on all input-ports are multiplexed into a single stream that is fed to the common memory for storage. At the same time, an output stream of packets is formed by retrieving packets from memory; the output stream is then demultiplexed, and packets are transmitted on the output-ports.

Inside the memory, packets are organized into separate logical queues. An individual logical queue per port is usually set, but there can be more if the switch supports various priority classes. A logical queue for output-port d is depicted in Fig. 2.2, incoming packets destined to output-port d causes queue length Q_d to expand and eventually packets are dropped if buffer-space is not available. A memory-sharing scheme selectively drops packets and controls the queue built-up inside memory before the buffer overflow. In effect, a memory-sharing scheme intends to provide to all the output-ports a fair access to the memory resources by controlling the use of buffer-space. Consequently, it maintains an efficient utilization of the switching system.

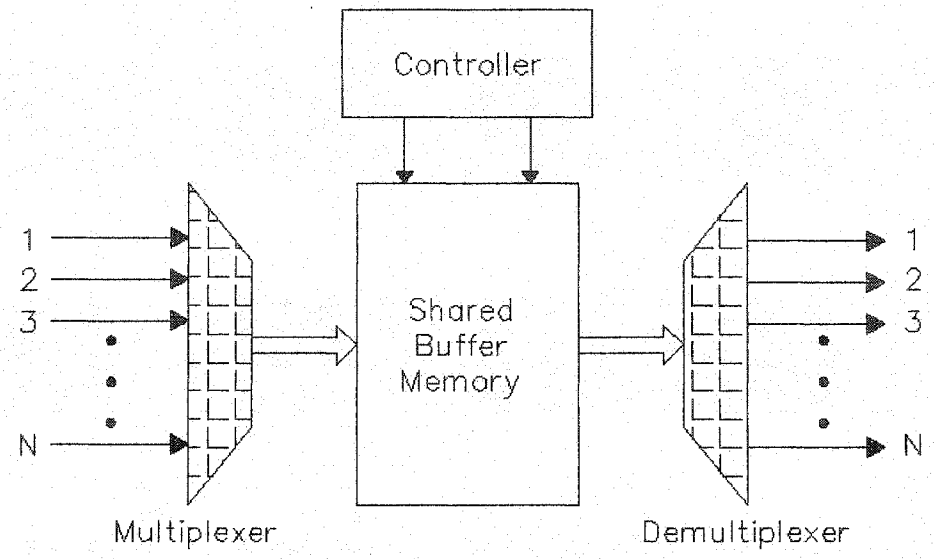


Fig. 2.1 Basic architecture of shared-memory switches

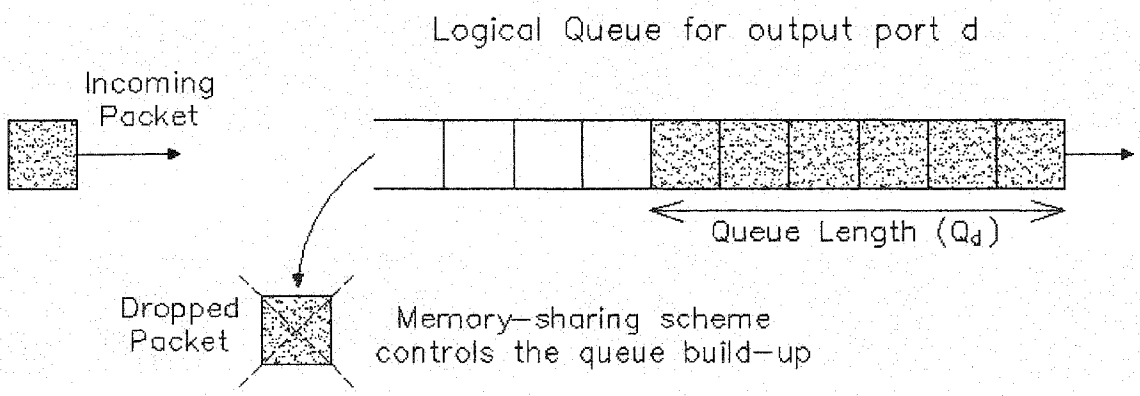


Fig. 2.2 Logical queue in shared-memory switch

It is necessary to have some kind of control on sharing of the common buffer-space among the packets for different output-ports of the switch. In the case of complete memory sharing, it is possible for packets of a given output-port or a group of output-ports (monopolizing ports) to completely occupy the common buffer-space and in effect block the passage of packets belonging to non-monopolizing ports of the switch. The

provision of complete-sharing of the common memory results in unfair use of common-buffer-space. Because of this, the complete-sharing of common-buffer-space in shared-memory packet switches can result in starvation for some output-ports of the switch and inefficient output-utilization. In order to alleviate this problem of unfairness, it is common to restrict the occupancy of the common-buffer-space in order to always allow passage to all input-output pairs. The memory-sharing schemes, namely the individual-static threshold, global-static threshold, dynamic threshold and SMDA based memory-sharing schemes are used to restrict complete occupancy of the common memory by packets of a given output-port or a set of output-ports. In this chapter we compare the impact of various memory-sharing schemes on the throughput and packet-loss performance of a switch under a given bursty-traffic.

2.3 Individual-Static Threshold Based Sharing Scheme

This is a straightforward scheme used to control, on an individual basis, the output-queue build-up inside the shared-memory switch. Under this scheme, a restriction is placed on the maximum length of the output queues [28] to a pre-determined value which is defined as the individual-static threshold value (ST). The individual-static threshold (ST) value is set to a multiple α of the total buffer-space (B)

$ST = \alpha \cdot B$ packets *where* $0 < \alpha < 1$

An individual output queue (Q_d) inside the common buffer-space is not allowed to exceed the ST value. The packets of the output-queues that exceed ST value are dropped as illustrated in Fig. 2.3

if Q_d < ST packets **then**

Accept packets for the output-port *d*

else

Drop the packets for output-port *d*

Fig. 2.3 Individual-static threshold scheme to regulate the sharing of the memory space

This scheme prevents any individual output-queue from completely occupying the common buffer-space and hence attempts to improve fairness and switch throughput. This technique of restricting the maximum length of individual queues works well in preventing a single output queue from completely occupying the common buffer-space. However, at higher loads, it is still possible for a group of output queues to completely occupy the common memory and unfairly deny (drop) the packets belonging to other source-destination pairs to access the common buffer-space for switching purposes.

2.4 Global-Static Threshold Based Sharing Scheme

According to this scheme [33], a restriction is put on the occupancy status of the entire global memory-space. In this scheme, a predetermined limit, called the global-static threshold value (GT) is imposed on the occupancy of the global memory-space (B). This GT value is calculated as follow; where (1-α) is a proportionality constant imposed on the occupancy of global memory-space (B)

GT = (1-α)·B packets *where* $0 < \alpha < 1$

If the occupancy of the global memory-space reaches that threshold value (GT) then the packets only from qualifying output-ports are admitted to the remaining memory space =

($\alpha \cdot B$) packets. A predetermined admittance policy is used to qualify the output-ports whose packets will be admitted in the remaining memory space. An example of an admittance policy is given in the section below.

2.4.1 Admittance Policy for Qualifying Output-Ports

Once the global-static threshold value (GT) is reached on the occupancy of the entire global memory-space then the admittance policy accept packets for only those output-ports whose output-queue length is less than ($\alpha \cdot B$) packets. Where B is the total shared-memory space and α is a proportionality constant ($0 < \alpha < 1$) imposed on the occupancy of global memory-space (B).

The output-ports with queue length less than ($\alpha \cdot B$) packets are considered underrepresented ports. Once the threshold limit GT is reached for the occupancy of the entire global memory-space then only the admittance policy goes in effect. The admittance policy selectively admits packets to the remaining memory space for only the underrepresented output-ports with queue length less than ($\alpha \cdot B$) packets. The algorithm for sharing of the common buffer-space among the packets of competing output-ports and with this admittance policy is given in Fig. 2.4

if (ΣQ_i) < GT packets **then** *where* $GT = (1-\alpha) \cdot B$

Accept packets for all the output-ports

elseif Q_d < ($\alpha \cdot B$) packets **then**

Accept the packets for the output-port *d*

else

Drop the packets for output-port *d*

Fig. 2.4 Global-static threshold scheme to regulate the sharing of the memory space

The global-static threshold based memory-sharing scheme achieves an efficient use of common buffer-space compared to individual-static threshold based memory-sharing scheme especially under conditions of low offered load. However, in the global-static threshold scheme the switch output-ports may not be fully utilized due to backlog in the remaining space =($\alpha \cdot B$) packets under conditions of high bursty-traffic loads.

2.5 Dynamic Threshold Based Sharing Scheme

Dynamic threshold based memory-sharing scheme is described in detail in [30]. According to this scheme, the occupancy of the memory that dynamically changes with the traffic conditions impose dynamically changing restrictions on the active output-ports from entering the remaining memory space at any given time. Each queue length (Q_d) inside common buffer-space is limited to a predetermined value called the dynamic threshold (DT) value. This DT value is function of the remaining buffer-space and it could increase or decrease depending on the traffic conditions at time t. Let *B* be the total buffer-space and ΣQ_i the sum of all queue lengths, i.e., the total memory used to store packets, then the dynamic threshold (DT) value at time t is calculated:

$DT(t) = \alpha \cdot (B - \Sigma Q_i)$ packets *where* $\alpha > 0$

Where α is proportionality constant of the available memory space ($B - \Sigma Q_i$) at time *t*. Packets belonging to output queue lengths (Q_d) with values less than DT are allowed to be stored in the remaining buffer-space; otherwise packets are dropped as illustrated in Fig. 2.5

if Q_d < DT packets **then**

Accept packets for the output-port *d*

else

Drop the packets for output-port *d*

Fig. 2.5 Dynamic threshold scheme to regulate the sharing of the memory space

Dynamic threshold scheme is inherently adaptive and dynamically respond in time according to the unused memory space. If there is sufficient buffer-space it allows active output-ports to increase their output queues as much as necessary. Contrary, if the buffer nearly overflows it imposes very restrictive conditions in a way only packet for less active ports are accepted. DT scheme reduces queue lengths by blocking new arrivals and waiting for the queue lengths of active ports to reduce naturally by the work of the switching system.

2.6 SMDA Based Memory-Sharing Scheme

Another memory-sharing scheme, namely the shared-memory with dedicated access (SMDA) is similar to the scheme called sharing with minimum allocation (SMA) scheme mentioned in [29]. SMDA or SMA based memory-sharing scheme aims to guarantee full utilization of the output-ports first, and then attempts to maximize the throughput for a given bursty-traffic. Under this scheme, a packet switch uses both the shared-memory and dedicated memory for its output-ports. A small percentage of total memory is dedicated to each output-port and the remaining memory is shared among all the ports. For a given output-port, the dedicated memory is first used to store the packets

and when the dedicated portion of the memory is full then only the packets can access shared-memory space of the switch.

Dedicated memory space for the SMDA scheme represents the minimum number of packet locations within memory space allocated to each output-port for its individual use and is calculated as following.

$$\text{Dedicated memory per port} = \frac{\alpha \cdot B}{N} \text{ packets} \qquad \textit{where } 0 < \alpha < 1$$

A portion of the total memory space B is divided equally among all the ports for its dedicated use. The amount of remaining memory space is shared among all the ports and is calculated as following.

$$\text{Shared-memory space} = B - N \cdot \frac{\alpha \cdot B}{N} \text{ packets}$$

Here, B is the total memory space, α is a proportionality constant for SMDA scheme and N is the number of I/O ports for NxN packet switch. Under this scheme, when the shared-memory space is occupied due to traffic backlog then the inactive ports still have a dedicated memory to allow its packets a passage through the memory space. Unlike other memory-sharing schemes, the SMDA or SMA scheme guarantee full output-utilization even under the conditions of backlog that is common with bursty Internet traffic.

2.7 Performance Evaluation

A simulation study is used to evaluate the performance of a shared-memory switch employing the various memory-sharing schemes. The measures of interest

considered for performance evaluation are the offered load for a bursty-traffic of a given average burst length (ABL), the average throughput, and the Packet-loss ratio (PLR) for a given memory size used in the system. Section 2.7.1 describes the probability nature of the bursty-traffic. Next, section 2.7.2 evaluates the various memory-sharing schemes using a balanced bursty-traffic scenario where bursts of data-traffic are uniformly distributed to all the output-ports. Section 2.7.3 considers an unbalanced bursty-traffic scenario. Burst of incoming packets have a greater chance to be destined to some output-ports than others output-ports.

2.7.1 Bursty-Traffic Model

To study performance of the various memory-sharing schemes, a bursty-traffic [7][42] is generated using a two state ON–OFF model (Fig. 2.6). Packets arrive at each input in a slot-by-slot manner and each input alternates between active and idle periods of geometrically distributed duration. During an active period, packets destined for the same output arrive continuously in consecutive time slots. There is at least one packet in an active period. An active period is usually called a burst. For the duration of and idle period, there are consecutive empty time slots due to the absence of traffic.

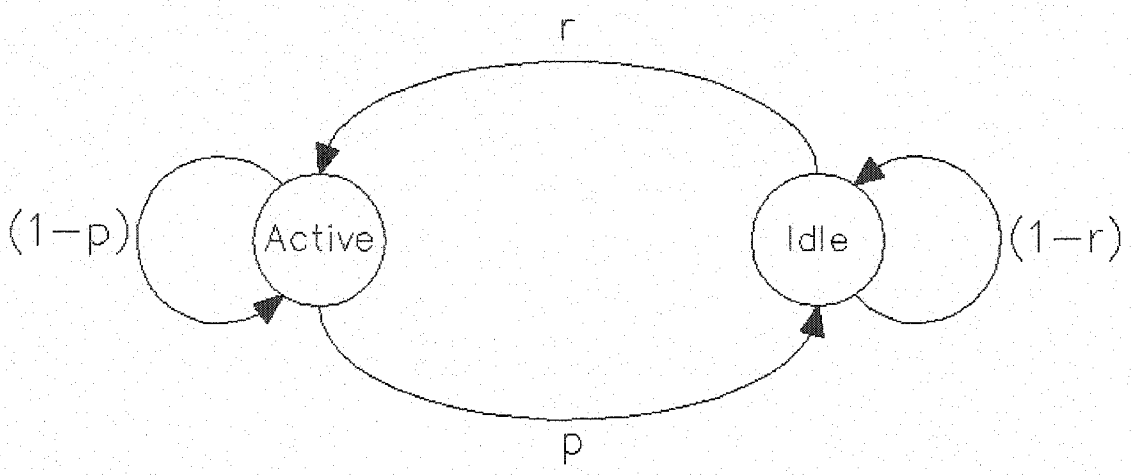


Fig. 2.6 Two-state ON-OFF model

If p and r characterize the duration of the active and idle period, respectively, then the probability that the active period lasts for a duration of i time slots is given by

$$P(i) = p(1-p)^{i-1}, \qquad \text{for } i \geq 1$$

and the corresponding average burst length (ABL) is given by

$$E_B[i] = \sum_{i=1}^{\infty} i \cdot P(i) = \frac{1}{p}$$

Similarly, the probability that an idle period lasts for j time slots is

$$R(j) = r(1-r)^j \qquad \text{for } j \geq 0$$

and the corresponding mean idle period is given by

$$E_I[j] = \sum_{j=0}^{\infty} j \cdot R(j) = \frac{1-r}{r}$$

Hence, for a given p and r, the offered load L is given by

$$L = \frac{E_B[i]}{E_B[i] + E_I[j]} = \frac{r}{r + p - rp}$$

2.7.2 Evaluation under Balanced Bursty-Traffic Conditions

The shared-memory switch in the simulation study is an NxN = 32x32 ports where the input and output lines work at identical rates. The number of data-packets that can be stored in the shared-memory space is 1,024 packets. All ports have access to the common memory space which is organized into logical queues corresponding to the different output-ports. Each input line of the switch receives data-packets from a bursty source with an average burst length (ABL) = 16 packets. Incoming bursts of packets are uniformly distributed to the output-ports, i.e. bursts of packets have equal probability to be destined to any output-port. Values of α constant utilized for the simulation are α = 0.1, 0.3, 0.5, 0.7, and 0.9 for all memory-sharing schemes. The effect of the memory-sharing scheme in the throughput performance and packet-loss is more evident at higher loads. As load is incremented for the switch, some packets might be dropped due to the lack of available memory space to buffer them. For that reason, the memory-sharing scheme should maximize the passage of packets through all input-output pairs by regulating the use of the memory space efficiently.

First, we evaluate average throughput (Fig. 2.7) in a shared-memory switch employing individual-static threshold based sharing scheme with α = 0.1, 0.3, 0.5, 0.7 and 0.9. We have a higher throughput at higher loads (>80%) with α = 0.1 compared to others values of α constant. Value of α = 0.1 means that any queue length could take up to 10% of the buffer-space. While limiting the amount of buffer-space for output-ports at higher loads guarantees a reasonable good access of packets to all the output-ports, the advantage of shared buffer-space in the switch is reduced. As a result, at low and mid loads packets could be dropped even when there is idle buffer-space. For the throughput

performance given for different α values, we also measure the corresponding packet-loss ratio (PLR) with balanced bursty-traffic. Fig. 2.8 shows the occurrence of packet-loss at low loads (<60%) for α = 0.1. This packet-loss is caused by the restriction in size for queue lengths that limits the use of buffer-space for output-ports. Also packet-loss at low loads varies considerably with different α values. In the individual-static threshold scheme the setting of α constant results difficult. At higher loads, it is preferable to have small α values. However at low loads, there is significant packet-loss with small α values; and it is necessary to have big α values to allow queue length for output-ports to expand in order to allow packets enter the switch to use the available buffer-space.

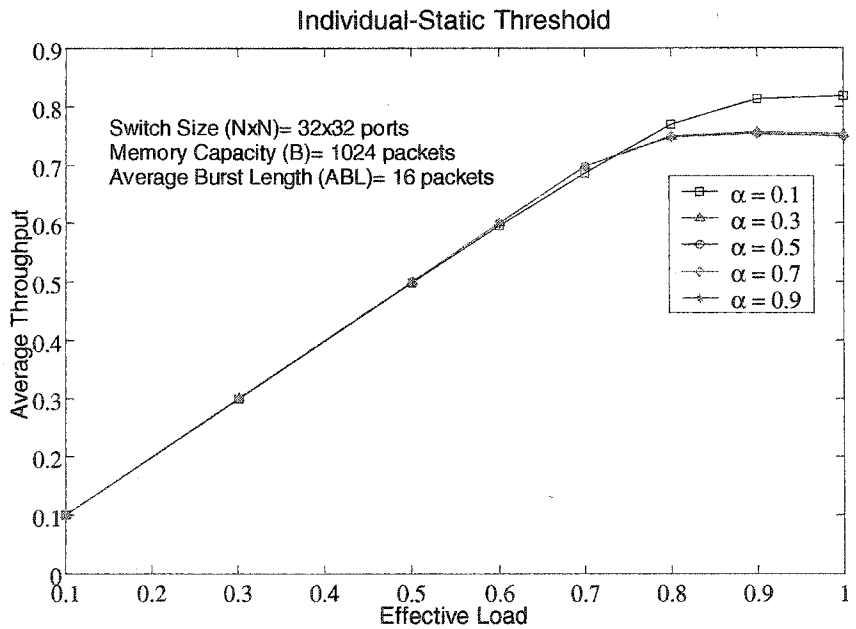


Fig. 2.7 Average throughput with balanced bursty-traffic conditions for individual-static threshold based sharing scheme

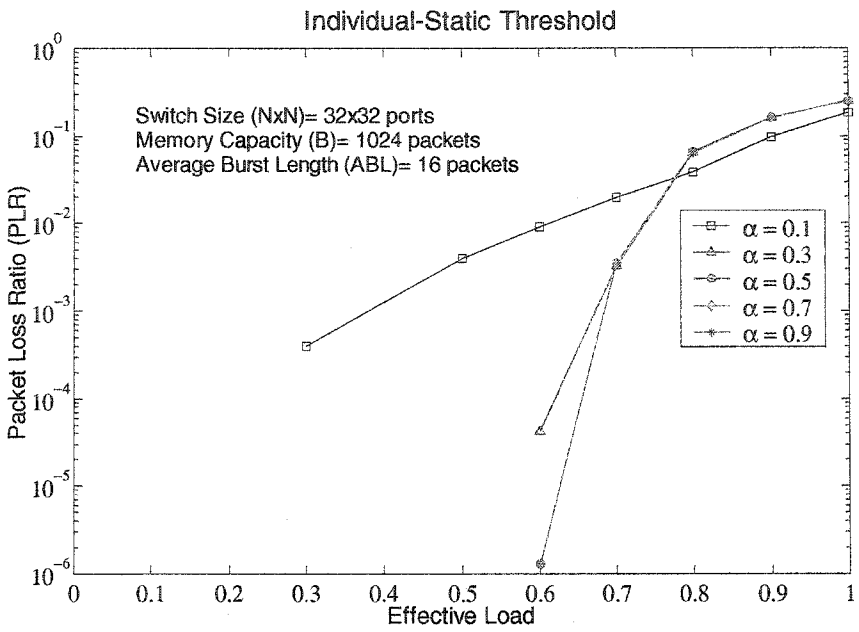


Fig. 2.8 Packet-loss ratio (PLR) with balanced bursty-traffic conditions for individual-static threshold based sharing scheme

Fig. 2.9 shows the average throughput in a shared-memory switch using the global-static threshold based sharing scheme to regulate the use of buffer-space. Throughput performance is higher with $\alpha = 0.1$ compared to others α values. For $\alpha = 0.1$, the global-static threshold is equal to 90% of the buffer-space. That means the memory is completely shared before the occupancy of buffer-space reaches 90% capacity and then after only packets for underrepresented ports are accepted into the switch according to the admittance policy. In the global-static threshold scheme, it is preferable to have a small α value that allow a complete sharing of memory up to the threshold value; and when load increases close by the full memory capacity, packets are selectively admitted to the remaining of the buffer-space. For the throughput performance in Fig. 2.9, the corresponding packet-loss ratio (PLR) (Fig. 2.10) is also measured under identical balanced bursty-traffic conditions. Unlike PLR in individual-static threshold scheme (Fig.

2.8), PLR in global-static threshold scheme (Fig. 2.10) present less variation for different values of α constant. Furthermore, there is no packet-loss in global-static threshold scheme at low loads (<60%) because queue lengths are no restricted in size at low loads.

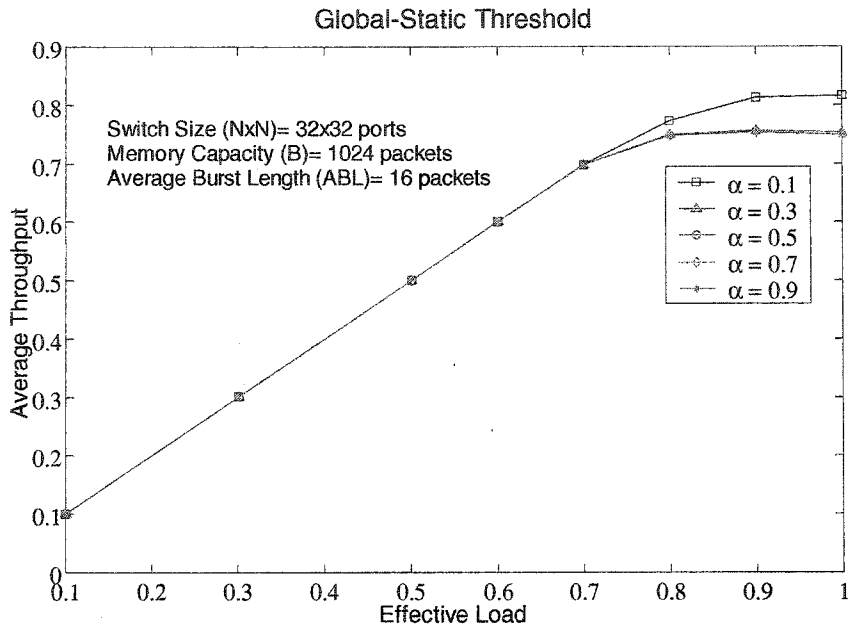


Fig. 2.9 Average throughput with balanced bursty-traffic conditions for global-static threshold based sharing scheme

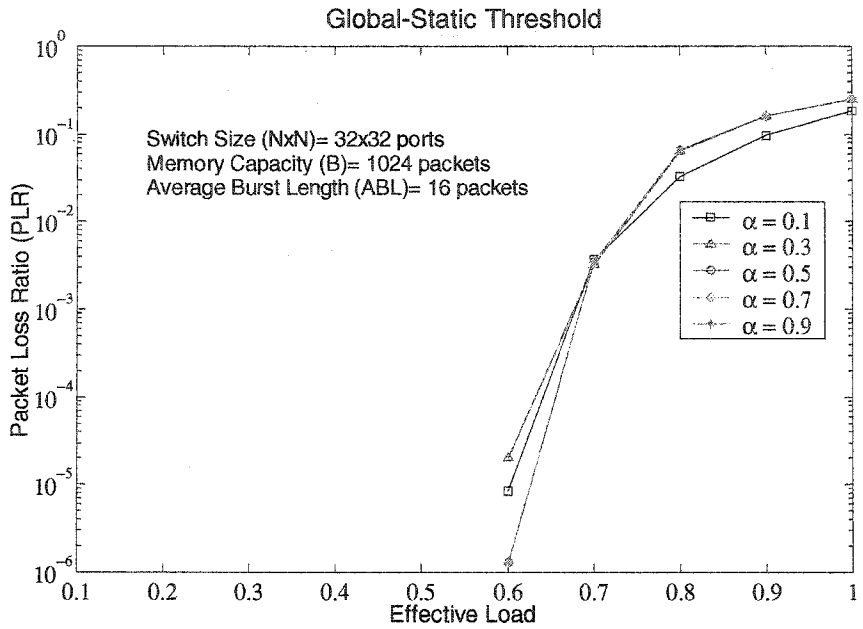


Fig. 2.10 Packet-loss ratio (PLR) with balanced bursty-traffic conditions for global-static threshold based sharing scheme

Evaluation of average throughput in a shared-memory switch using dynamic threshold based sharing scheme is presented in Fig. 2.11. Throughput is evaluated under balanced bursty-traffic. Dynamic threshold scheme achieves the highest throughput (86%) with large α values at full load (100%) compared to the other memory- sharing schemes. Results for throughput are very similar as α value is incremented. Therefore, dynamic threshold scheme is very robust for variations in α constant. For the throughput performance, we also measure the corresponding packet-loss ratio (PLR) under identical balanced bursty-traffic conditions. PLR for large α values is decreased; α = 0.9 represents that at any time, queue lengths for output-ports less than 90% of the idle buffer-space are permitted to built-in inside memory and their packets are accepted into the switch. In the dynamic threshold scheme, it is better to have big values of α constant in order to obtain the lowest PLR when burst of packets are uniformly distributed to the output-ports.

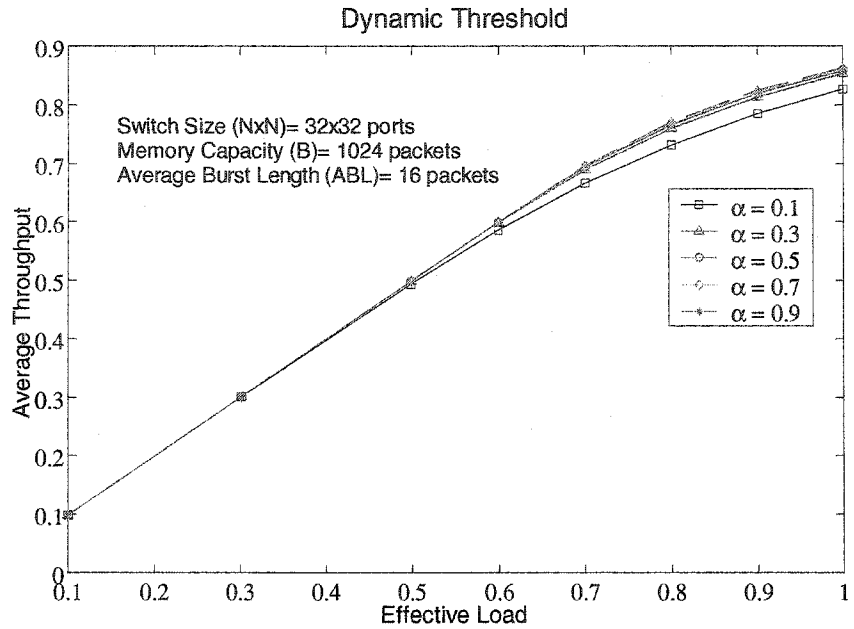


Fig. 2.11 Average throughput with balanced bursty-traffic conditions for dynamic threshold based sharing scheme

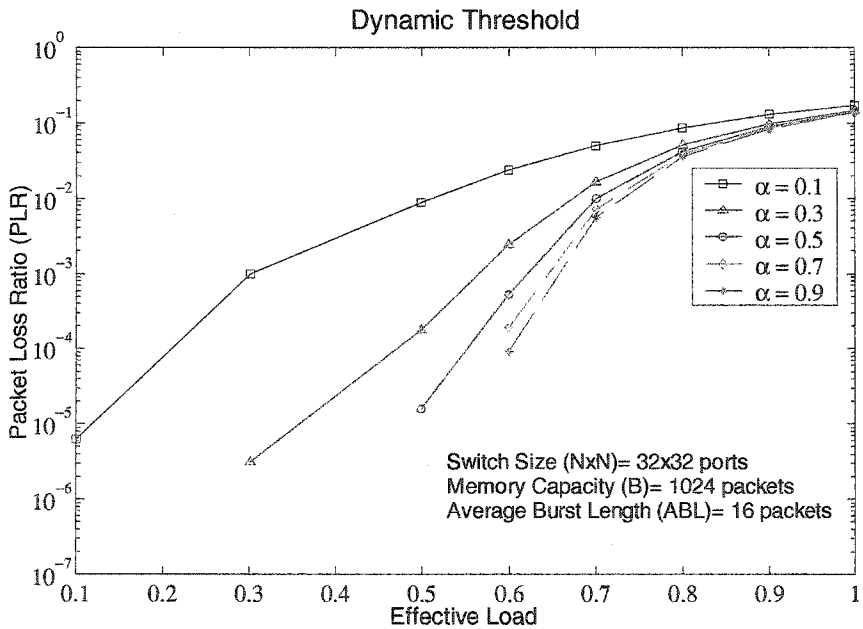


Fig. 2.12 Packet-loss ratio (PLR) with balanced bursty-traffic conditions for dynamic threshold based sharing scheme

Fig. 2.13 shows the average throughput in a shared-memory switch using the SMDA based memory-sharing scheme with balanced bursty-traffic. Throughput performance is higher using small α values. It is noticed that throughput in SMDA scheme (81%) is less than throughput for dynamic threshold scheme (86%) at full load (100%). This smaller throughput is explained due to the fact SMDA scheme dedicates an amount of buffer-space to each output-port exclusively, reducing the total amount of memory space shared by all output-ports. For the throughput performance, the corresponding packet-loss ratio (PLR) is also measured under identical balanced bursty-traffic conditions. SMDA scheme with $\alpha = 0.1$ present the lowest packet-loss compared to other α values. When $\alpha = 0.1$, the total memory space (1,024 packets) is divided in the dedicated memory (96 packets) and the shared memory (928 packets); where the dedicated memory (96 packets) allocates 3 packet locations exclusively to each output-port. It is better in the SMDA scheme to have a small amount of buffer-space dedicated to output-ports under balanced bursty-traffic conditions.

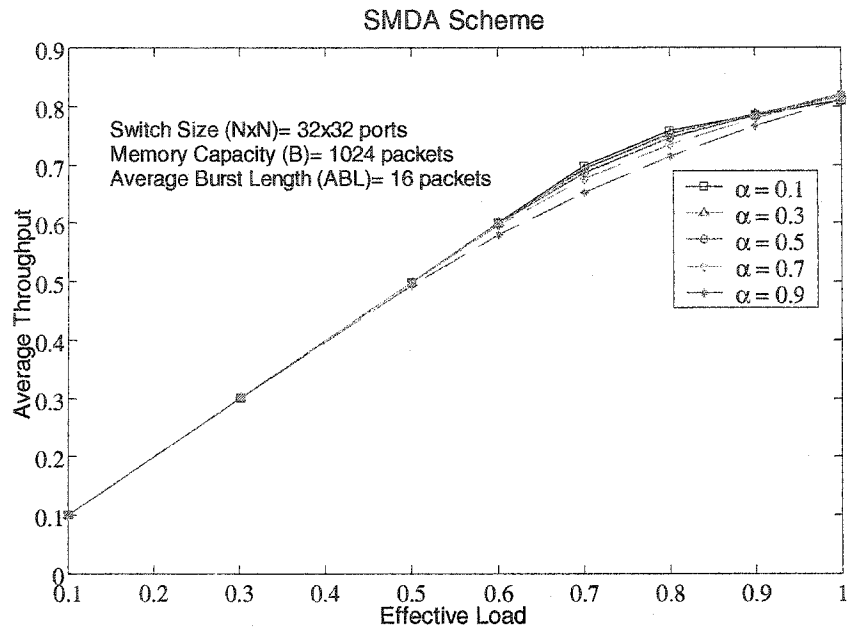


Fig. 2.13 Average throughput with balanced bursty-traffic conditions for SMDA based sharing scheme

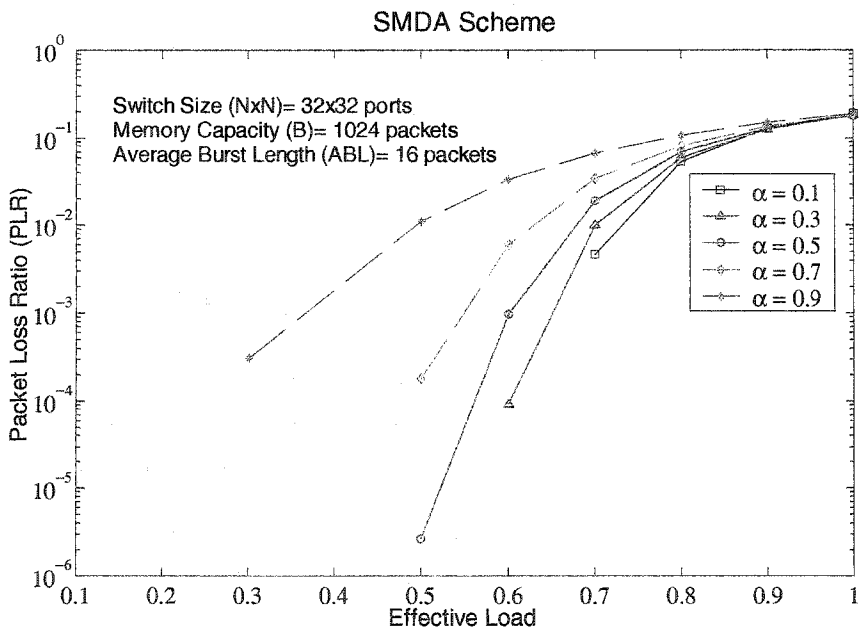


Fig. 2.14 Packet-loss ratio (PLR) with balanced bursty-traffic conditions for SMDA based sharing scheme

2.7.3 Evaluation under Unbalanced Bursty-Traffic Conditions

A switch with NxN = 32x32 ports is used for the simulation study. Buffer capacity in the shared-memory is 1,024 packets. All ports have access to the common memory space which is organized into logical queues corresponding to the different output-ports. Each input line of the switch receives data-packets from a bursty source with an average burst length (ABL) = 16 packets. Incoming bursts of packets are unevenly distributed to the output-ports. For that reason, some ports have a greater chance to receive packets than others. This unbalanced bursty-traffic scenario produces two classes of output-ports: very active output-ports and lightly active output-ports. In this simulation study, the probability that a burst of packets is destined to a very active port is considered to be four times greater than that of a lightly active port. Also, in this simulation, we consider the number of very active output-ports to be equal to that of the lightly active output-ports.

Packets destined to very active output-ports might attempt to take over most of the buffer-space; affecting the access to the switch to others packets destined to less active output-ports. This agglomeration of packets in highly loaded output-ports causes packets to be dropped due to insufficient buffer-space to handle these hot spots. In effect, an unbalanced bursty-traffic degrades the performance of a switching system. As illustrated in Fig. 2.15, average throughput in a switch with unbalanced bursty-traffic is considerably lower compared to the throughput with balanced bursty-traffic under identical switch resources and the memory-sharing scheme used.

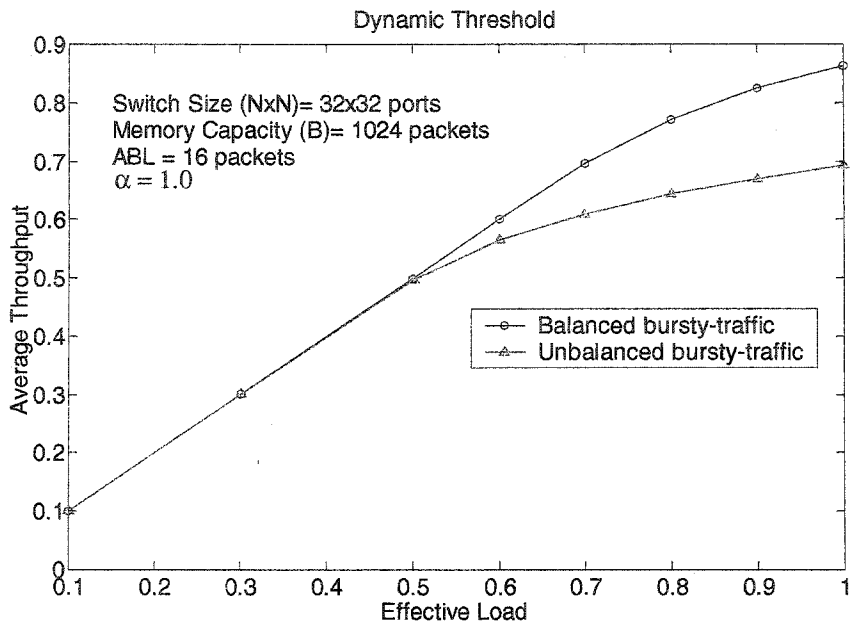


Fig. 2.15 Average throughput in a switch with balanced bursty-traffic and unbalanced bursty-traffic

The memory-sharing scheme should avoid the starvation in buffer-space caused by some very active output-ports due to the unbalanced bursty-traffic. The overall utilization of the switching system could be maintained high, if packets destined for less active output-ports are permitted to gain a fair access to the buffer-space. Furthermore, performance of the lightly active output-ports compared to very active output-ports allows us to measure the effectiveness of the memory-sharing scheme under evaluation.

Throughput versus α constant is shown in Fig. 2.16 for all memory-sharing schemes at 90% load. A high load (90%) will intensify the differences among the various memory-sharing schemes. The interval of α constant extends in (0, 1) for all memory-sharing schemes except for the dynamic threshold scheme that could be any value greater than zero. Individual-static threshold and global-static threshold based sharing schemes have a high throughput at small α values. For the applied load of 90% with unbalanced

bursty-traffic, both schemes namely the individual-static and global-static schemes experience a rapid degradation in performance when the α value is increased.

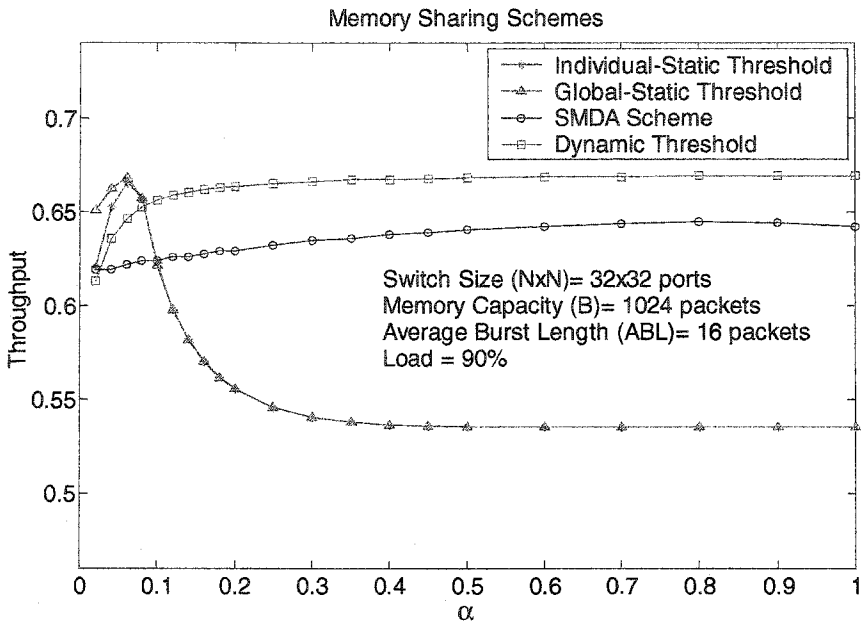


Fig. 2.16 Throughput vs. α constant, for different memory-sharing schemes at 90% load

SMDA and dynamic threshold based sharing schemes are very stable in variations of α constant. At high loads (90%) throughput increases slightly in SMDA scheme with greater α values, nevertheless this means that a large percentage of memory is dedicated to each output-port which reduces the advantages of the sharing effect. Dynamic threshold scheme shows the best performance, and stays stable with variations in α constant, but α value could be greater than one.

Fig. 2.17 shows the throughput versus α constant, for all memory-sharing schemes at 60% load. Decreasing the switch-load slow down the throughput variations with different α values compared to Fig. 2.16. Throughput performance for individual-

static threshold and global-static threshold schemes for 60% of applied load (Fig. 2.17) suffer a notable variation within the range they perform well at load of 90% (Fig. 2.16). SMDA scheme for 60% of applied load (Fig. 2.17) presents a decrease in throughput as α is incremented. This is in contrast to the throughput value at higher load of 90% (Fig. 2.16). It is apparent that SMDA scheme performs well under overload conditions. However, for smaller loads of 60%, the SMDA throughput somewhat decreases with increase in α value. This phenomenon occurs at low load because when α is incremented more memory space is reserved for each output-port (dedicated buffer), which decreases the advantage of memory sharing. Under this situation, there is a higher probability that some ports may have empty buffers while other ports may be discarding packets due to their buffers being full.

Dynamic threshold scheme performs similarly at high and low loads. It adapts to the changing traffic conditions, while there is a high occupancy of the memory space only packets from underrepresented output-ports are accepted to the remaining buffer-space. On the other hand when most of the buffer-space is idle queue lengths are allowed to expand and packets are accepted for very active output-ports

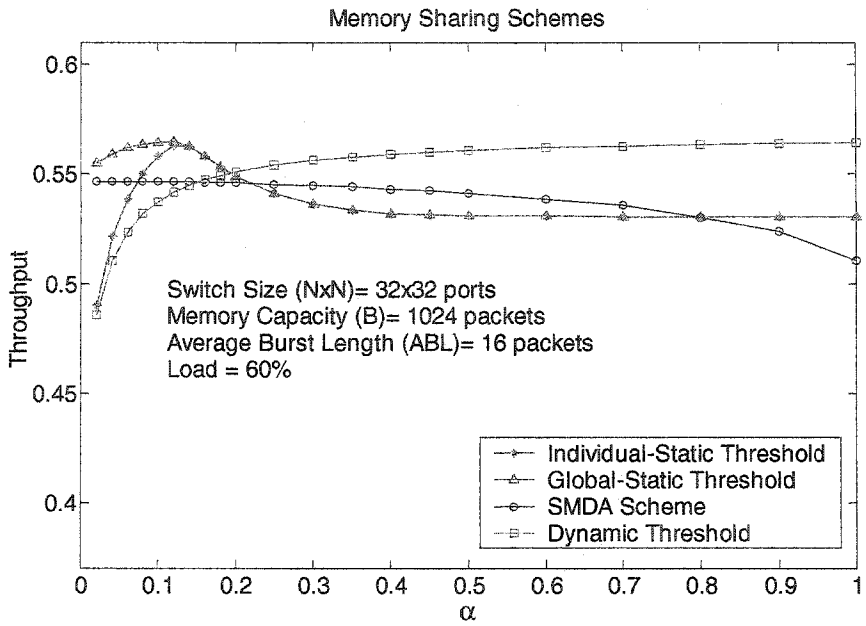


Fig. 2.17 Throughput vs. α constant, for different memory-sharing schemes at 60% load

The remaining figures in this chapter presents packet lost ratio (PLR) for each memory-sharing scheme under evaluation. Because of the unbalanced distribution of traffic to the output-ports, the packet-loss ratio (PLR) is evaluated individually for each of the two different classes of the output ports. We should expect the very active output-ports to incur a higher packet-loss compared to lightly active output-ports. A good sharing policy should increase the utilization of the switching system by allowing the packets for less active output-ports to also have access to the shared memory resources for switching purposes.

We evaluate in Fig. 2.18 and Fig. 2.19 the packet-loss ratio (PLR) using the individual-static threshold based sharing scheme under unbalanced bursty-traffic conditions. Individual-static threshold scheme with low values of α constant (i.e. α = 0.1, 0.2 and 0.3) causes the static threshold (ST) to be small. This in turn, limits to short

lengths the output queues. As shown in Fig. 2.18, there is packet-loss at low loads (<50%) caused by buffer-space denied to some output-ports due to the restriction imposed by the static threshold (ST). While a small ST value maintain short lengths in output queues and it ensures a fair access to buffer-space for output-ports; it also causes a great amount of buffer-space to be maintained idle when only a few output-ports are actively receiving packets. Therefore, at low loads some active output-ports are dropping packets (as seen with $\alpha = 0.1$ and 0.2) even when there is available buffer-space due to restriction imposed by ST value. Fig. 2.18 also indicates an extensive variation in packet-loss for the different values of α constant and the resulting ST value. It implies that the Individual-static threshold scheme is not robust for variation of α values and the imprecise setting of ST value can produce a poor performance of the switching system. Packet-loss ratio (PLR) is depicted in Fig. 2.19 with high values of α constant (i.e. $\alpha =$ 0.6, 0.7 and 0.8). Individual-static threshold scheme with high values of α constant produces a large static threshold (ST) value. This allows output queues to expand to long lengths, not restricting their built in inside memory. As α value get closer to one, the Individual-static threshold scheme tends to complete share the buffer-space. Consequently, a large ST value permit packet destined for very active output-port to block packets destined for less active ports; and lightly active output-ports suffer the same packet-loss experienced by very active output-ports as shown in Fig. 2.19.

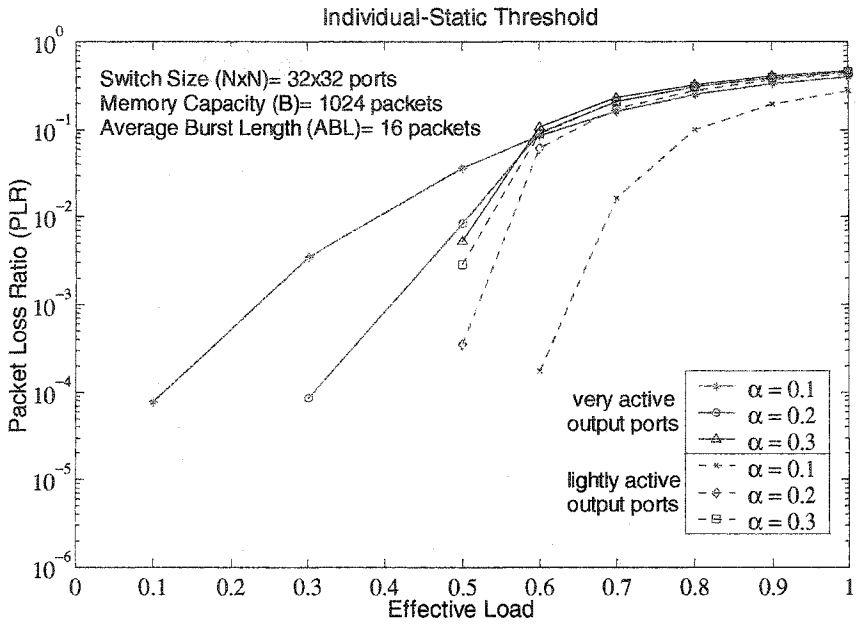


Fig. 2.18 Packet-loss ratio (PLR) for very active output-ports and lightly active output-ports in individual-static threshold based sharing scheme with $\alpha = 0.1$, 0.2, and 0.3

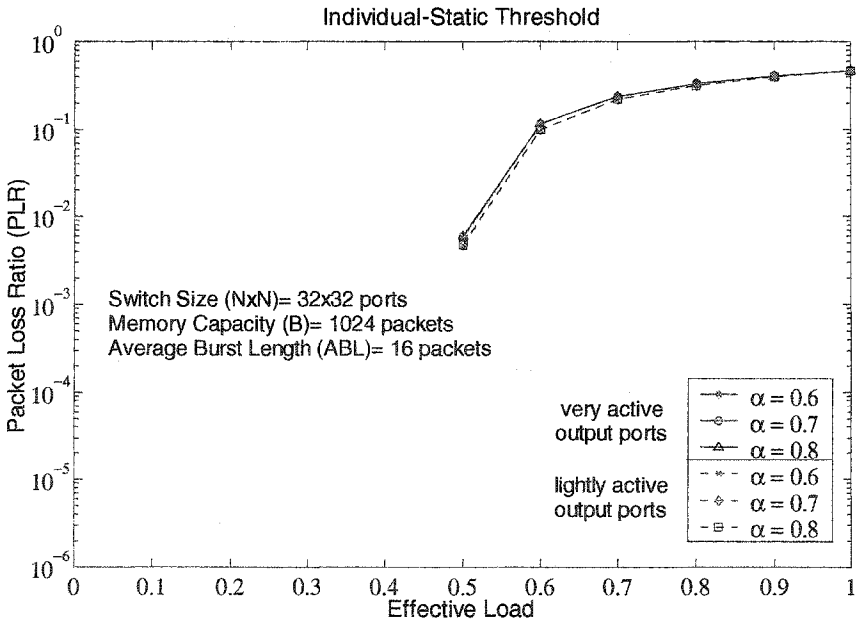


Fig. 2.19 Packet-loss ratio (PLR) for very active output-ports and lightly active output-ports in individual-static threshold based sharing scheme with $\alpha = 0.6$, 0.7, and 0.8

Packet-loss ratio (PLR) is evaluated in Fig. 2.20 and Fig. 2.21 for a shared-memory switch using the global-static threshold based sharing scheme under unbalanced bursty-traffic conditions. Unlike Individual-static threshold scheme (Fig. 2.18), global-static threshold scheme (Fig. 2.20) with low values of α constant (i.e. α = 0.1, 0.2 and 0.3) does not suffer packet-loss at low loads (<50%). Global-static threshold scheme allows the share of the buffer-space before the global threshold (GT) value is reached. After the GT value is reached only packet for less active output-ports are permitted to access the remaining buffer-space. Global-static threshold scheme also presents a wide variation in Packet-loss for the different values of α constant; so we need to be careful in the setting of α constant in order to obtain the desired performance of the switching system. Packet-loss ratio (PLR) is presented in Fig. 2.20 with high values of α constant (i.e. α = 0.6, 0.7 and 0.8). Like individual-static threshold scheme (Fig. 2.18), global-static threshold scheme with high values of α constant (Fig. 2.20) allows complete sharing of the buffer-space. As a result lightly active output-ports would incur the same packet-loss as that of very active output-ports. A higher value of global threshold (GT) allows most of the buffer-space to be completely shared. This can cause the very active output ports to monopolize the common memory space, which in turn would prevent the underrepresented output-ports from accessing the common buffer-space.

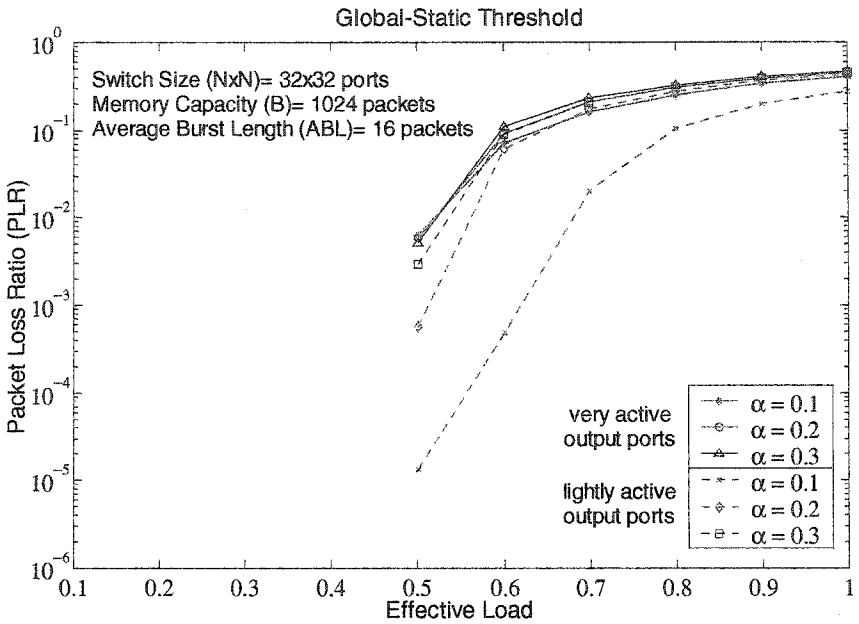


Fig. 2.20 Packet-loss ratio (PLR) for very active output-ports and lightly active output-ports in global-static threshold based sharing scheme with $\alpha = 0.1$, 0.2, and 0.3

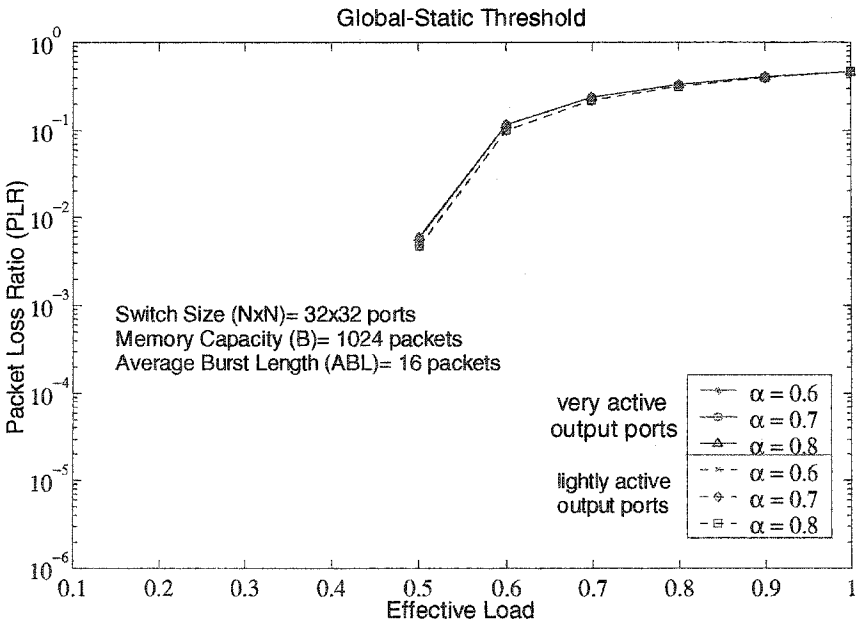


Fig. 2.21 Packet-loss ratio (PLR) for very active output-ports and lightly active output-ports in global-static threshold based sharing scheme with $\alpha = 0.6$, 0.7, and 0.8

Fig. 2.22 and Fig. 2.23 show packet-loss ratio (PLR) in a shared-memory switch using the dynamic threshold based sharing scheme under unbalanced bursty-traffic conditions. The value of α constant used in dynamic threshold scheme is $\alpha > 0$, with $\alpha =$ 0.5, 1.0, 2.0, 3.0, 5.0 and 9.0. The dynamic threshold scheme offers the best performance compared to the others memory-sharing schemes in this simulation study.

First, lightly active output-ports have the lowest packet-loss ratio (PLR) compared to other memory-sharing schemes. Dynamic threshold scheme provides packets destined for lightly active output-ports the best access to memory resources in the presence of congested output-ports. Fig. 2.22 and Fig. 2.23 show a difference of two orders of magnitude in the PLR between lightly active and very active output-ports. A fair memory-sharing scheme should ensure that an output-port drops packets (if necessary) proportional to the traffic present on it. The dynamic threshold scheme always allows the lightly active output-ports to access the memory space for switching purposes and hence the packet-loss incurred for this class of ports stays extremely low.

Second, the dynamic threshold scheme shows very stable PLR for different values of α constant. Different values of α constant do not produce drastically different switch performance. In Fig. 2.23, only a large value of α constant (i.e. $\alpha = 9.0$) shows a slightly difference in PLR among lightly active output-ports.

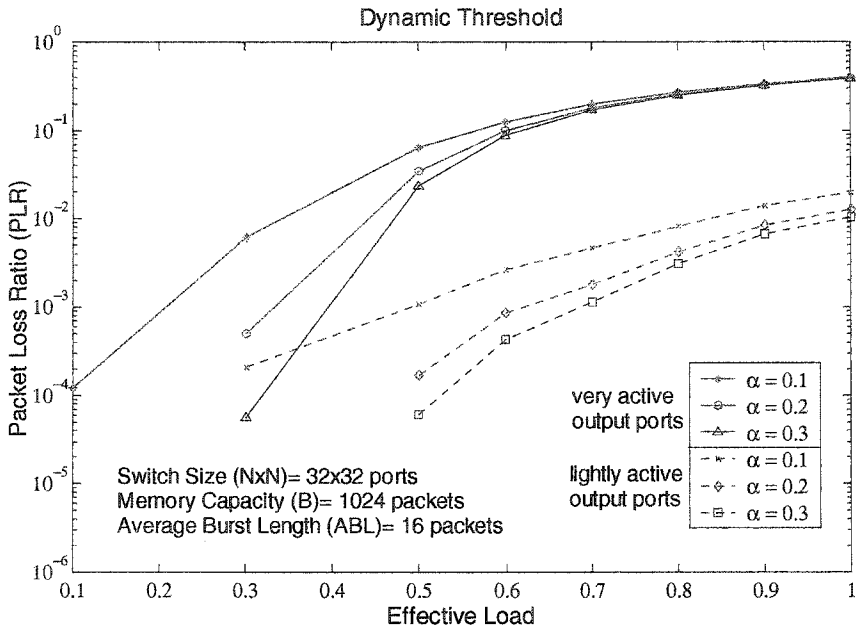


Fig. 2.22 Packet-loss ratio (PLR) for very active output-ports and lightly active output-ports in dynamic threshold based sharing scheme with α =0.5, 1.0, and 2.0

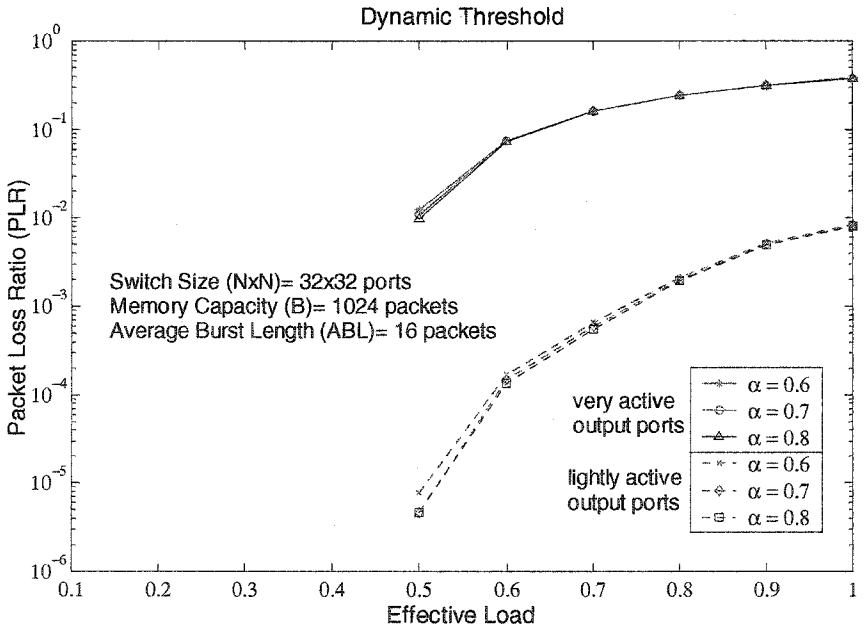


Fig. 2.23 Packet-loss ratio (PLR) for very active output-ports and lightly active output-ports in dynamic threshold based sharing scheme with α =3.0, 5.0, and 9.0

Fig. 2.24 and Fig. 2.25 depict the packet-loss ratio (PLR) incurred by a shared-memory switch using the SMDA based memory sharing scheme under unbalanced bursty-traffic conditions. In the SMDA scheme each class of output-ports have similar packet-loss ratio (PLR) among them for the various values of α constant. In SMDA scheme as the value of α constant is incremented, more buffer-space is reserved exclusively for each output-port. Results in Fig. 2.24 and Fig. 2.25 show that the PLR value for lightly active output ports are slightly decreased at higher loads (>60%) for greater values of α constant. This decrease in PLR is due to the fact that at high loads (> 60%), it is better for output-ports have more dedicated buffer-space. This will help prevent starvation for lightly active output ports even if the majority of the shared buffer-space is occupied by some very active output-ports. On the contrary if we dedicate too much buffer-space exclusively to each output-port (with high α values) then the performance-enhancing effect of sharing will be diminished and higher PLR can be experienced at lower loads. Under this situation, the active output-ports drop packets due to the lack of buffer-space while other output-ports may have idle buffer-space. This waste in buffer-space causes packet-loss at lower loads (e.g. 30% load) as shown in Fig. 2.25.

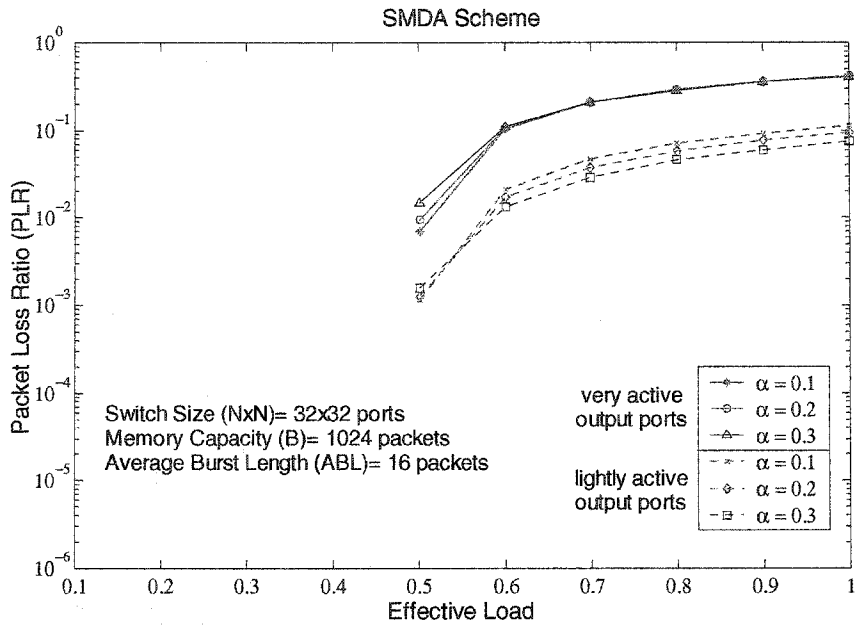


Fig. 2.24 Packet-loss ratio (PLR) for very active output-ports and lightly active output-ports in SMDA based sharing scheme with $\alpha = 0.1$, 0.2, and 0.3

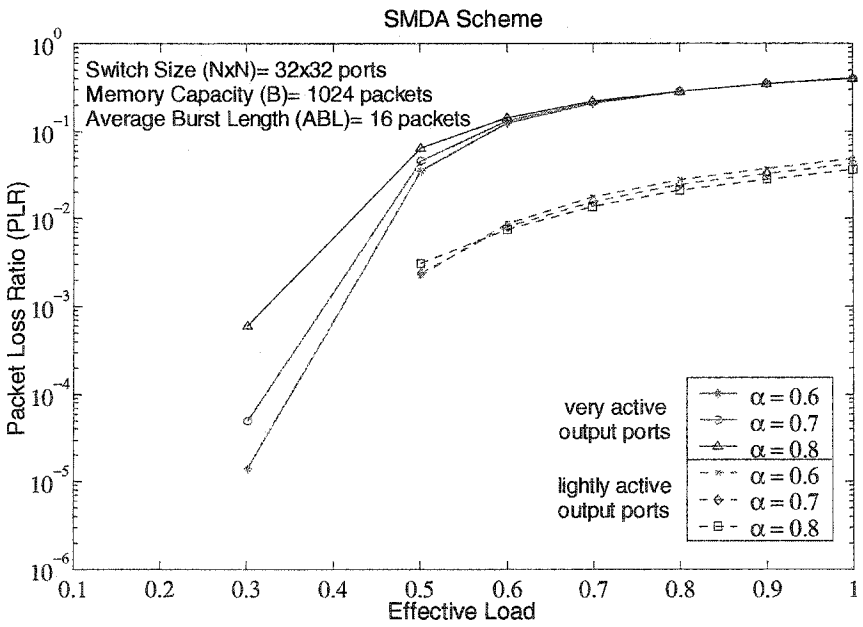


Fig. 2.25 Packet-loss ratio (PLR) for very active output-ports and lightly active output-ports in SMDA based sharing scheme with $\alpha = 0.6$, 0.7, and 0.8

CHAPTER 3

SWITCHING ARCHITECTURES DEPLOYING SHARED PARALLEL MEMORY-MODULES

3.1 Introduction

Internet router architecture deploying multiple memory-modules and shared-memory scheme can enhance the throughput and packet-loss performance of the switching system significantly [38][39][40][41][42][43]. The two well-known architectures that fall in this category are the shared-multibuffer (SMB) switch architecture [39] invented by Yamanaka et al and the sliding-window (SW) switch architecture [42] invented by Dr. Kumar. Fig. 2.26 shows that these two shared parallel-memories architectures are a subset of the shared-memory switches. They both are characterized by deployment of parallel memory-modules where the memory-modules are physically separate, nevertheless shared by all the input and output-ports of the switch. These architectures use the space-time-space (STS) model where the multiplexing and demultiplexing stages in the traditional shared-memory switches are replaced with crosspoint space switches. These features make these two architectures overcome the

memory-speed bottleneck created in traditional shared-memory switches due to high memory-bandwidth requirements of the broadband lines.

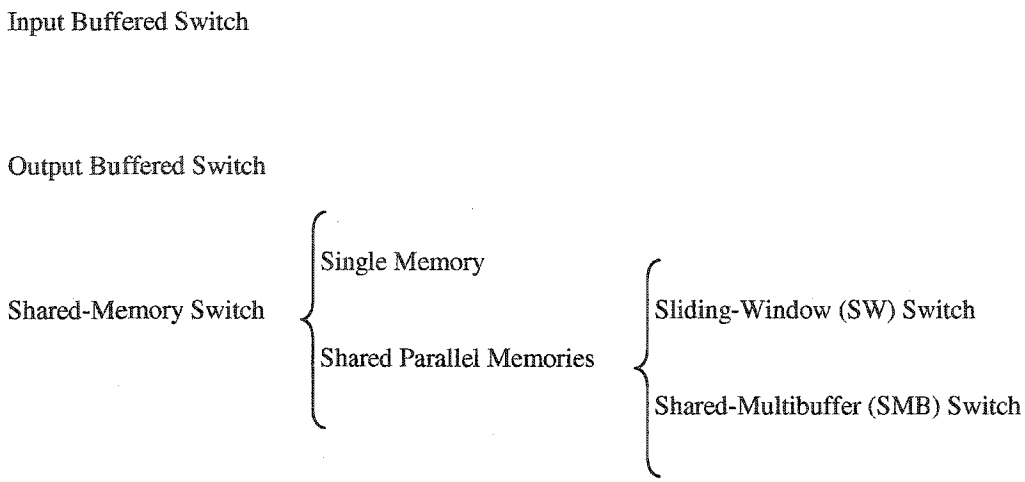


Fig. 3.1 Classification of switching architectures according to buffer strategy

One of the main differences between these two architectures is that SMB-switch architecture in [39] has centralized control whereas the SW-switch architecture has decentralized control [42]. Furthermore, the switching operation of SW-switch architecture is partitioned in multiple pipelined stages. As a result of its decentralized control and pipeline operation, the SW-switch architecture can be scaled to much higher capacity compared to that of SMB-switch architecture. Another difference between these two architectures is the switching scheme deployed by these switching systems. Both switching systems write multiple packets arriving in a given input cycle to the parallel memory-modules. Similarly, they both read multiple packets in a given output cycle out of the memory-modules for different output lines. Ideally, both the switching systems should be able to maximize parallel write and read of packets to the parallel memory-modules, so that all the packets input/output in a given cycle require only one write/read

memory-cycle. If data packets switched in a given cycle are written and read in parallel to and from different memory-modules respectively then the required memory-speed is equal to the line-speed. However, the lack of 100% parallel operations for memory-modules for data packets in every cycle, can increase the number of memory-cycles needed to write/read packets to/from the parallel memory-modules. Therefore, at times, multiple packets input in a given switch-cycle will need to be written/read to/from the same memory-modules and hence require speeding up of the memory-modules compared to input line-speed. This will, in turn, increase speed of memories modules to solve memory-contention when multiple packets are needed to be written/read to/from a memory-module in one cycle.

This chapter compares these two classes [39][42] of router/switch architectures, and performance of their assignment schemes to distribute packets to the memory-modules. Simulation results are used to evaluate throughput performance and memory-bandwidth requirement of these two switching systems.

3.2 Shared-Multibuffer (SMB) Switch Architecture

The Shared-Multibuffer (SMB) switch architecture [39] is illustrated in Fig. 3.2. Multiple memory-modules are shared among all the input/output-ports through cross-point switches. The stored packets are read out and transferred to destination ports by an output-side cross-point switch. The control block for the switching system is centralized and maintains a buffer address queue for each output-port and an idle-addresses pool to store the vacant addresses.

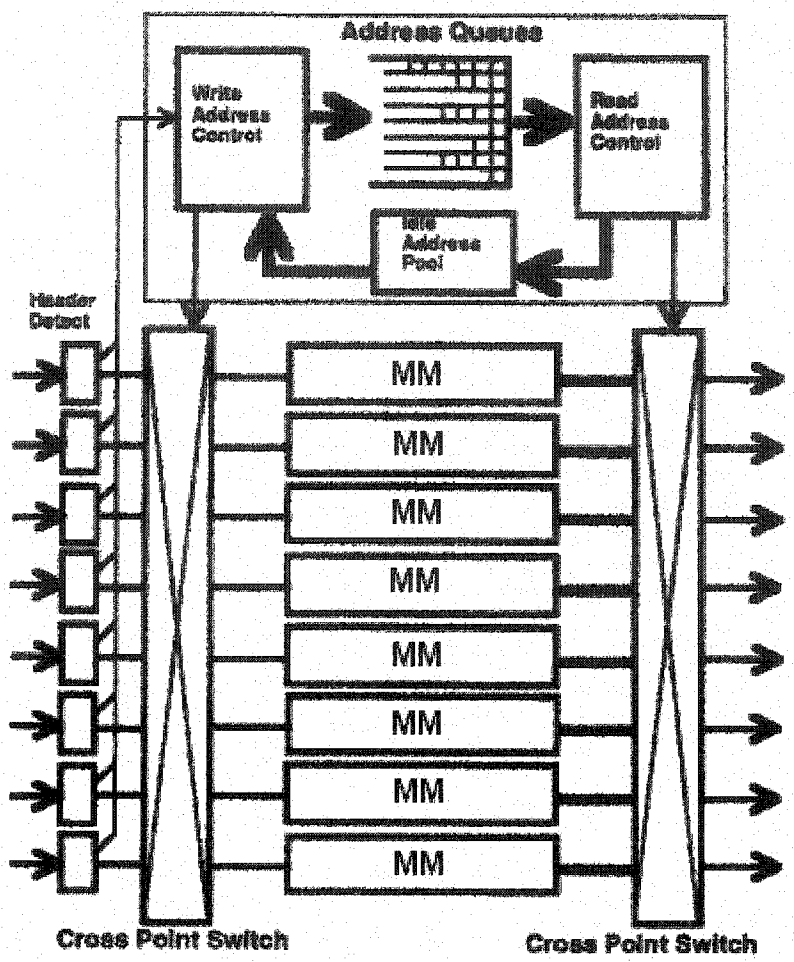


Fig. 3.2 Shared-Multibuffer (SMB) switch architecture [39]

For complete sharing of memory-modules, the buffer address-queue for each output-port needs to be as large as the total memory space. The centralized controller is responsible for coordinating all the switching functions for the SMB-switching system, which in turn limits the scalability of this architecture.

3.2.1 Admittance Policy for the SMB-Switch Architecture

Fig. 3.3 shows the admittance policy for the SMB-switch architecture, that is the necessary steps in the switching system in order to accept incoming packets. Arriving packets within a switch-cycle are represented by set X in step **200**. A packet is removed from non-empty set X and determined output-port d and queue length Q_d. (steps **202** and **204**). It is checked for available space in the memory-modules, in step **206** summation of all queue lengths should be less than memory capacity $m \cdot \sigma$, where we assume, m = number of memory-modules and σ = packet locations in each memory-module. Furthermore, in order to share the common memory space among input and output-ports, the dynamic queue length threshold [30] is used to regulate the sharing of memory space between output-ports in step **208** (dynamic threshold scheme was described in detail in chapter 2). Queue length for output d should be less than dynamic-threshold (DT) value to admit packet. Both conditions in steps **206** and **208** should be met in order to accept an incoming packet into the SMB architecture (step **210**); otherwise packet is denied access to the SMB architecture and dropped (step **212**). This process is repeated for all packets in set X.

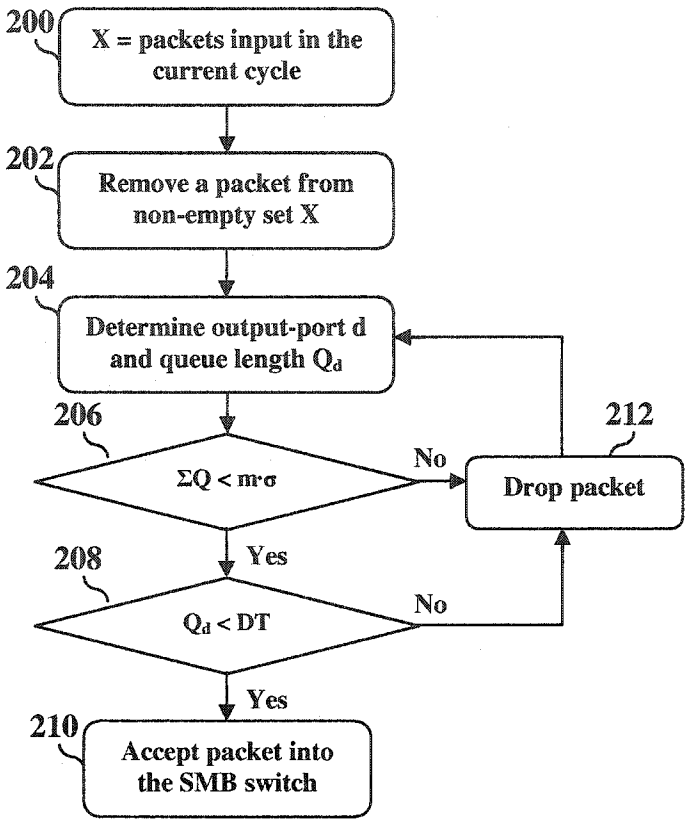


Fig. 3.3 Admittance policy for the SMB-switch architecture

3.2.2 Assignment of Memory-Module for the SMB-Switch Architecture

Once a packet is accepted into the SMB architecture, we need to assign a memory-module to store the incoming packet. In SMB architecture [39], an incoming packet is assigned to the least occupied memory-module. In other words, the less occupied memory-module is given higher priority for an incoming packet to be written to. Operation of SMB architecture to assign a memory-module can be depicted by flow chart and packet-counter array PC[i] in Fig. 3.4. PC[i] is used as a counter to represent the occupancy of i^{th} memory-module deployed in the switching system; PL_{min} holds the shortest value found in the search through PC array, i.e. PL_{min} represents the shortest

length of packets available in the memory-modules; i represents the memory-module having the lowest occupancy, and the one that is selected to store an incoming packet. Also y is a temporary variable used in the calculation. First, in step **300** variables y and PL_{min} are initialized with values from slot 1 in the PC array, i.e. PC[1]. Then a search is done through the packet-counter array PC[i] to find the least occupied memory-module (steps **302**, **306** and **308**). Every time, we find a PC slot with a shortest value than the current value of PL_{min}, it is assigned to PL_{min} and $i = y$ in step **308**. The search is done through the entire packet-counter array PC[i]; variables i, and PL_{min} are updated as necessary through this search, once y variable reaches m value (number of memory-modules) in steps **304** or **310**, the search had been completed through the entire PC array and we have the memory-module (= variable i) to store the incoming packet in the switching system. For the example of Fig. 3.3 the search in PC array is done from $i = 1$ to $i = m$. The shortest value found is assigned to $PL_{min} = 5$ in slot $i = 4$, and it becomes the memory-module = $i = 4$ to store the incoming packet.

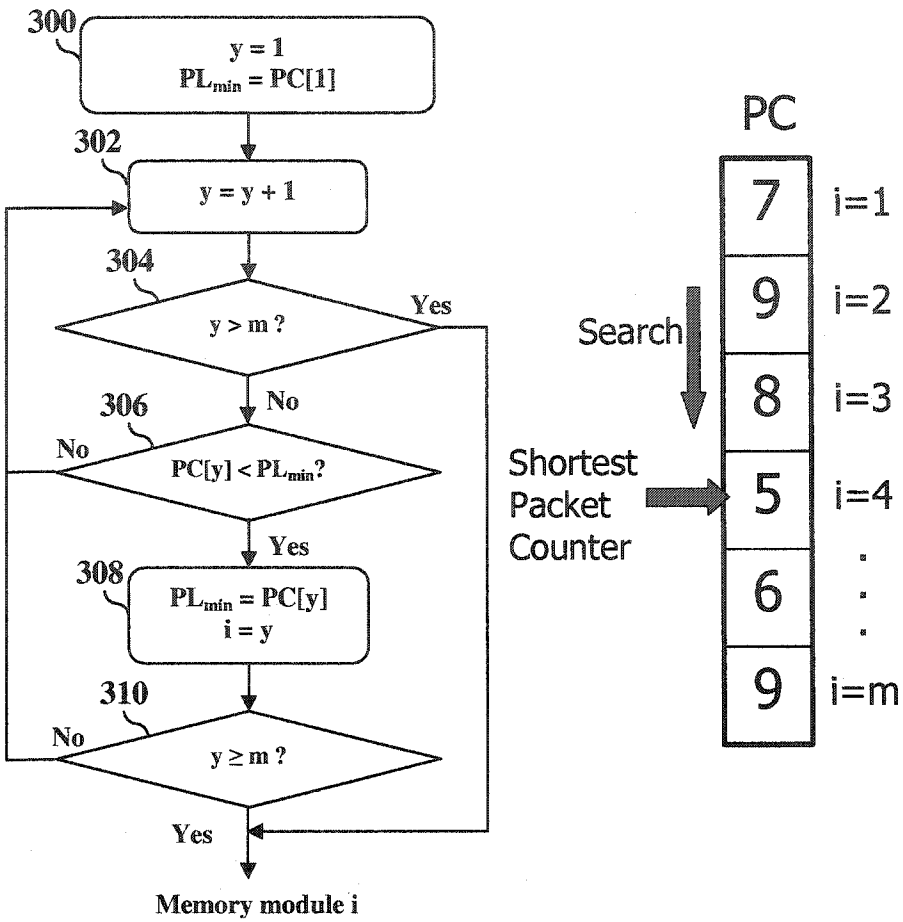


Fig. 3.4 Assignment of memory-module (i) to store packet

In the SMB architecture [39], it usually happens that two or more packets scheduled to go out in the same switch-cycle, get assigned to the same memory-module. For such a memory contention during the READ stage, the switching system requires speedup of memory-modules to resolve memory-conflicts. This requires the memory-bandwidth to be increased to be able to output multiple packets from the same memory-module in the same output cycle. We should notice that in SMB architecture, packets could be assigned to different memory-modules, and the WRITE stage can operate at the

input line-speed. Therefore, the increase in memory-bandwidth is only required in the READ stage for this architecture.

3.3 Sliding-Window (SW) Switch Architecture

The class of sliding-window switch architecture [42] is characterized by deployment of parallel memory-modules and decentralized control. The overall switching function of the SW-switch architecture is partitioned into various stages such that all stages can perform the needed switching functions independently based only on the information available locally and in a self-routing tag (*i, j, k*) attached to packets. These independent stages of the switch can operate in a pipeline fashion to achieve overall switching operation. The switching operation is decentralized in the sense that there is no central controller directly connected to various components of the switching system and controlling/managing various operations of the switching system.

The SW based switching system consists of the following independent stages, namely, the self-routing parameter assignment circuit, the input-interconnection network, parallel WRITE stage, parallel READ stage and output-interconnection network. The input lines of the switch are denoted by $\mathbf{1}_1, \mathbf{1}_2, \ldots, \mathbf{1}_N$ and the output lines are denoted by $\mathbf{2}_1, \mathbf{2}_2, \ldots, \mathbf{2}_N$. Input lines carry the incoming data packets and the output lines carry the outgoing data packets after being switched to their output destination by the SW-switching system of Fig. 3.5.

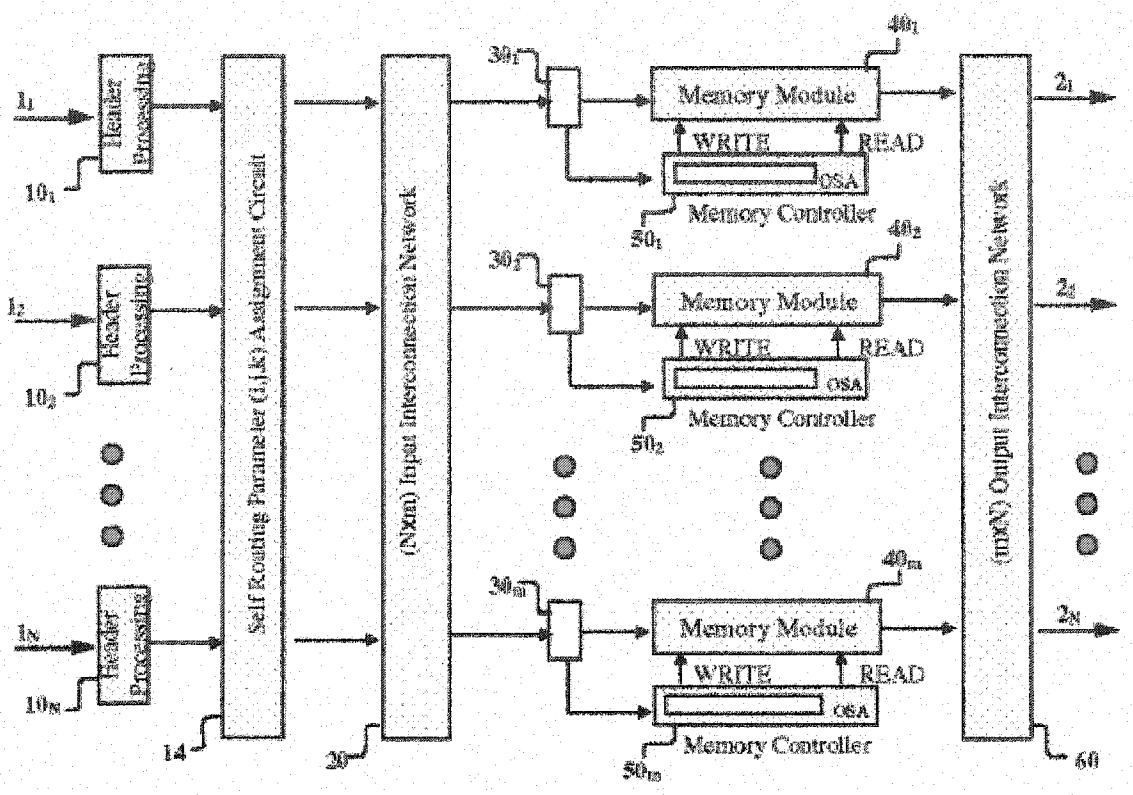


Fig. 3.5 Sliding-Window (SW) switch architecture [42]

The incoming packets are processed by header processing circuits for extraction of the output-port destination address denoted by *d*. The destination address of incoming packets is forwarded to a self-routing parameter assignment circuit **14**. The self-routing parameter assignment circuit **14** uses the output destination information and a parameter assignment method to produce an additional set of self-routing parameters (*i, j, k*) for incoming data packets. The self-routing parameters (*i, j, k*) are then attached as a self-routing tag to the incoming data packets. Thereafter, incoming packets use the attached self-routing tag (*i, j, k*) to propagate independently through various stages of the SW switching system of Fig. 3.5. Functions of these parameters are described below:

$i \rightarrow$ this parameter designates the memory-module that the packet will be stored in

$j \rightarrow$ this parameter designates the memory-location in the i^{th} memory-module in which the packet will be stored

$k \rightarrow$ this is an additional parameter for scan-plane which helps decide when a given packet is to be read out of the memory for its output from the switch

The input interconnection network **20** in Fig. 3.5 uses the parameter i of the routing tag of an incoming packet to route the packet on a given input line to its i^{th} output line which in turn is connected to the respective i^{th} memory-module. Input modules $\mathbf{30_1}$, $\mathbf{30_2}$, ..., $\mathbf{30_m}$ are used corresponding to each one of the memory-modules $\mathbf{40_1}$, $\mathbf{40_2}$, ..., $\mathbf{40_m}$. The input modules $\mathbf{30_1}$, $\mathbf{30_2}$, ..., $\mathbf{30_m}$ can be used for multiple purposes such as serial to parallel conversion, packet processing to provide the parameters j and k information from the packet's self-routing tag to memory controllers $\mathbf{50_1}$, $\mathbf{50_2}$, ..., $\mathbf{50_m}$, etc. The memory controllers $\mathbf{50_1}$, $\mathbf{50_2}$, ..., $\mathbf{50_m}$ use the parameter j to write the received packet in the j^{th} memory location of the corresponding memory-modules $\mathbf{40_1}$, $\mathbf{40_2}$, ..., $\mathbf{40_m}$.

Corresponding to each memory controller $\mathbf{50_1}$, $\mathbf{50_2}$, ..., $\mathbf{50_m}$ in Fig. 3.5 there is one output scan array (OSA) each with σ slots. The j^{th} slot of the OSA holds the scan value of a received packet stored in the corresponding j^{th} location of its memory-module. OSA of each memory controller is updated at the time of write and read of data packets to and from the respective locations in the memory-modules. During the packet WRITE cycle of an incoming packet to j^{th} memory location in a given i^{th} memory-module, the associated scan-plane value (k) of the received packet is stored in the corresponding j^{th} slot in the OSA of the corresponding memory controller. During the READ cycle of a packet from the j^{th} location of a memory-module, the corresponding j^{th} slot in the OSA is set to zero to indicate empty memory location in the corresponding memory-module. During the

packet READ cycle, the data packets are output from parallel and independent memory-modules $\mathbf{40_1}$, $\mathbf{40_2}$, ..., $\mathbf{40_m}$ and are finally routed to respective output destinations $\mathbf{2_1}$, $\mathbf{2_2}$, ..., $\mathbf{2_N}$ by the output interconnection network **60**. The output interconnection network **60** makes use of the output-port destination information d stored in a packet's header to route each packet to a final output destination $\mathbf{2_1}$, $\mathbf{2_2}$, ..., $\mathbf{2_N}$

3.3.1 Admission Control in SW-Switch Architecture

The preliminary steps to admit packets into the SW-switch architecture are shown in Fig. 3.6. Arriving packets within a switch-cycle are represented by set X in step **500**. A packet is removed from non-empty set X and determined output-port d and queue length Q_d (steps **502** and **504**). If queue length for output d (Q_d) is less than $p \cdot \sigma$ and DT values (steps **506** and **508**), then packet is received; otherwise packet is dropped in step **512**. The maximum queue length for an output-port is specified by $p \cdot \sigma$, where p is the number of scan-planes and σ the number of packet locations in each memory-module. This $p \cdot \sigma$ limit forces a specific sharing scheme in the switching system. In addition, dynamic queue length threshold [30] is used to regulate the sharing of memory space between output-ports (dynamic threshold scheme was described in detail in chapter 2). Steps in Fig. 3.6 are repeated for all input packets in non-empty set X.

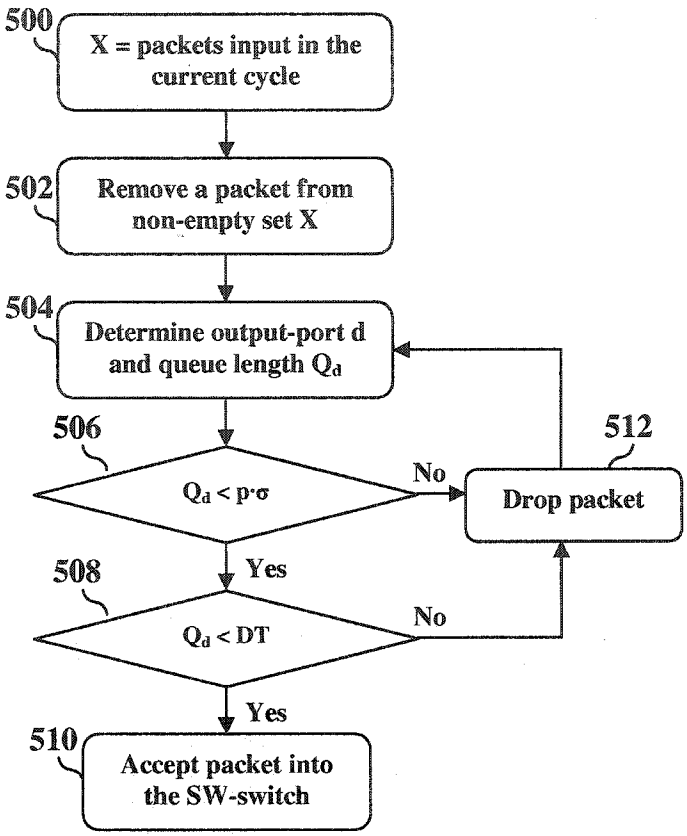


Fig. 3.6 Preliminary steps to admit packets into the SW-switch architecture

In the SW-switch architecture [42], first (*j, k*) parameters are computed. Then *i* parameter is computed based on the values of (*j, k*) parameters. There are different methods to assign a memory-module (*i* parameter) to store an incoming packet that has a direct impact on the performance of the SW-switch architecture. These methods involve a search in the output scan vector (OSV). Total memory can be seen as composed of σ OSV arrays. Each OSV consists of *m* slots, which correspond to the *m* memory-modules in the switching system. A search should be done for every input packet in the corresponding OSV to compute *i* parameter and store the incoming packet in the *i* memory-module. Two different methods are presented for the assignment of *i* parameter, as presented in [44][45][46] namely *assignment scheme-1* and *assignment scheme-2*.

3.3.2 Assignment Scheme–1 for i-Parameter

In this scheme, the first available slot found in the corresponding j^{th} OSV is assigned as the *i* parameter (Fig. 3.7). According to this scheme, the packets belonging to the same input cycle are assigned values of *i* in an increasing order. If none of the greater values of *i* are available, then only the smaller values are chosen in a mod fashion (step **602**). Packet could be dropped in step **608**, if not available slot was found in the entire OSV, that is when temporary variable *y* reaches *m* value (entire number of memory-modules) in step **604**. For the example in Fig. 3.7, slots in j^{th} OSV that have a value of zero indicate available memory locations. The search start at $i = 2$ and continues until available slot $i = 4$ is found, which becomes the *i* parameter assigned in self-routing tag to incoming packet.

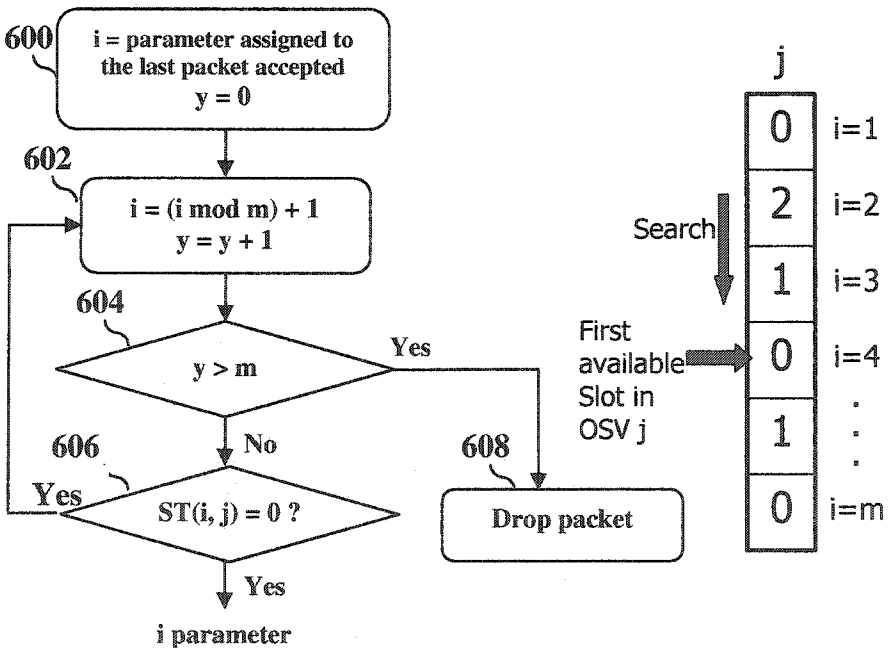


Fig. 3.7 Assignment scheme-1 in SW-switch architecture

Some packets in the same cycle could obtain similar values for *i* parameter because the search is done at different OSVs. Available slots in OSV arrays might have the same positions. Therefore in *assignment scheme-1* at times, there might be multiple packets that need to be written to the same memory-module in the same switch-cycle. Because of the need to write multiple packets to a memory-module in the same input cycle, it becomes necessary to speedup the memory-modules (hence increasing the memory-bandwidth of the switching system) in the WRITE stage to be able to write multiple packets to the same memory-module.

3.3.3 Assignment Scheme –2 for i-Parameter

For this scheme, similarly to *assignment scheme-1*, packets belonging to the same input cycle are assigned values of *i* in an increasing order; and the corresponding j^{th} OSV is searched to find an available slot (Fig. 3.8). However, an additional array called *temp* array is used to keep track if that slot was employed previously by another packet in the same switch-cycle. If slots are available in both, the OSV being searched and *temp*, (i.e., conditions in **706** and **708** are satisfied) then that slot number is assigned as the *i* parameter. Packets could be dropped in step **710**, if the search is completed in OSV and *temp*; and conditions in steps **706** or **708** are not fulfilled simultaneously. Fig. 3.8 shows an example where a search is done through j^{th} OSV and *temp array*. The search for available slots (where slots have a zero value) start at $i = 1$ and continue to $i = 4$ where both slots are available in j^{th} OSV and *temp array*. Thus $i = 4$ is assigned as the corresponding *i* parameter in self-routing tag for incoming packet.

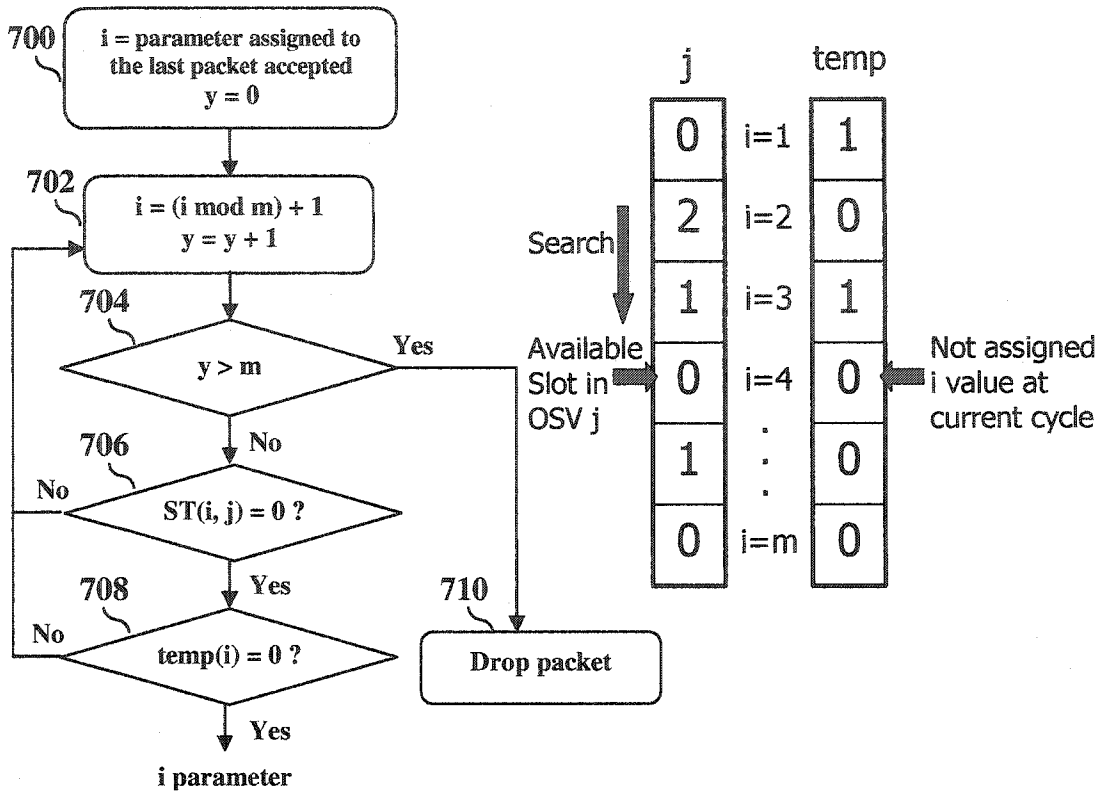


Fig. 3.8 Assignment scheme-2 in SW-switch architecture

Additional *temp* array ensures packets obtain different values of *i* parameter in a given switch-cycle. As a result, *assignment scheme-2* is able to guarantee that only one packet is stored to a memory-module during a switch-cycle. Incoming packets are written in different memory-modules and the WRITE stage is able to complete 100% parallel operation each cycle. Thus the required memory-speed is equal to the line-speed for this scheme.

3.4 Simulation Results

The measures of interest considered in the simulation studies are the average-case memory-bandwidth requirement, worst-case memory-bandwidth requirement, average throughput, and packet-loss ratio against offered load. The bursty data-traffic [7][42] generated for the simulation has an average burst length (ABL) = 8 packets and the incoming bursts of packets are uniformly distributed to all the ports. For these simulation experiments, we consider that architectures use the dynamic queue length threshold [30] with $\alpha = 1$ to fairly regulate the sharing of the buffer-space among competing ports. Initial simulation experiments measure performance of three different configurations for buffer accommodation in a sliding-window switch. All these configurations maintain the same total memory deployed in the switch = 2,048 fixed-size packets, however their m and σ values are varied to measure their impact on the switch performance.

The number of memory-modules used for three configuration are m = 32, 64 and 128. Also the corresponding number of packet locations (σ) in a memory-module vary from σ = 64, 32, and 16. First, configuration-1 has m = 32 and σ = 64. Even though, this configuration is not suggested for the sliding-window switch [42] due to the insufficient memory-modules for parallel memory-operations. SW architecture recommends use of at least $m = 2 \cdot N$ in [42], where N is the switch size. Configuration-2 partitions the total memory space as m = 64 and σ = 32. In this configuration, we intend to see the effect of increasing the memory-modules and yet keeping total memory size constant. Configuration-3 partitions the total memory space as m = 128 and σ = 16. In this configuration also, we intend to see the effect of increasing the memory-modules while keeping the total memory-storage constant for the switch. By performing simulations

under a bursty-traffic, we are interested to measure and compare how these three configurations for the deployed memory space of a fixed size in the switch might have varying impact on memory-bandwidth requirement, throughput performance and packet-loss ratio (PLR)

Average memory-bandwidth requirement of the switching system measures the number of memory-cycles required on average to store incoming packets or read outgoing packets in one switch-cycle. If all packets in a given switch-cycle are written or read to/from different memory-modules successfully then the memory-bandwidth requirement will be one. However, the lack of parallelization of memory-write or memory-read operations requires the architecture to increase the memory-bandwidth of the memory-modules to be able to write or read several packets in one switch-cycle. Furthermore, the worst-case memory-bandwidth requirement measures the number of memory-cycles required by packets to be written or read into/from memory-modules when the switching system is operating at full load (100%).

3.4.1 SW-Switch Architecture with Assignment Scheme-1

First, we evaluate the average memory-bandwidth requirement for the sliding-window (SW) switch architecture using *assignment scheme-1* to store incoming packets (Fig. 3.9) into memory-modules. We should keep in mind that the average memory-bandwidth requirement is measured at the WRITE stage, where contention occurs within SW architecture. The switch under evaluation is a NxN = 32x32 ports with a total memory of 2048 packets. There are three configurations mentioned above for the buffer accommodation; i.e. $m = 32$ with $\sigma = 64$, $m = 64$ with $\sigma = 32$, and $m = 128$ with $\sigma = 16$.

Fig. 3.9 shows that configuration-1, i.e. $m = 32$ with $\sigma = 64$, needs the highest average memory-bandwidth requirement due to insufficient number of memory-modules for parallel write operations. The $m = 2N$ (Configuration-2) provides the least average memory-bandwidth requirement in the switching system. Furthermore, SW architecture requires at least $m = 2N$ memory-modules (as mentioned in [42]) for an optimal performance of the switching system. Increasing the number of memory-modules, $m > 2N$ (configuration-3) causes a small increment in the average memory-bandwidth requirement for higher loads (> 80%) compared to configuration-2. This increment is produced by additional memory-write operations due to a higher throughput in the switching system as shown in Fig. 3.10

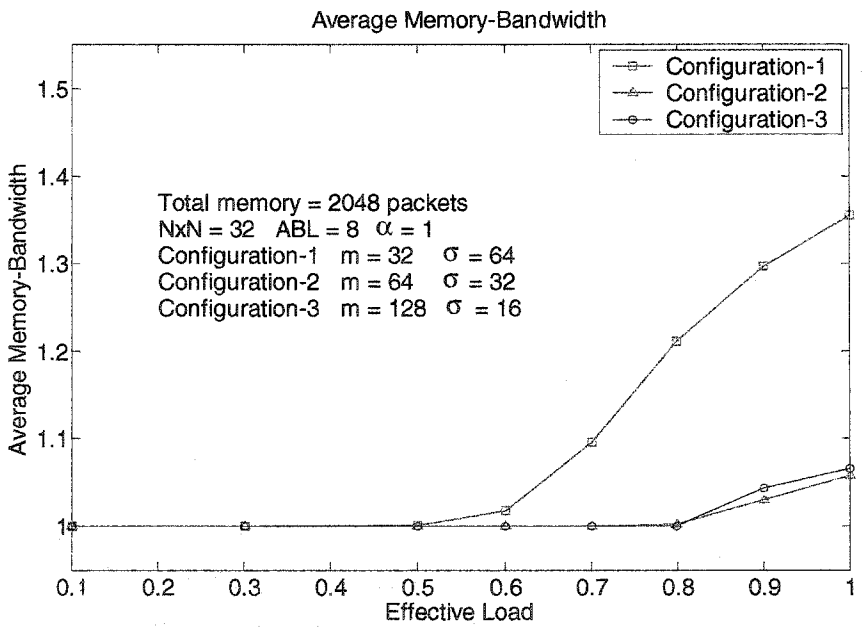


Fig. 3.9 Average memory-bandwidth for SW architecture using assignment scheme-1 to write data-packets into memory-modules

In *assignment scheme-1* the incoming packets requires one or more memory-cycles to be written to the memory-modules. Several write operations are performed

within each switch-cycle to solve contention when two or more packets are sent to the same memory-module. The worst-case memory-bandwidth requirement is illustrated in Table 3.1. That is the number of memory-cycles required by the packets to be written into memory-modules when the switching system is operating at the full load (100%). In Table 3.1, the percentage of packets are shown along with the required number of memory-cycles that it took to store them in the memory-modules of the switch operating at 100% load.

Load = 100%

	1 cycle	2 cycles	3 cycles	4 cycles	5 cycles
Configuration-1	50.43124%	41.36466%	7.76236%	0.43021%	0.01153%
Configuration-2	88.91764%	11.06017%	0.02215%	0.00003%	0%
Configuration-3	87.72566%	11.96270%	0.30964%	0.00200%	0%

Table 3.1 Worst-case memory-bandwidth requirement for packets to be written into memory-modules at 100% load in SW-switch architecture

Packets in configuration-1 require the greatest number of memory-cycles to be stored (5 cycles). It is obvious from Table 3.1 that more than 99% of the packets require no more than 3 memory-cycles in configuration-1. Increasing the number of memory-modules (configuration-2 and configuration-3) decreases the requirement (for 99% of the packets) to a maximum of 2 memory-cycles. Incrementing the number of memory-modules makes it easier to find available slots in the corresponding OSV arrays to designed different memory-modules to store incoming packets.

Fig. 3.10 shows the throughput performance for the SW-switch architecture with *assignment scheme-1* for above-mentioned three configurations. This experiment shows that the throughput is very similar for all these configurations. Though as the number of

memory-modules are greater, throughput has a marginal increase, i.e., configuration-3 (m = 128, σ = 16) shows the higher throughput.

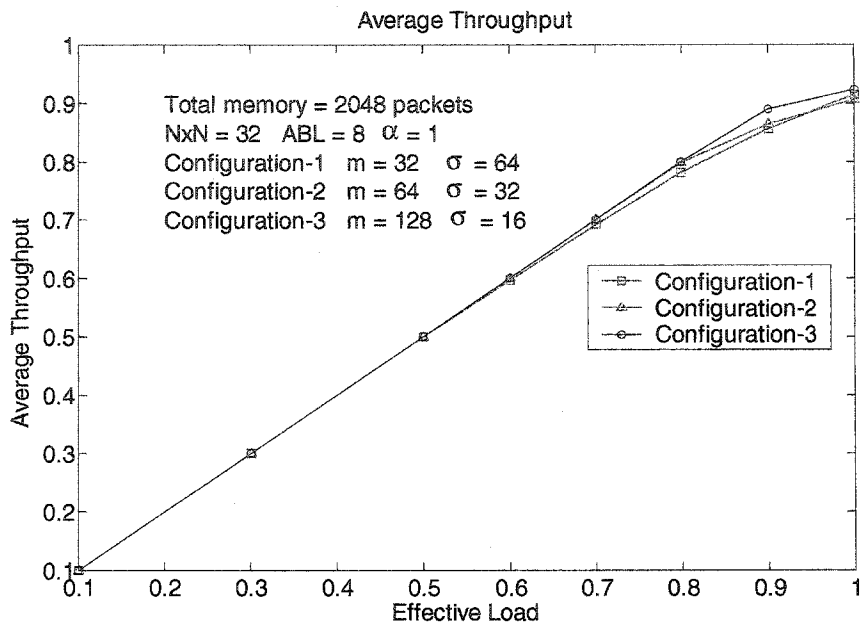


Fig. 3.10 Average throughput for SW architecture using assignment scheme-1 to write data-packets into memory-modules

For the throughput performance given for different configurations (Fig. 3.10), we also measure the corresponding packet-loss ratio (PLR) under identical traffic conditions. Fig. 3.11 shows that the packet-loss ratio (PLR) is significantly different for the various configurations considered in this experiment. The PLR decreases significantly as the number of memory-modules are increased for the same fixed total memory size deployed in the switching system. In order to realize smaller Packet-loss, it is preferred to have larger numbers of memory-modules (m) with smaller number of memory locations (σ) (configuration-3) as compared to having smaller number of memory-modules (m) with larger number of memory locations (σ) (configuration-1).

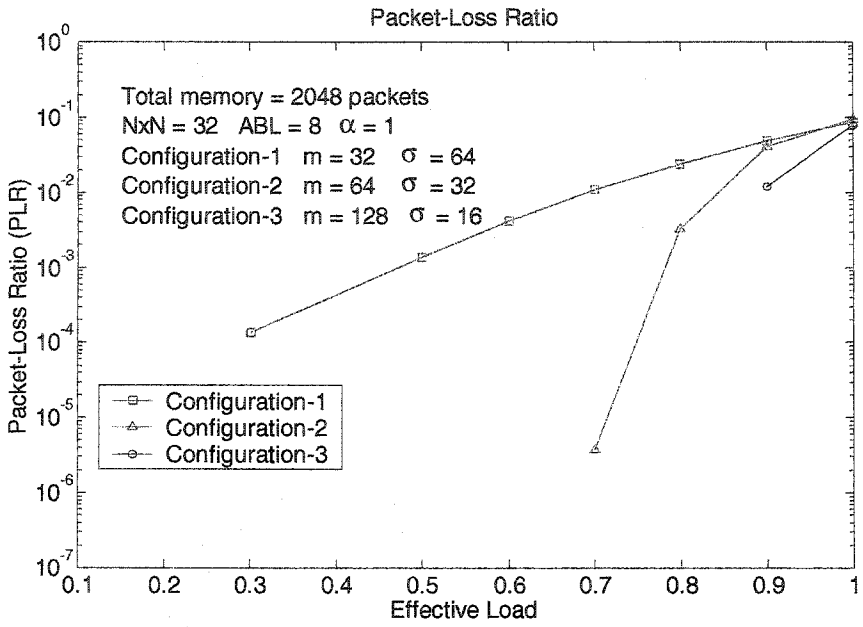


Fig. 3.11 PLR for SW architecture using assignment scheme-1 to write data-packets into memory-modules

3.4.2 SW-Switch Architecture with Assignment Scheme-2

Fig. 3.12 and Fig. 3.13 show the throughput performance and packet-loss respectively for the SW-switch architecture using *assignment scheme-2* for the designation of memory-modules to store incoming packets. Also three configurations are evaluated for the buffer accommodation, i.e. $m = 32$ with $\sigma = 64$; $m = 64$ with $\sigma = 32$; and $m = 128$ with $\sigma = 16$. Average throughput (Fig. 3.12) increases with greater number of memory-modules. Comfiguration-3 presents the highest throughput for the same fixed total memory-size (2,048 packets). We also measure the corresponding packet-loss ratio under identical traffic conditions using *assignment scheme-2*. Fig. 3.13 also shows that the packet-loss ratio (PLR) decreases significantly as the number of memory-modules deployed in the switch are increased while keeping the total memory-size fixed in the switching system.

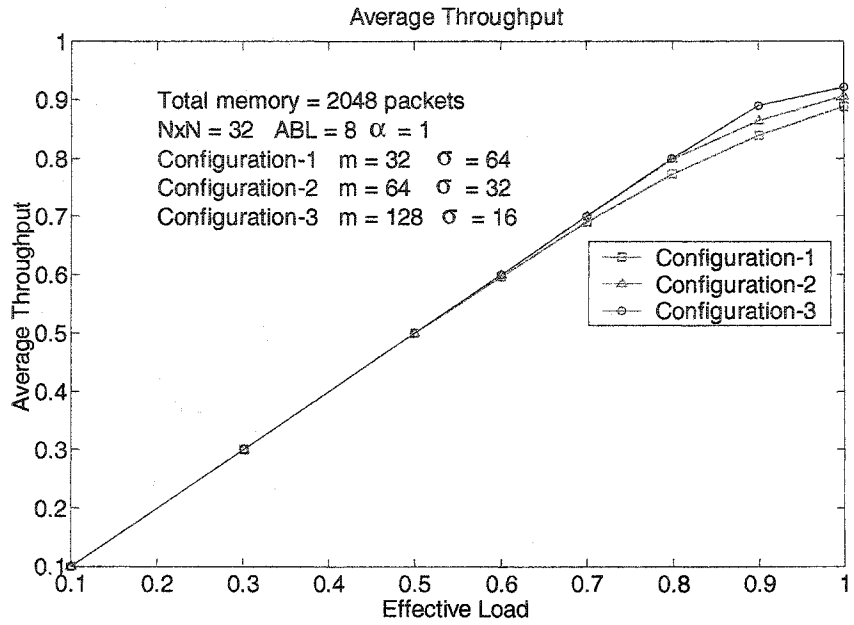


Fig. 3.12 Average throughput for SW architecture using assignment scheme-2 to write data-packets into memory-modules

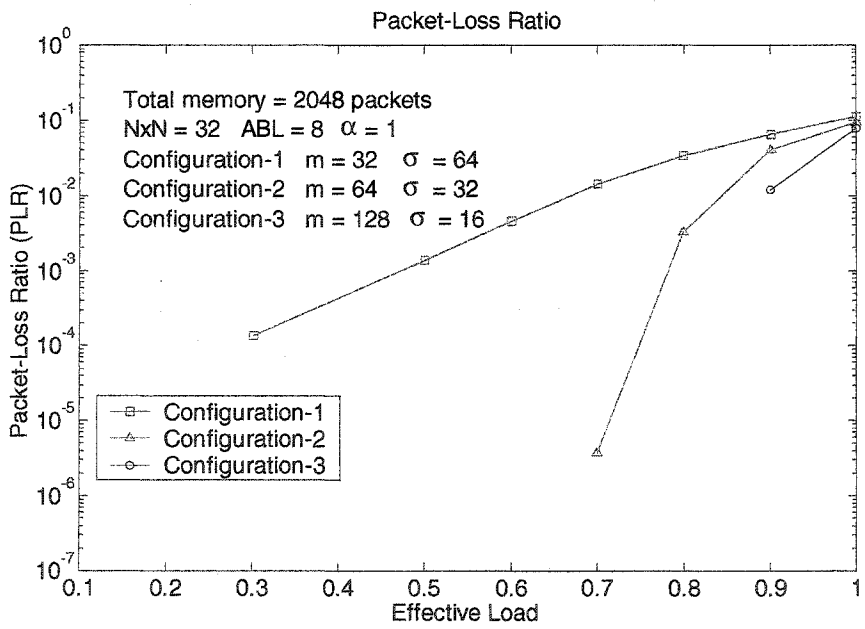


Fig. 3.13 PLR for SW architecture using assignment scheme-2 to write data-packets into memory-modules

Average memory-bandwidth requirement for the SW-switch architecture using *assignment scheme-2* is one for the three memory configurations. This is because of the fact that at all times; packets accepted into SW-switch architecture are successfully placed in different memory-modules of the switching system. Therefore, *assignment scheme-2* in the SW architecture provides the ideal memory-bandwidth requirement in which memory-speed is equal to the line-speed for the switching system.

Assignment scheme-2 requires a search in two arrays to find unique available slots that has not been assigned earlier to packets of the same switch-cycle. Thus, *assignment scheme-2* might seem more complex than *assignment scheme1* that only requires a search in one array to assign a memory-module. The implementation details of these two algorithms can be seen in [46]. As seen by results in Fig. 3.10, Fig. 3.11, Fig. 3.12, and Fig. 3.13, there is no significant difference in throughput performance and packet-loss between *assignment scheme-1* and *assignment scheme-2*. However, unlike *assignment scheme-1,* the *assignment scheme-2* provides a very good performance with the advantage of reduced memory-bandwidth requirement that could enable us to build very large-size switches.

3.4.3 Performance Comparison of SMB-Switch Architecture and SW-Switch Architecture

In this simulation experiment, the performance of SMB-switch architecture and SW-switch architecture are compared. A switch with NxN = 32x32 ports and total memory equal to 2,048 packets is used for evaluation. For the buffer configuration, the

configuration-2 is considered, i.e., $m = 64$, $\sigma = 32$, where m is the number of memory-modules and σ is the number of packet locations in each memory-module.

Fig. 3.14 shows the average memory-bandwidth requirement for SMB architecture and SW architecture. Memory-bandwidth for SMB architecture is evaluated at the READ stage in the memory-modules. In SMB architecture, two or more packet could need to be read out from the same memory-module during a given switch-cycle. Thus, memory-contention occurs during read operations, contrary to memory-contention in SW architecture that occurs during write operations.

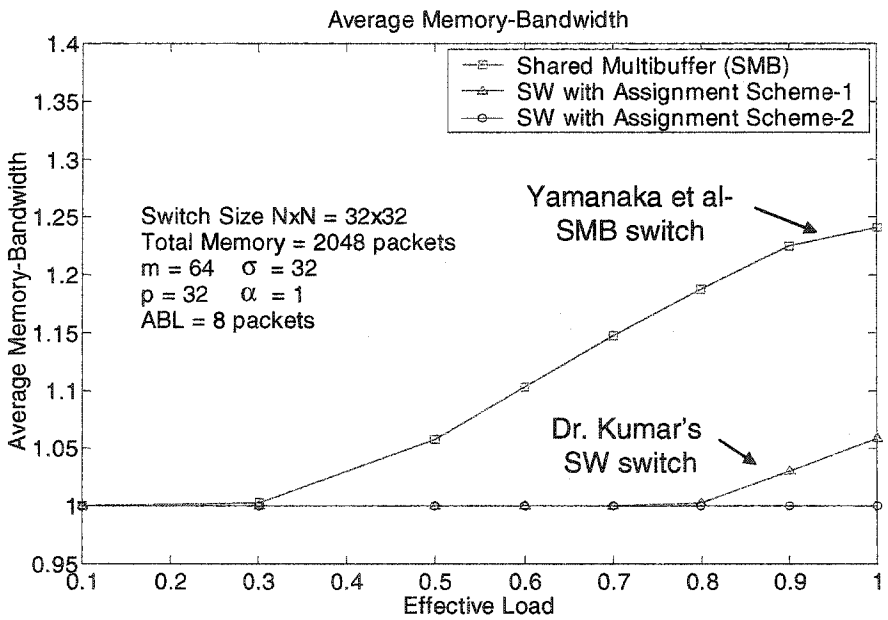


Fig. 3.14 Average memory-bandwidth evaluations of SMB and SW architectures at the READ and WRITE stages respectively

As shown, in the simulation results in Fig. 3.14, the average memory-bandwidth requirement for Yamanaka's SMB switch is much higher compared to that of Dr. Kumar's SW switch for a given bursty-traffic. High memory-bandwidth causes the memory access requirements become more stringent, which in turn limits the scalability

of the SMB-switching system. Memory-bandwidth requirement in SW architecture with *assignment scheme-1* increases only at higher loads (> 80%). However, the memory-bandwidth requirements for SMB architecture starts increasing at low loads (>30%) in Fig. 3.14.

The worst-case memory-bandwidth requirement for SMB architecture and SW architecture is presented in Table 3.2 for 100% load. SMB architecture requires up to seven memory-cycles to read all packets from the memory-modules. The maximum number of memory-cycles required in SW architecture with *assignment scheme-1* (to store packets) is four (Table 3.2). Although more than 99% of packets requires at most two memory-cycles to be written in memory-modules for SW architecture.

Load = 100%

	1 cycle	2 cycles	3 cycles	4 cycles	5 cycles	6 cycles	7 cycles
Shared Multibuffer	63.65900%	29.16323%	6.25252%	0.84300%	0.07670%	0.00543%	0.00012%
SW with Scheme-1	88.91764%	11.06017%	0.02215%	0.00003%	0%	0%	0%
SW with Scheme-2	100%	0%	0%	0%	0%	0%	0%

Table 3.2 Worst-case memory-bandwidth requirement for packets to be written or read into/from memory-modules at 100% load for SMB and SW architectures

SW architecture with *assignment scheme-2* presents a constant value of one in Fig. 3.14 and 100% packets requires only one memory-cycle (Table 3.2) to be written in memory-modules. It means the *assignment scheme-2* achieves 100% parallel write and read operations for memory-modules of the SW-switching system. *Assignment scheme-2* drops packets when available slots in the same position are not found in both OSV and *temp array* (other reasons of packets-loss were explained in section 3.3.1). At times, an available slot is found in the corresponding OSV, but that slot was used previously during the current switch-cycle (slot in *temp* array is different from zero). If no similar slots are available in both OSV and *temp array* during the search, packet is dropped (lack of

availability of slots usually happens at higher loads) and the throughput of the SW-switch architecture could be decreased. Though, SW-switch architecture with *assignment scheme-2* performs very well and its throughput performance is similar or better than other schemes as shown by Fig. 3.15 and Fig. 3.16

Fig. 3.15 shows average throughput for SMB architecture and SW architecture with unlimited memory-bandwidth resources for the switching system, even though this is not a realistic situation. Under these unrealistic conditions, SMB architecture presents higher throughput for higher loads (> 80%) compared to SW architecture. This higher throughput is explained due to the fact SMB architecture is less restrictive to assign a memory-module to receive a packet; and the least occupied memory-module is chosen to store an incoming packet. While in SW architecture, total memory is divided in σ OSV arrays; and a packet looks for space only in the corresponding OSV.

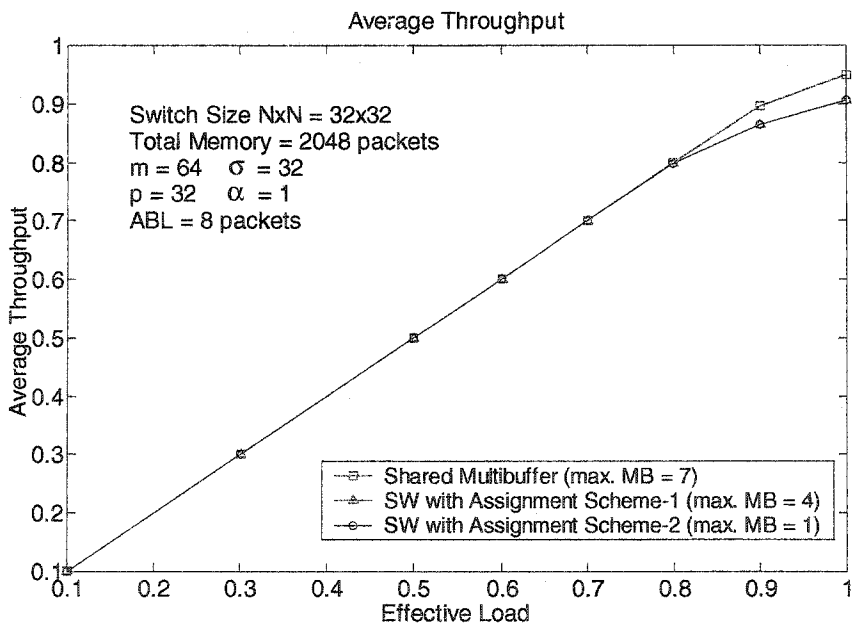


Fig. 3.15 Average throughput for SMB and SW architectures with unlimited memory-bandwidth requirement (maximum memory-bandwidth = 7)

High memory-bandwidth requirements, in a switch with parallel memory-modules, could become a bottleneck in design of high capacity routers and switches. An unlimited memory-bandwidth condition is not practical and scalability of the switch can become limited due to the finite memory-bandwidth of the system. Memory-bandwidth is generally finite and it imposes a physical restriction on the number of write/read operations that can be performed in a given switching system. Since it is not practical to build a memory-based system with no restrictions on the memory-bandwidth, we perform another experiment to evaluate both SMB and SW switch architecture of same size and traffic, under conditions of finite memory-bandwidth (MB = 1). That is the memory-speed is same as the line-speed of the switch.

When the memory-bandwidth requirement is limited to one, i.e. the memory-speed is same as the line-speed, then the throughput decreases drastically in Yamanaka's SMB switch compared to that of Dr. Kumar's SW switch of same switch size and memory space (Fig. 3.16). Packets are dropped when multiple packets try to access or depart from the same memory-module during a given switch-cycle, only one packet could be written and one packet could be read out from each memory-module during a switch-cycle (other reasons of packets-loss were explained in sections 3.2.1 and 3.3.1).

Performance evaluation shows that Dr. Kumar's SW switch with *assignment scheme-1* experiences a slight decrease on its average throughput at higher loads (> 80%) as a result of packets dropped due to contention in memory resources. A superior performance is demonstrated for Dr. Kumar's SW switch with *assignment scheme-2*. For the throughput performance given in Fig. 3.16, we also measure the corresponding packet-loss ratio (PLR) under identical traffic conditions. Fig. 3.17 shows that there is

significant difference in packet-loss ratio (PLR) between SMB architecture and SW architecture. Yamanaka's SMB switch has a very poor performance discarding packets at low loads (>30%) due to the limitation in memory-bandwidth (MB = 1)

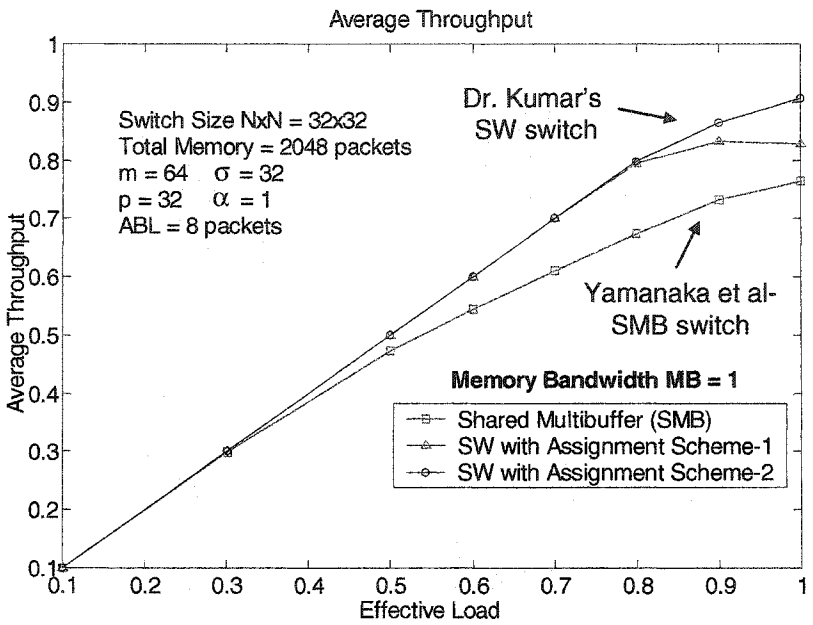


Fig. 3.16 Average throughput for SMB and SW architectures with memory-speed equal to the line-speed (memory-bandwidth = 1)

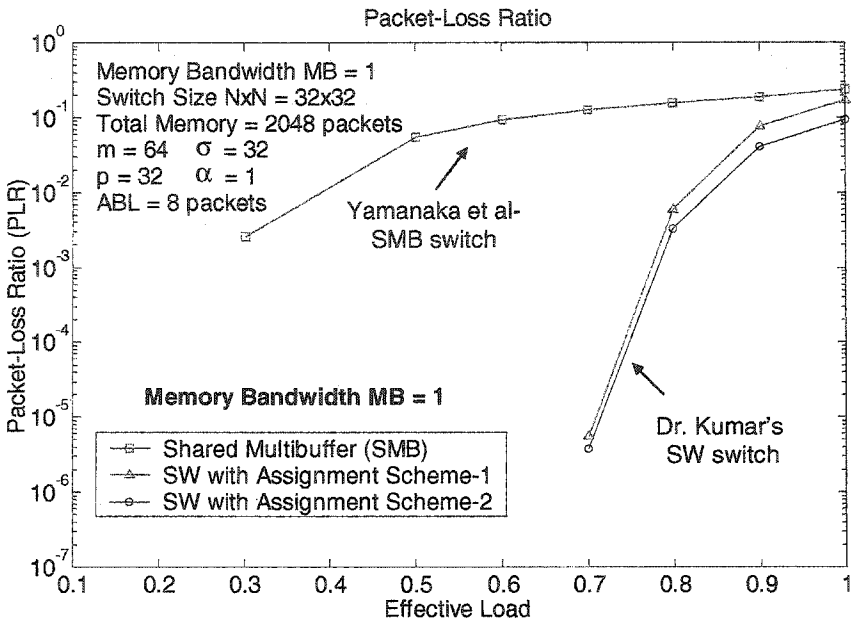


Fig. 3.17 Packet-loss ratio for SMB and SW architectures with memory-speed equal to the line-speed (memory-bandwidth = 1)

CHAPTER 4

PRIORITY SWITCHING FOR ARCHITECTURE WITH MULTIPLE SHARABLE MEMORY-MODULES

4.1 Introduction

Meeting quality-of-service (QoS) requirements for emerging real-time applications on the Internet imposes new challenges. These applications include real-time audio, video, and mission-critical financial and security data. Such real-time applications require bounded end-to-end delay, bounded packet-loss rates, and guaranteed bandwidth from the network. To address these issues, there are several approaches within the network to establish means to provide QoS to the various traffic classes. Resource reservation [47][48] is a useful method to allocate network resources along the data paths before performing the data transmission; admission control [49][50] can regulate the setup of new connections in order to make sure network spare capacity is always guaranteed for current and new users; differentiated services [51][52] is aimed at supporting service differentiation for aggregated traffic.

Priority switching in Internet routers and switches can provide differentiated service at the switch level to individual traffic classes. There are of two different types: time (or delay) priority and space (or loss) priority. Time priorities provide preferential service to the high priority class in order to control its end-to-end delay and jitter. Shared multibuffer architecture (SMB) is extended in [38] to support time priority by adding priority bits on its searchable address queue. Variations on iSLIP scheduling algorithm in [53] for input-queued switches are presented to support time priority at multiple priority levels. Also, a TF (threshold with feedback) scheduling algorithm is proposed in [54] with two priority levels. On the other hand, space priorities tend to provide preferential access to the memory space and minimize the packet loss in the high priority class. Pushout scheme [55][56] and dynamic queue length threshold [57] employ space priority for buffer management in a shared-memory switch with multiple priority classes.

SW-switch architecture employs feasible-size memory-modules in parallel that can achieve the necessary memory-bandwidth required in modern routers and switches [42] and overcome the memory speed bottleneck of traditional shared-memory switches. This chapter provides a mechanism for priority switching in the SW-switch architecture for two priority classes. Traffic contains packets with two priority levels, i.e., high priority packets and low priority packets. SW-switch architecture with priority offers to the high priority class a privileged allocation of memory resources (space priority) and their packets are processed and serviced earlier (time priority) than packets for the low priority class.

4.2 Sliding-Window

Global memory in SW-switch architecture [42] can be seen logically as an m x σ array (Fig. 4.1) where m is the total number of memory-modules and σ the number of memory locations in each module. Rows represent the memory-module (i parameter) and columns memory locations (j parameter). The vector formed by one column is called the output scan vector (OSV). Thus, j OSV denotes the vector assembled by j location in all m memory-modules. Incoming packets belonging to the same output queue are placed in consecutive OSV's. When a packet is placed on the last OSV ($j = \sigma$) the next packet belonging to the same output queue is stored on the first OSV ($j = 1$) in a circular fashion. Every recirculation of OSV's in global memory is designated by a different scan-plane (k parameter).

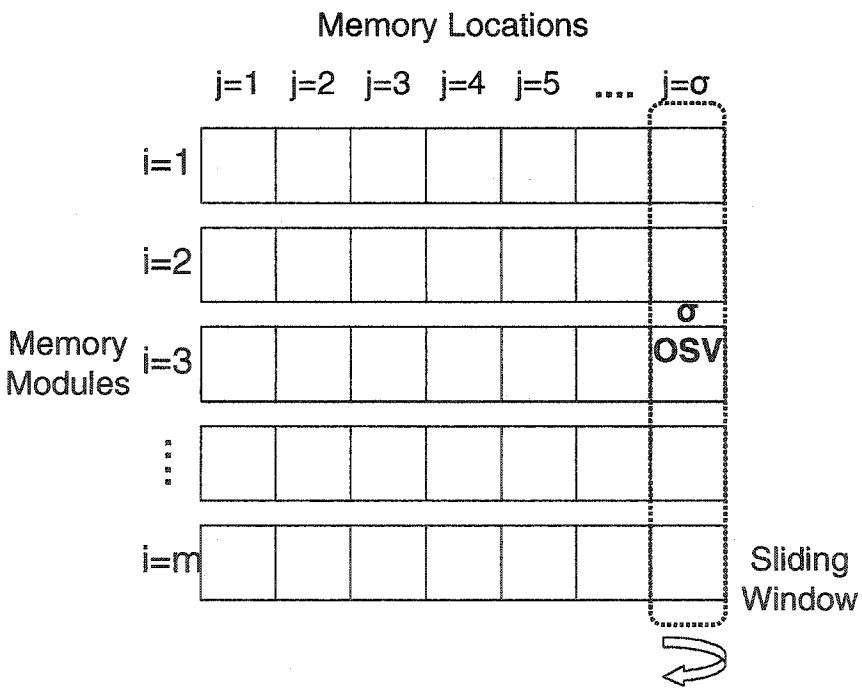


Fig. 4.1 Global memory in SW-switch architecture [42]

The sliding-window (SW) depicted in Fig. 4.1 scans global memory to determine outgoing packets that needs to be read out of memory. This sliding-window (SW) advances every switch cycle to traverse all memory locations in memory-modules and OSV in a circular manner. Also, departing packets are determined by the various scan-planes. The packets that correspond to the OSV on the current scan-plane (k parameter) are serviced. The sliding-window (SW) is specified by variable *SW.j* and *SW.k* for the corresponding OSV and scan-plane respectively. In this case variables *SW.j* and *SW.k* together represent a pointer to the current position of the sliding-window (SW) in the global memory-space Fig. 4.2 shows a flow diagram that computes the increment of variables *SW.j* and *SW.k* every switch cycle. Several functions in the SW-switch architecture with priority utilize the same operation for the increment of two variables. For that reason, we use ++mod (SW.j) and ++mod (SW.k) to denote the increment operation of *SW.j* and *SW.k* in the following sections. In Fig. 4.2, at step **200** *SW.j* variable is incremented by one every cycle; when $SW.j = \sigma$, the next increment will change $SW.j = 1$ (hence *SW.j* values range from 1 to σ and then again starts from 1 in a circular fashion). If $SW.j = 1$ in **202** then a new scan-plane is started. This operation can be performed by taking the mod for *SW.k* and p. Increment operation is shown in step **204**

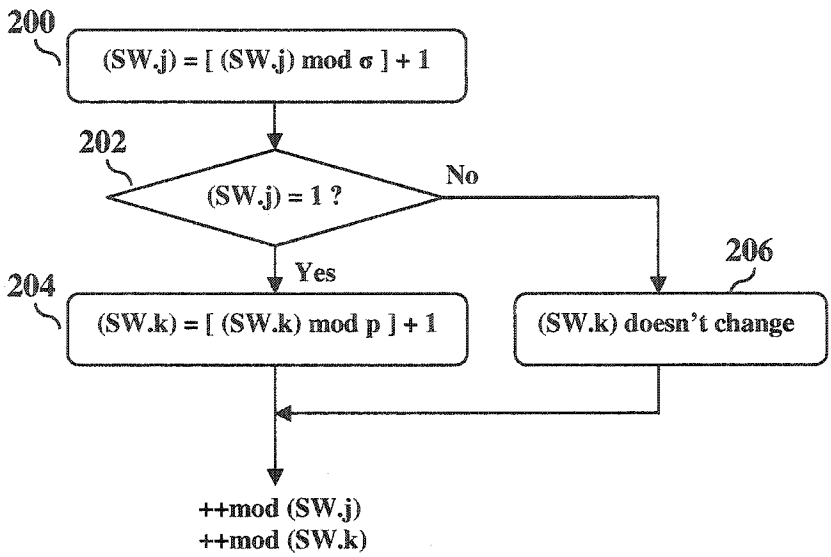


Fig. 4.2 Increment of variables SW.j and SW.k in the sliding-window

4.3 Admittance Policy for the SW-Switch Architecture with Priority

Fig. 4.3 shows the preliminary steps to admit packets into the SW-switch architecture with priority. For an NxN switch, up to N data packets could be received from the input lines every cycle; step **300** indicates that X packets are received in a given cycle. A packet is removed from the non-empty set X (step **302**) and its priority pr and output-port d is determined from packet header at step **304**. At step **306** queue length Q_d, for output-port d, total buffer space used at all memory-modules ΣQ_i, and dynamic threshold values DT are determined. Information collected in steps **304** and **306** is employed by the admittance policy at step **308** to determine if the incoming packet is dropped or accepted into the switching system. The admittance policy applied to packets at step **308** determines if a packet is eligible to be admitted to the switch. This process is repeated for all packets of set X until no packet is left.

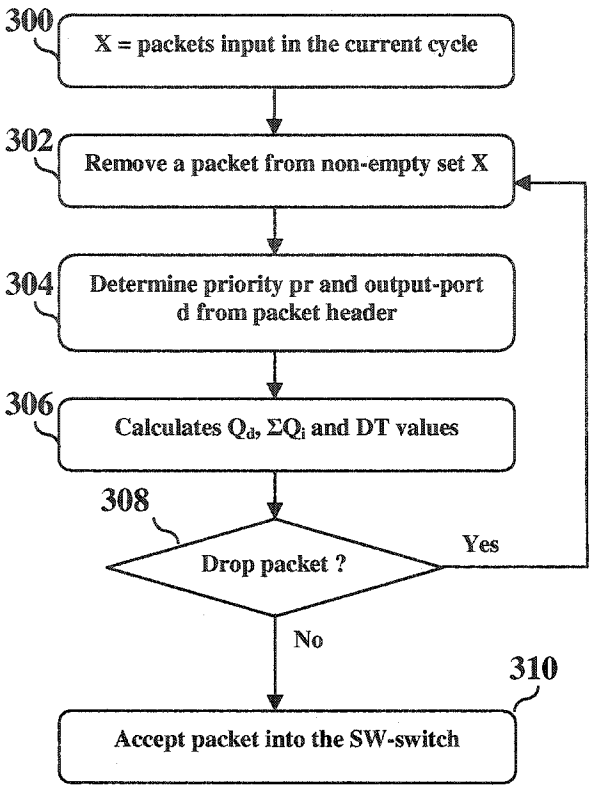


Fig. 4.3 Steps to admit packets into the SW-switch with priority

The admittance policy (step **308**) requires three conditions that should be satisfied in order to accept an incoming packet into the switching system.

- It is verified that there is available buffer space in the shared memory of the switch; summation of all queue lengths should be less than the total memory space at the SW-switch ($\Sigma Q_i < m \cdot \sigma$).
- Queue length for output-port d should be less than the maximum physical length $p \cdot \sigma$.
- Also the queue length for output-port d is regulated not to exceed a dynamic threshold value DT_{pr} which is different for each priority class pr

The admittance policy can be given as follows:

```
if ( ΣQi < m·σ ) and ( Qd < p·σ ) then

    if ( pr = H ) and ( Qd < DTH ) then
        Accept packet into the SW-switch

    elseif ( pr = L ) and ( Qd < DTL ) then
        Accept packet into the SW-switch

    else
        Drop packet
```

Fig. 4.4 Admittance policy for incoming packets

Fig. 4.4 presents the pseudo-code for the admittance policy in the SW-switch architecture with priority. First, it is verified that there is available buffer space ($\Sigma Q_i < m \cdot \sigma$) and that the queue length not exceed the maximum physical length ($Q_d < p \cdot \sigma$). Then depending on the priority class, dynamic threshold value DT is calculated for each priority class; DT_H for high priority and DT_L for low priority. If $m \cdot \sigma$ is the total memory space and $\Sigma Q_i(t)$ is the buffer space used by all ports at time t, the dynamic threshold value DT at time t is calculated as follow:

$DT_H(t) = \alpha_H \cdot (m \cdot \sigma - \Sigma Q_i(t))$ for high priority

$DT_L(t) = \alpha_L \cdot (m \cdot \sigma - \Sigma Q_i(t))$ for low priority

DT value is a function of the available buffer space and α is a proportionality constant where $\alpha_H > \alpha_L$. A packet destined to output-port d will be accepted at time t if its queue length Q_d is less than the DT value ($Q_d < DT$). A small α value will restrict the size of queue lengths. In contrast, a large α value will not set significant control in queue lengths and will permit some ports to monopolize the buffer-space. While any output queue can be formed by a combination of packets from the various priority classes;

incoming high priority packets with a greater α_H value will see an upper threshold value DT_H than low priority packets and the high priority class will have a higher probability to be accepted into SW-switch. As we shall see later, the appropriate setting of α_H and α_L will allows us to eliminate the packet-loss in the high priority class The dynamic queue length threshold scheme [30][57] provides the means to obtain a unique quality-of-service (QoS) for each priority class. Specifically, {all priorities--whole buffer capacity--all priorities} *AWA* scheme explained in [57] is employed for the management of the buffer space with two priority classes in the SW-switch architecture.

4.4 Determination of Self-Routing Parameters (i, j, k)

Parameter assignment circuit (PAC) in Fig. 4.5 produces self-routing tags *(i, j, k)* that are attached to incoming packets to self-propagate through the various stages of the SW-switch architecture with priority. PAC circuit uses a set of counters and tables to facilitate determination of self-routing parameters (*i, j, k*). To enable faster assignments, PAC circuit uses two separate processors that work in a pipeline fashion. *Processor-1* **500** obtains the destination port *d* and the priority level *pr* of incoming packet from packet's header and calculates *j* and *k* parameters. Thereafter, *processor-2* **530** computes *i* parameter based on availability of array locations. The sliding-window (SW) in Fig. 4.5 is denoted by counters *SW.j* and *SW.k* (twice by **510** and **540**). With every cycle, SW counters **510** and **540** update their value independently according to the ++mod () operation described previously.

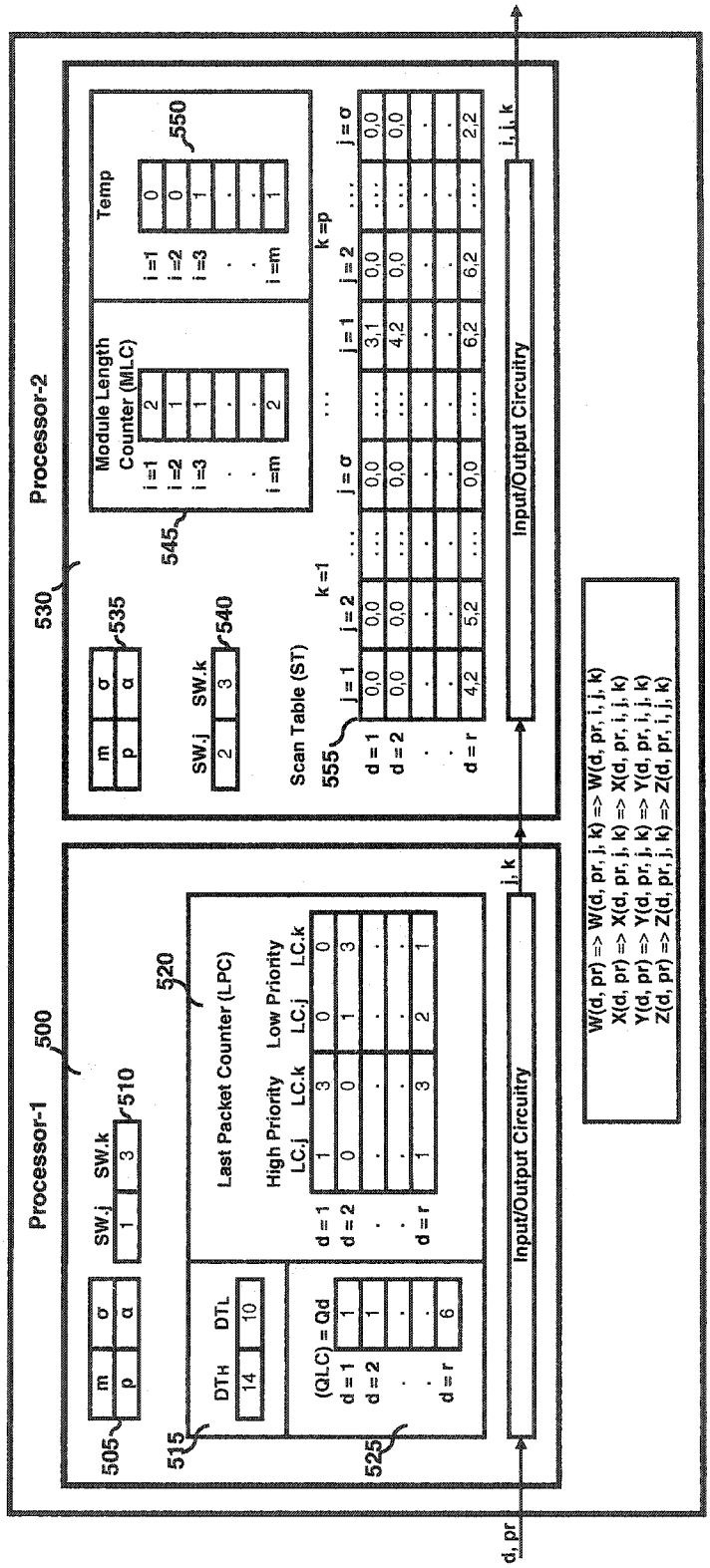


Fig. 4.5 Parameter assignment circuit that produce self-routing tag (i, j, k)

For *processor-1* in Fig. 4.5, queue length counter (QLC) **525** maintains queue lengths for all output-ports. Counter **515** compute dynamic threshold (DT) values for each priority class. These DT_{pr} values in **515** are an upper limit to lengths of output queues for each priority class. In addition last packet counter (LPC) **520** stores j and k parameters assigned to the last packet accepted into the SW-switch for each output-port d and priority class *pr*. In *processor-2* of Fig. 4.5, the number of packets stored at each memory-module is kept at module length counter (MLC) **545**. Also *temp* counter **550** maintains which memory-modules are selected to store incoming packets in the same cycle. This *temp* counter **550** allows us to have 100% parallel WRITE operations in memory-modules. Counters MLC and *temp* are used to determine i parameter.

Scan Table (ST) **555** holds i parameter and priority level *pr* for every packet accepted into the SW-switch. Size of table ST **555** depends on the switch size and the number of scan-planes used to force a sharing scheme. The number of rows is equal to the switch size N and the number of columns is equal to the maximum queue length permitted $p \cdot \sigma$. A shifting operation is done at ST table **555** every time a high priority packet is accepted and the corresponding packet information placed in ST table **555**. Information for low priority packets until that time in ST table **555** is shifted in order to have information for high priority packets in consecutive slots at ST table **555**. As a result, high priority packets could be scanned by the sliding-window (SW) and serviced by the switching system ahead of time than low priority packets.

4.4.1 Determination of Parameters (j, k)

Depending on the priority class, the *j* and *k* parameters are calculated differently. Fig. 4.6 shows flow diagram to calculate *j* and *k* parameters for low priority packets. Queue length Q_d in QLC **525** (Fig. 4.5) is incremented by one (step **600**) to account for incoming packet at output-port *d*. At steps **602** and **604** $(LC.j)_{pr,d}$ counters in LPC **520** (Fig. 4.5) are checked to know if there are packets inside memory-modules. If $(LC.j)_{pr,d}$ is different from zero that means, there are packets designed to output-port *d* and priority level *pr* (*pr* = *L* for low priority and *pr* = *H* for high priority).

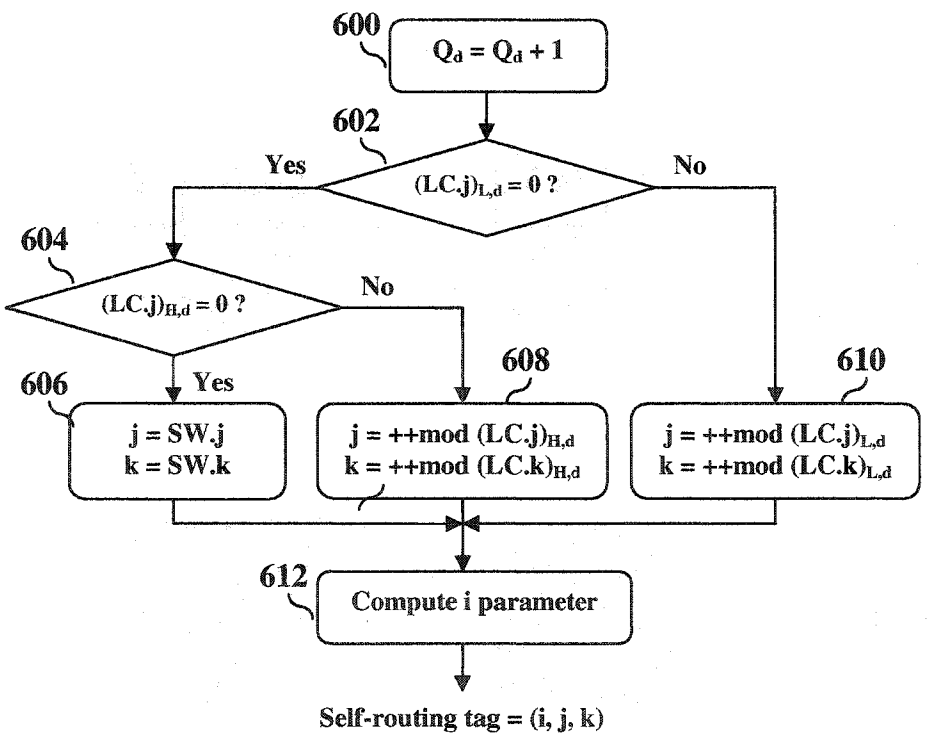


Fig. 4.6 Flow diagram to calculate j and k parameters for low priority class

When $(LC.j)_{L,d}$ and $(LC.j)_{H,d}$ are zero there are no packets for output-port *d* inside buffer space. Hence the new arrival is the first and should not wait inside memory. At

step **606** values assigned as j and k parameters are $SW.j$ and $SW.k$ values (**510** in Fig. 4.5). Consequently, the incoming packet is placed in the current location of the sliding-window (SW) and will be serviced in the current cycle. If $(LC.j)_{L,d}$ and/or $(LC.j)_{H,d}$ are different from zero that means there are packets inside buffer space for output-port d and parameters assigned to the incoming packet follows the first-in first-out (FIFO) order maintained within each priority class. Therefore, if there are only high priority packets inside memory then j and k parameters are calculated in step **608** applying ++mod () operation over $(LC.j)_{H,d}$ and $(LC.k)_{H,d}$. These $(LC.j)_{H,d}$ and $(LC.k)_{H,d}$ variables correspond to the j and k parameters assigned to the last high priority packet accepted. Similarly, if there are low priority packets inside memory then j and k parameters for incoming packet are calculated in step **610** applying ++mod () operation over $(LC.j)_{L,d}$ and $(LC.k)_{L,d}$; i.e., variables that correspond to the j and k parameters assigned to the last low priority packet admitted into memory. i parameter is calculated at step **612**. The procedure to calculate this parameter will be explained later. Finally the self-routing tag (i, j, k) is obtained at the end of the flow diagram.

Flow diagram to calculate j and k parameters for high priority packets is shown in Fig. 4.7. Queue length Q_d in QLC **525** (Fig. 4.5) for output-port d is incremented by one at step **700** to account for incoming packet at output-port d. At step **702**, if $(LC.j)_{H,d} = 0$ in LPC **520** (Fig. 4.5) then there is no high priority packets inside memory and j and k parameters for incoming packet are equal to $SW.j$ and $SW.k$ values (**510** in Fig. 4.5) at step **704**. That is incoming packet will be serviced in the current cycle because there are no other packets waiting in the queue line for output-port d to be read out from memory-modules. If $(LC.j)_{H,d} \neq 0$ there are high priority packets inside memory then j and k

parameters are calculated in step **706** applying ++mod () operation over $(LC.j)_{H,d}$ and $(LC.k)_{H,d}$ variables.

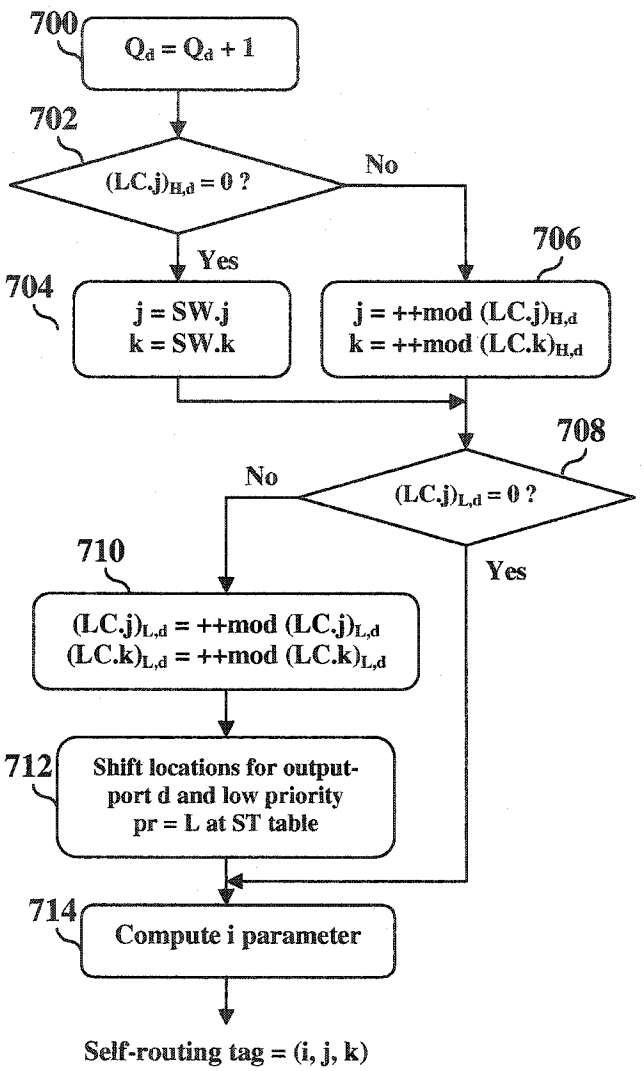


Fig. 4.7 Flow diagram to calculate j and k parameters for high priority class

Once *j* and *k* parameters have been calculated for incoming packet, information for low priority class (if any) already present in table and counters should be shifted or updated in order to incorporate information from the incoming high priority packet. For that reason, if there are low priority packets inside memory in step **708**, i.e., $(LC.j)_{L,d} \neq 0$,

steps **710** and **712** should be executed. Consequently is applied ++mod () operation over $(LC.j)_{L,d}$ and $(LC.k)_{L,d}$ variables (step **710**). Also, locations at ST table **555** (Fig. 4.5) for output-port d and low priority $pr = L$ are shifted (step **712**). If $(LC.j)_{L,d} = 0$ then no action is necessary at ST table **555** and $(LC.j)_{L,d}$ and $(LC.k)_{L,d}$ variables. Finally, i parameter is calculated (step **714**) and the self-routing tag (i, j, k) is obtained at the end of the flow diagram.

4.4.2 Determination of i-Parameter

The scheme used to assign i parameter to packets is depicted in Fig. 4.8. A search is done through module length counter (MLC) **545** and *temp* **550** arrays (Fig. 4.5) to determine i parameter that denotes the memory-module where packet will be stored.

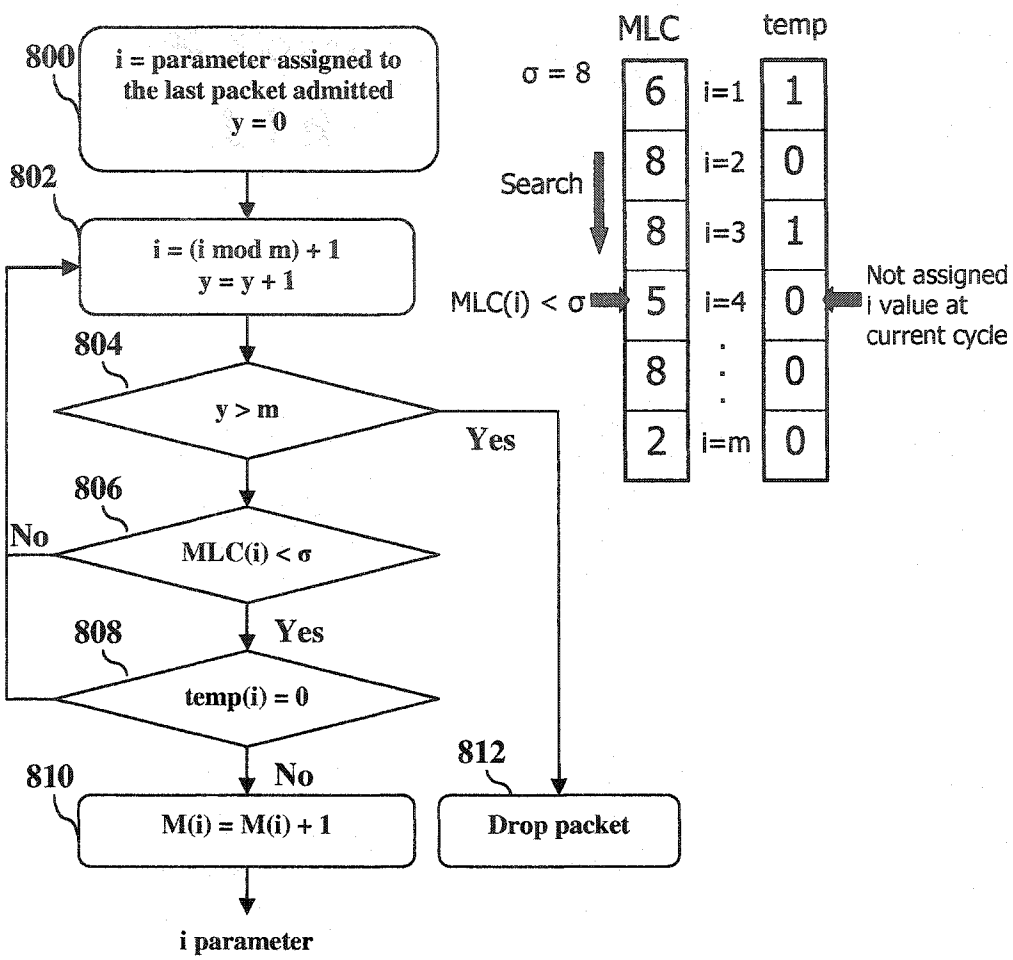


Fig. 4.8 Scheme to assign i parameter

At the start, *i* variable has the value assigned to the last packet admitted and an auxiliary variable *y* is initialized to zero (step **800**). Subsequently *i* variable is incremented by one in a mod fashion to initiate the search at step **802**. Also *y* variable is incremented by one to count the number of searches done. The search for *i* parameter is done through module length counter (MLC) **545** and *temp* **550** arrays. MLC **545** counter holds the number of packets written to each memory-module, thus MLC (*i*) count should be less than the maximum packet capacity σ in a memory-module. Array *temp(i)* equals to zero indicates that this *i* slot is available and it was not previously assigned to a packet

in the current cycle. *Temp* array **550** is used to make sure that only one packet will be written to any memory-module within a cycle. Therefore, SW-switch architecture with priority allows 100% parallel WRITE operations to the memory-modules. When conditions $M(i) < \sigma$ and $temp(i) = 0$ in steps **806** and **808** respectively are satisfied, then the current i value is assigned as the i parameter. At step **810**, MLC counter **545** is incremented by one to account for packet that will be written into i memory-module. If search for i parameter has been done for all m locations in MLC **545** and *temp* **550** arrays and any condition in steps **806** or **808** was not met then packet is dropped at step **812**.

4.5 Memory Controllers

Memory-modules $\mathbf{40_1}$, $\mathbf{40_2}$,..., $\mathbf{40_m}$ in SW architecture are physically separate but logically linked to provide a shared-memory space among input and output-ports. Memory controllers $\mathbf{50_1}$, $\mathbf{50_2}$,..., $\mathbf{50_m}$ are responsible to compute addresses to write and read packets from memory-modules $\mathbf{40_1}$, $\mathbf{40_2}$,..., $\mathbf{40_m}$. Each memory-module has its local controller that works with local information and self-routing tag (i, j, k) attached to packets. Also all memory controllers are connected to a high speed bus to provide a means of communication between them. Fig. 4.9 shows the detailed structure of a memory controller **50** to illustrate the different tasks performed by it. Input module **30** receive packet with self-routing tag (i, j, k) and header parameters (d, pr). Information (d, pr, j, k) is sent to the memory controller **50** and data-packet is written to the memory-module **40**.

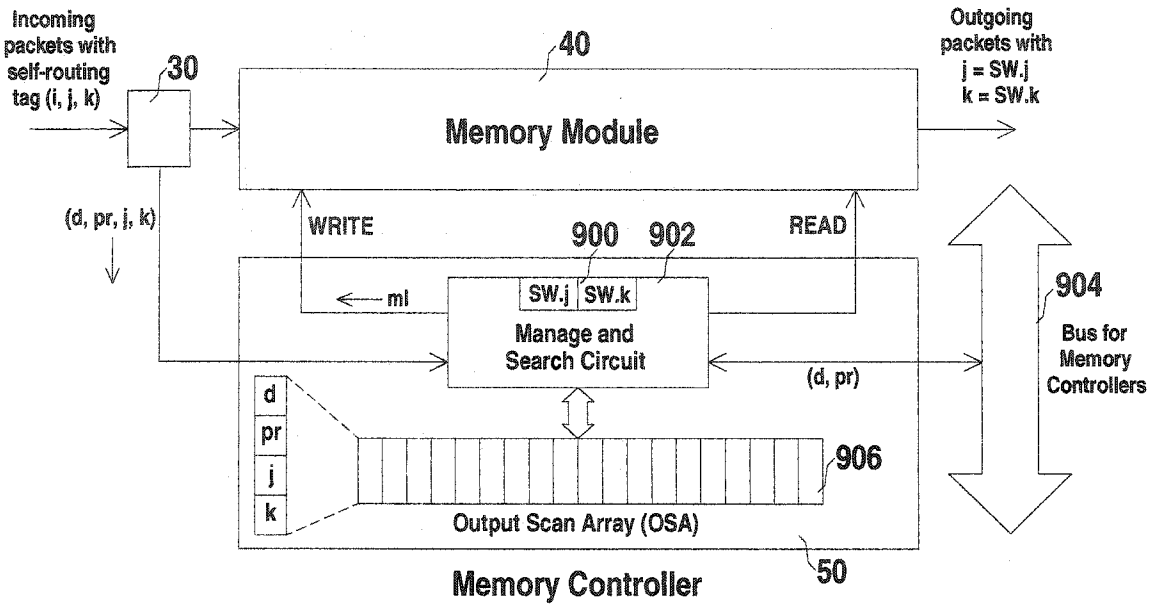


Fig. 4.9 Memory controller for memory-module

Manage and search circuit (MSC) **902** places (*d, pr, j, k*) values in an available memory location (*ml*) of the output scan array (OSA) **906** which has σ locations. Also data-packet is written into *ml* location of memory-module **40**. Counters **900** *SW.j* and *SW.k* update its value according to the ++mod () operation described previously. Information in OSA array **906** help to determine which packet(s) needs to be read out from memory-module **40**. A search is done every cycle in OSA array **906** to find locations with $j = SW.j$ and $k = SW.k$ that correspond to packets that need to be read out from memory-module **40**. High speed bus **904** enables memory controllers to communicate with each other and exchange updates at OSA array **906**. MSC circuit **902** transmits or receives through bus **904** *d* value which indicates that locations in OSA array **906** need to be updated. Every time a high priority packet is received, ++mod () operation should be applied over *j* and *k* parameters for locations in OSA array **906** that correspond

to output-port d and $pr = L$. This ++mod () operation over j and k parameters is done in all memory controllers $\mathbf{50_1}$, $\mathbf{50_2}$,..., $\mathbf{50_m}$. Hence, multiple memory controllers from $\mathbf{50_1}$, $\mathbf{50_2}$,..., $\mathbf{50_m}$ could request update for j and k in OSA array **906** for the duration of a cycle.

4.5.1 WRITE Stage

WRITE operation is shown by a flow chart in Fig. 4.10. That is, steps necessary to write data packet into memory-modules $\mathbf{40_1}$, $\mathbf{40_2}$,..., $\mathbf{40_m}$. Initially, Routing tag (i, j, k) and header parameters (d, pr) from incoming packet are sent to memory controller **50** by the input module **30** (Fig. 4.9). At step **1000**, MSC circuit **902** obtains (d, pr, j, k).parameters.

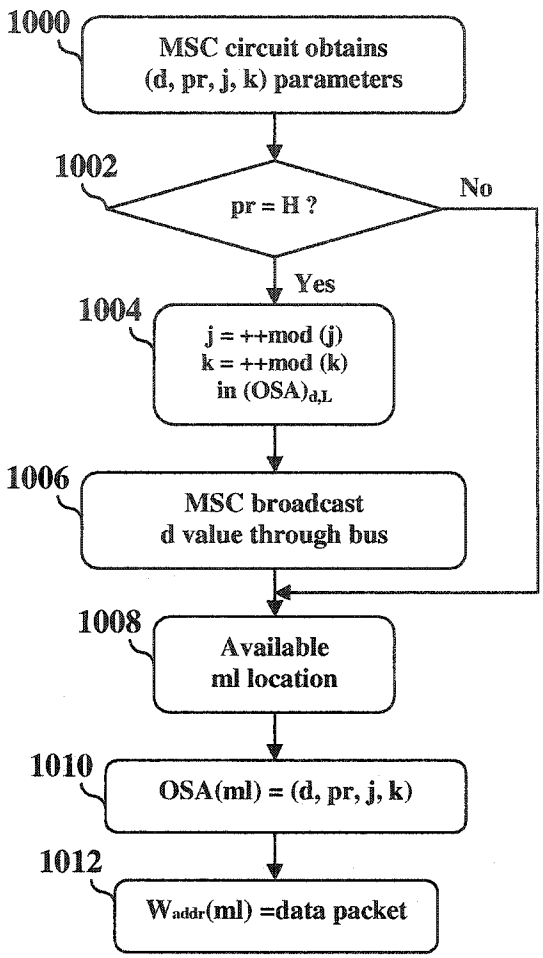


Fig. 4.10 WRITE operation of packets to memory-modules

If *pr* = *H* in step **1002**, then it means that a high priority packet is received and it requires the information to be modified for low priority packets in order to place high priority packets in the queue. Consequently, ++mod () operation is applied over *j* and *k* parameters at each location of OSA array **906** designed to output-port *d* and low priority (*pr* = *L*) at step **1004**. Also MSC circuit **900** broadcasts *d* value to all memory controllers $\mathbf{50_1}$, $\mathbf{50_2}$,..., $\mathbf{50_m}$ through bus **904** in step **1006**. Whichever memory controller $\mathbf{50_1}$, $\mathbf{50_2}$,..., $\mathbf{50_m}$ receiving *d* value through bus **904** from other memory controller should complete step **1004**. Thus, multiple locations in OSA array **906** for various output-ports should be

updated according to step **1004** every cycle. If $pr \neq H$ (low priority packet) steps **1004** and **1006** are skipped and flow diagram continue in step **1008**. An available *ml* location is found in OSA **906** at step **1008**. Subsequently (*d, pr, j, k*) values are stored in *ml* location of OSA **906** in step **1010**. Also data packet is written in *ml* location of memory-module **40** at step **1012**. In the WRITE stage packets are written in parallel to memory-modules $\mathbf{40_1}$, $\mathbf{40_2}$,..., $\mathbf{40_m}$. SW-switch architecture with priority is able to complete 100% parallel write operations every cycle. Hence, the write speed in memory-modules $\mathbf{40_1}$, $\mathbf{40_2}$,..., $\mathbf{40_m}$ is equal to the input line-speed.

4.5.2 READ Stage

Fig. 4.11 shows READ operation of packets from memory-modules $\mathbf{40_1}$, $\mathbf{40_2}$,..., $\mathbf{40_m}$. Every cycle counters **900** *SW.j* and *SW.k* are incremented according with ++mod () operation in order to move the position of the sliding-window to next OSV inside memory space (step **1100**). Packets inside the sliding-window (SW) need to be serviced and read out from memory-modules $\mathbf{50_1}$, $\mathbf{50_2}$,..., $\mathbf{50_m}$ during the current cycle.

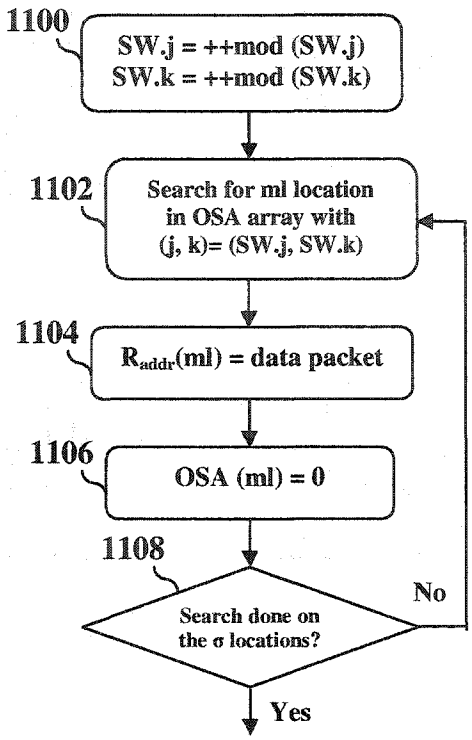


Fig. 4.11 READ operation of packets from memory-modules

Therefore, MSC circuit **902** search for *ml* location in OSA array **906** (Fig. 4.9) with $j = SW.j$ and $k = SW.k$ (step **1102**). Locations in OSA array **906** with these j and k parameters holds information of packets that needs to be output in the current cycle. Thus, packet with *ml* location matching $j = SW.j$ and $k = SW.k$ is read out from memory-module **40** (step **1104**). As well, at step **1106** the *ml* location is set to zero in OSA array **906** to free the memory location as a result of packet read out of the switch. In step **1108** flow diagram loops back to sequentially repeat steps **1102** through **1106** until the search function is completed in all σ locations of OSA array **906**. SW-switch architecture with priority could need several packets to be read out from a memory-module **40** within a cycle. Speedup of memory-modules $\mathbf{40}_1$, $\mathbf{40}_2$,..., $\mathbf{40}_m$ at READ stage is necessary to solve

contention when multiple packets are output from the same memory-module during the same cycle.

4.6 Illustration of the Sliding-Window Switch with Priority

Fig. 4.12 shows a 4x4 SW-switch with priority and its configuration. The number of memory-modules (m=6) is greater than the switch size (*NxN*=4x4), which helps to increase parallel operations at the READ stage and reduce the memory-bandwidth requirement. The maximum number of packet that can be stored at a memory-module are $\sigma = 4$. Thus, total memory space is $m \cdot \sigma = 24$ packets. Also the number of scan-planes are $p = 3$, that allocate a maximum physical queue-length for any output-port of $p \cdot \sigma = 12$ packets, so the maximum queue length $p \cdot \sigma$ forces a partial sharing scheme for the switch, i.e., a very active output-port could take up to half of the total memory space.

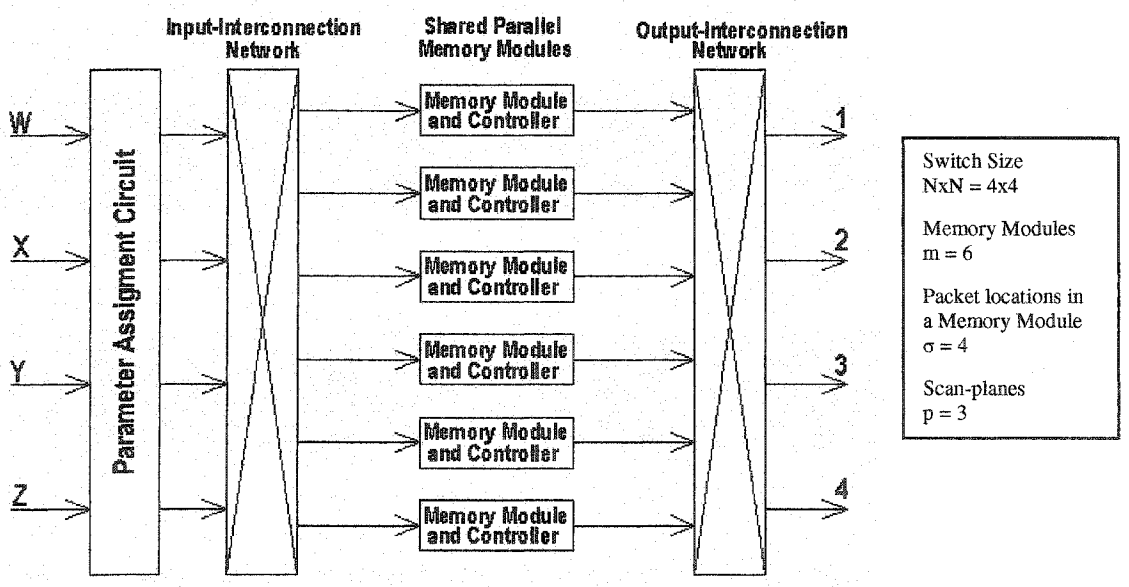


Fig. 4.12 Example of a 4x4 SW-switch with priority and its configuration

Dynamic threshold (DT) scheme [30][57] is used to avoid that a few output-ports could monopolize the use of memory space. In this scheme, values for the proportionality constant are α_H=1.0 for high priority packets and α_L = 0.7 for low priority. These values provide a higher probability for high priority packets to be admitted into the switch compared to low priority packets.

Fig. 4.13 presents packets received and output from the 4x4 switch system of Fig. 4.12 for up to 16 pipeline cycles. Input-ports are denoted by *W, X, Y,* and *Z* respectively. Each packet received is denoted by destination port and priority level. For instance, *3L* at input-port *W* and cycle-2 represents a packet designed to output-port 3 and low priority *L*. Also in Fig. 4.13 output-ports are designed by 1, 2, 3, and 4 respectively. Packets read out from the switch are denoted by the port of entry and the cycle when they entered the switch. For example, *X1* at output-port 4 represents a packet that entered by input line *X* at cycle-1.

Pipeline Cycle	Input W	Input X	Input Y	Input Z	Pointer SW(j,k)	Output 1	Output 2	Output 3	Output 4
1	1L	4H	4L	4H					
2	3L	2L	3L	3H					
3	3H	2H	4L	3H					
4	1L	4H	4H	4L	(1,1)				
5	1L	4L	4L	2L	(2,1)	W1			X1
6	3H	3H	2L	1H	(3,1)		X2	Z2	Z1
7	4H	4L	2H	2L	(4,1)		X3	W3	Y1
8	3L	1H	3H	2H	(1,2)	W4		Z3	X4
9	1H	4L	4L	4H	(2,2)	W5	Z5	W2	Y4
10	4H	4H	4L	4L	(3,2)	Z6	Y6	W6	Y3
11	4H	4L	4H	4L	(4,2)		Y7	X6	W7
12	4L	4L	4H	4H	(1,3)	X8	Z8	Y8	Z4
13					(2,3)	W9	Z7	Y2	Z9
14					(3,3)			W8	W10
15					(4,3)				X10
16					(1,1)				W11

Input-ports
{W, X, Y, Z}

Output-ports
d = {1, 2, 3, 4}

Priority Level
L = low
H = high

Fig. 4.13 Packet streams received and output in a 4x4 SW-switch with priority

Packets pass through the SW-switch until the fifth cycle because the switching operation is partitioned in 5 pipelined stages. Besides pointer *SW(j, k)* shows in Fig. 4.13 is for the WRITE stage (fourth pipeline stage). Packets with high priority are favored to pass first through the switching system. For example, in cycle-1 three packets went into the switch for $d = 4$, packets *X1* and *Z1* with high priority are read out first (cycles 5 and 6) and then packet *Y1* with low priority (cycle 7).

Fig. 4.14 shows processor-1 from Fig. 4.5 in order to explain the admittance policy for incoming packets at the SW-switch with priority. At the beginning of cycle 12 counters have the values show in Fig. 4.14. First it is verified the memory space used not exceed the total capacity ($\Sigma Q_i = 10$) < ($m \cdot \sigma = 24$) and the maximum queue length limit permitted ($Q_d = 10$) < ($p \cdot \sigma = 12$). Dynamic thresholds values are DT1 = 14 and DT2 = 10 for high priority and low priority respectively. Packet 4L from input-port *W* at cycle 12 is designed for output-port $d = 4$ and low priority, queue length for d = 4 is compared with the threshold for low priority ($Q_d = 10$) < (DT2 = 10) which is false and it causes packet 4L to be dropped.

SW.j	SW.k
4	3

DT1	DT2
14	10

(QLC) = Qd	
d = 1	0
d = 2	0
d = 3	0
d = 4	10

Last Packet Counter (LPC)

	High Priority		Low Priority	
	LC.j	LC.k	LC.j	LC.k
d = 1	0	0	1	1
d = 2	0	0	0	0
d = 3	0	0	0	0
d = 4	1	1	1	3

Fig. 4.14 Counters for processor-1

Because packet 4*L* from input-port *W* at cycle 12 was dropped, no update is necessary in counters at processor-1. Next packet *4L* from input-port *X* at cycle 12 is similar to last packet dropped. For that reason counter values in processor-1 are identical, therefore packet is processed exactly the same and it is also dropped. For packet 4H from input-port Y at cycle 12 the priority level is different, i.e., H = high priority. Counters in processor-1 remain with the same values, but now when compared queue length for d = 4 with the threshold for high priority (Qd = 10) < (TH2 =14) which is true and that let packet 4H to be accepted. Thereafter counters in processor-1 are updated due to packet admitted into the SW-switch.

4.7 Simulation Results

To study performance of the SW-switch architecture with priority, a bursty-traffic is generated using a two state ON–OFF model [7][42], i.e., by alternating a geometrically distributed period during which no arrivals occur (idle period), by a geometrically distributed period during which arrivals occur (active period) in a Bernoulli fashion and vice versa. Each active period generates a burst of data-packets destined to the same output-port and belonging to the same priority class. When bursts of packets are generated, they have equal probability to be destined to any of the output-ports. Traffic contains 20% high priority packets and 80% low priority packets in this simulation study. Every input-port is connected to an independent bursty source with average burst length (ABL) = 8 packets. In this simulation study evaluation is done for a packet switch with NxN = 32x32 ports with a total memory B = 512 packets. Total memory is formed by m = 32 memory-modules; each memory-module can store up to σ = 16 packets. Partial

sharing of the buffer space is permitted among ports with the number of scan-planes set to $p = 16$; this value restricts the allocation of buffer space to a maximum of half of the total memory to any output-port.

The dynamic queue length threshold scheme [30][57] is used to regulate the sharing of the buffer space among ports. Specifically, {all priorities--whole buffer capacity--all priorities} *AWA* scheme explained in [57] is employed for the management of the buffer space with two priority classes. In order to provide high priority packets preferential access to the memory resources, value for α constant is greater for high priority than for low priority. As a result, dynamic threshold (DT) value is allowed to be greater for that incoming high priority packets and queue lengths are permitted to grow bigger in size than for incoming low priority packets. Also i parameter is computed first for incoming high priority packets than for low priority packets in order to increment the chances of finding available slots in OSV and *temp* arrays; reducing the probability of high priority packets being dropped in the WRITE stage of the switch. Output queues may contain a mix of high and low priority packets but first-in first-out (FIFO) order is maintained within each priority class. At each output queue, high priority packets are serviced first and when only low priority packets remain in the queue these packets are processed.

Fig. 4.15 and Fig. 4.16 show packet-loss ratio (PLR) in SW-switch with priority for the two priority classes. The setting of α constant can produce different packet loss within each priority class. In Fig. 4.15 values of $\alpha_L = 0.8$ for low priority and $\alpha_H = 2.0$ for high priority produces a very low PLR at very high loads ($\geq$ 90%) for the high priority class.

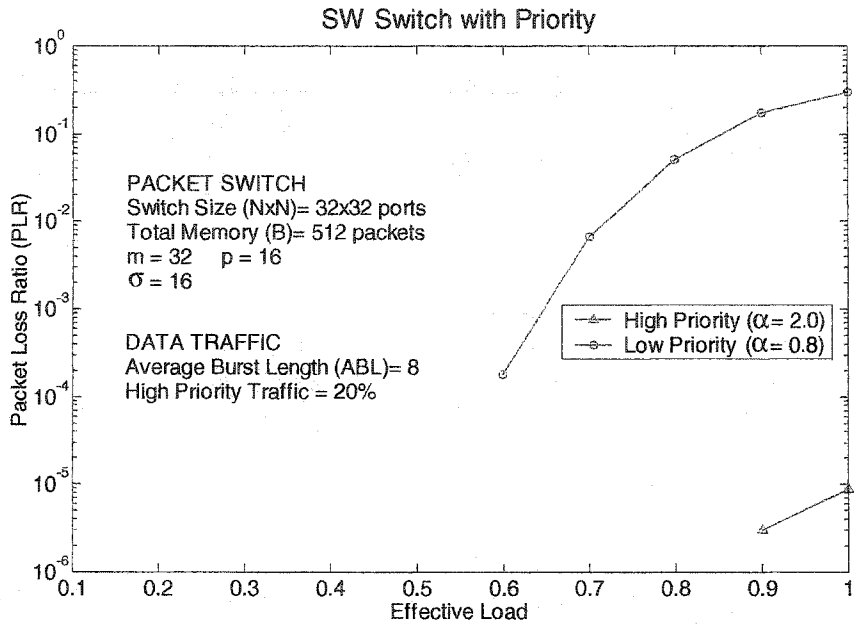


Fig. 4.15 Packet-loss ratio (PLR) in SW-switch with priority ($\alpha_H = 2.0$ and $\alpha_L = 0.8$)

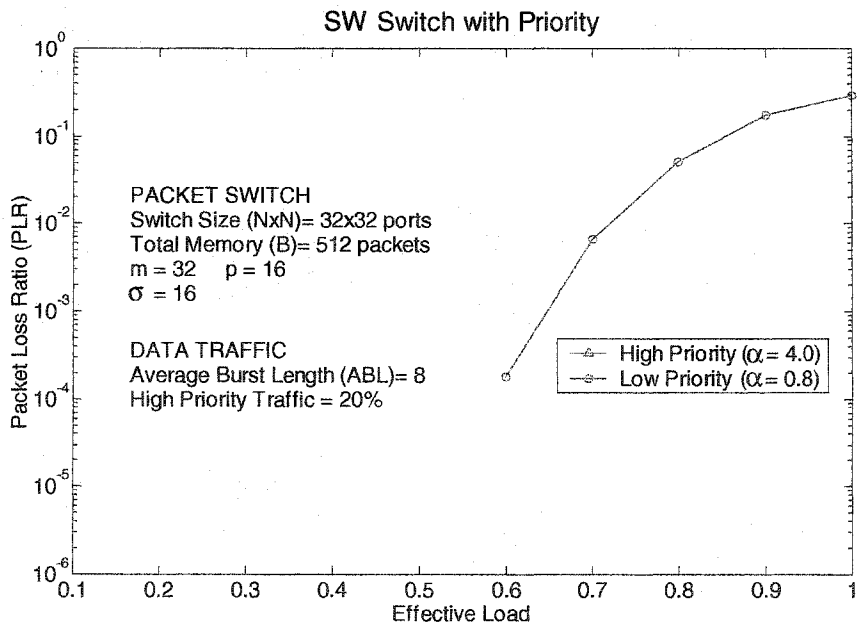


Fig. 4.16 Packet-loss ratio (PLR) in SW-switch with priority ($\alpha_H = 4.0$ and $\alpha_L = 0.8$)

Increasing value of α_H = 4.0 in Fig. 4.16 makes the PLR disappear for the high priority class. Therefore, we can set value of α constant in each priority class and increase the corresponding dynamic threshold (DT) value that regulates the sharing of the buffer space in order to provide different QoS guarantees to the various priority classes.

We continue to use in the following figures and tables values of α constant that produce no packet-loss in the high priority class (i.e. α_H = 4.0 and α_L = 0.8 for high priority and low priority respectively). We try to distinguish the performance of the two priority classes with the privileged allocations of memory resources to the high priority packets. As expected in Fig. 4.17, average throughput for high priority class is linear; as load increases the switch throughput also increases because no packet is dropped in the high priority class due to the setting of α constant. We should notice that average throughput in Fig. 4.17 for each priority class is scaled with respect to the percentage of packets belonging to each priority class at the input traffic, i.e., for example a 100% throughput in the high priority class means that all 20% of high priority packets present at the input traffic were successfully processed and forwarded by the packet switch. A sharp decrease in throughput performance is suffered for the low priority class at high loads (> 80%) due to insufficient buffer space; most of the memory resources are consumed by the high priority class. Average packet-delay (packet latency) is presented in Fig. 4.18 for the two priority classes. That is the average number of switch cycles required by packets to move from input-ports to output-ports of the packet switch. It is observed that packet latency is very low for the high priority class because at each output queue, high priority packets are processed and serviced ahead of low priority packets on the queue line to be read out from memory-modules.

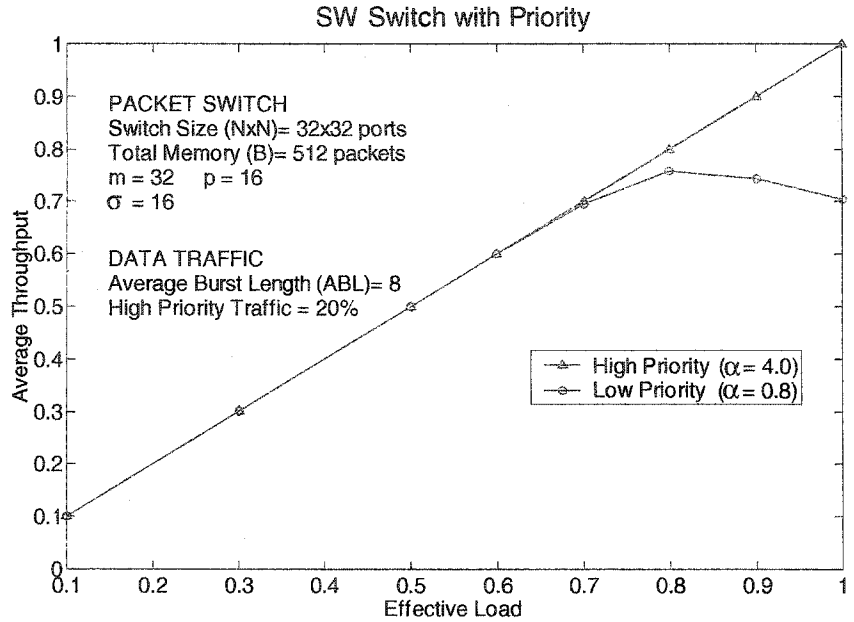


Fig. 4.17 Average throughput in SW-switch with priority for the two priority classes

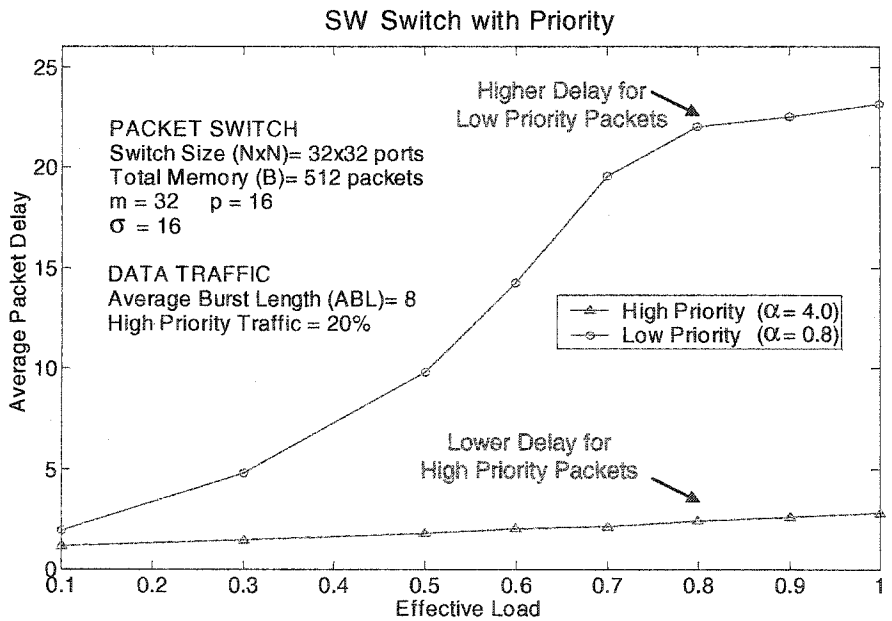


Fig. 4.18 Packet latency in SW-switch with priority for the two priority classes

Multiple packets could contend to be read out from a memory-module within a switch cycle. This memory contention at the READ stage is solved by speedup of memory-modules in order that two or more packets depart from any memory-module within a given switch-cycle. Fig. 4.19 shows the average memory-bandwidth requirement needed by memory-modules to solve contention. It is measured in Fig. 4.19 the number of memory-cycles required on average to read out outgoing packets. High priority class requires less average memory-bandwidth than low priority class in part because high priority packets represent only 20% of the total traffic. However, in overloaded conditions (> 80%) high priority packets consume a great amount of buffer space having a noticeable increase in the memory-bandwidth requirement. The amount of memory resources employed by the high priority class causes the memory-bandwidth requirement for the low priority class stabilizes and even decreases (Fig. 4.19) at overloaded conditions (> 80%) due to a decline on its throughput performance (Fig. 4.17). The worst-case memory-bandwidth requirement is illustrated in Table 4.1 and Table 4.2 for high priority class and low priority class traffic respectively. That is the number of memory-cycles required in a given switch cycle by packets to be read out from the memory-modules. For the simulation performed the high priority class requires up to five memory-cycles to avoid contention at READ stage (Table 4.1). It is observed that more than 98% of high priority packets can be read out from memory-modules in two memory-cycles. For the low priority class the maximum number of memory-cycles required to solve contention is eight (Table 4.2). However, more than 98% packets can be read out in three memory-cycles.

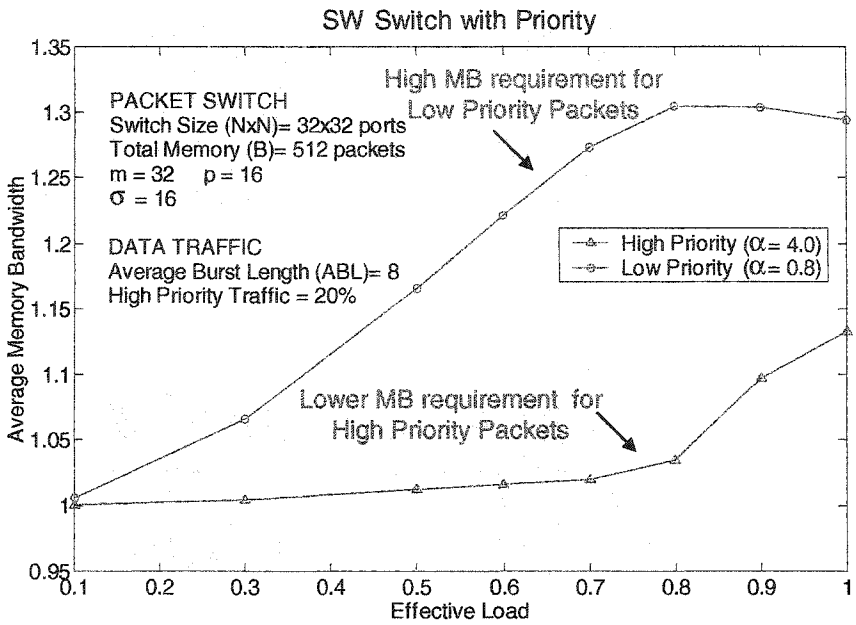


Fig. 4.19 Average memory-bandwidth requirement for SW-switch with priority for the two priority classes

	1 cycle	2 cycles	3 cycles	4 cycles	5 cycles
10% load	99.9464%	0.0536%	0%	0%	0%
30% load	99.2994%	0.6993%	0.0012%	0%	0%
50% load	97.7424%	2.2436%	0.0140%	0%	0%
60% load	97.1348%	2.8449%	0.0200%	0.0003%	0%
70% load	96.5383%	3.4282%	0.0333%	0.0002%	0%
80% load	93.9894%	5.8635%	0.1450%	0.0019%	0.0003%
90% load	83.5041%	15.5415%	0.9297%	0.0242%	0.0005%
100% load	78.1758%	20.2410%	1.5285%	0.0538%	0.0010%

Table 4.1 Worst-case memory-bandwidth requirement for SW-switch with priority for high priority class

	1 cycle	2 cycles	3 cycles	4 cycles	5 cycles	6 cycles	7 cycles	8 cycles
10% load	98.9063%	1.0911%	0.0026%	0%	0%	0%	0%	0%
30% load	87.6184%	11.8506%	0.5212%	0.0095%	0.0002%	0%	0%	0%
50% load	72.4665%	24.0579%	3.2221%	0.2418%	0.0113%	0.0005%	0%	0%
60% load	65.4814%	28.5388%	5.3597%	0.5764%	0.0416%	0.0020%	0%	0%
70% load	59.6292%	31.7444%	7.4864%	1.0387%	0.0951%	0.0059%	0.0002%	0%
80% load	56.4858%	33.2270%	8.7553%	1.3737%	0.1455%	0.0119%	0.0007%	0%
90% load	56.5506%	33.2027%	8.7272%	1.3711%	0.1368%	0.0110%	0.0006%	0%
100% load	57.4867%	32.6789%	8.4163%	1.2775%	0.1312%	0.0090%	0.0004%	0.0001%

Table 4.2 Worst-case memory-bandwidth requirement for SW-switch with priority for low priority class

CHAPTER 5

CONCLUSIONS

Internet routers and switches employ memory-modules to buffers packets during times of congestion. Particularly, shared-memory based switches use a central memory-module which is shared among ports. A background discussion on switching architecture and schemes has been presented in chapter-1 of this thesis. The memory-sharing scheme used in a shared-memory based switching system has a direct impact on the throughput performance and utilization of the switching system.

In chapter-2 of this thesis the throughput and packet-loss performance were evaluated and compared for various memory-sharing schemes, i.e., individual-static threshold, global-static threshold, shared memory with dedicated access (SMDA) and dynamic queue length threshold. The performance evaluation was done under balanced and unbalanced bursty traffic. It was observed that the throughput performance of the switch declined under unbalanced bursty traffic. The uneven distribution of bursts of incoming packets caused active output-ports become very congested leading to packet-loss. Every memory-sharing scheme evaluated in this thesis has a tunable parameter α that controls the degree of sharing of the common memory space. Value of α constant can

be optimized in each memory-sharing scheme to work satisfactorily under a particular traffic condition. The best performance in a shared-memory based switch was obtained using the dynamic queue length threshold scheme. This scheme was very robust against variations in α values; which is a desirable feature in order for the switch performance to be sustained under diverse traffic conditions. Furthermore, dynamic queue length threshold scheme provided the most efficient sharing of the common memory space leading to the higher throughput performance especially in overloaded traffic conditions.

The memory access speed restricts the ability for shared-memory based switches to be scaled to larger capacity. Switching architectures deploying parallel memory-modules that function together as a common memory space shared by all input and output ports provide one way to overcome memory-bandwidth constraint. There are two well known architectures that fall in this category, namely the sliding-window (SW) switch architecture invented by Dr. Kumar of UTPA and the shared-multibuffer (SMB) switch architecture invented by Yamanaka et al of Mitsubishi Electric Corp., Japan. In chapter-3 of this thesis the switching behavior of these two architectures were simulated. Performance evaluation was conducted to evaluate and compare the throughput, packet-loss, and memory-bandwidth of these two switches under bursty-traffic conditions. The SW-switch architecture incurred packet contention for memory resources only at the WRITE stage requiring speedup at the WRITE stage. On the other hand, SMB-switch architecture required speedup of memory-modules at the READ stage. For SW-switch, two different memory assignment schemes were evaluated. It was observed that the SW-switch architecture with *assignment scheme-2* was able to achieve 100% parallel-write and parallel-read operations at the cost of dropping packets that couldn't be stored in one

memory cycle. Based on the comparative evaluation, the memory-bandwidth requirement for SW-switch architecture was observed to be far less than that of the SMB-switch architecture.

In the chapter-4 of this thesis, the SW-switch architecture was modified and extended to handle differentiated quality-of-service (QoS). A version of dynamic queue length threshold called AWA was implemented in order to control the sharing of the common buffer space. The self-routing parameter assignment circuit (PAC) was modified and some communication was allowed between memory controllers for exchanging updates in parameter information. An increase in memory-bandwidth at the WRITE stage was necessary in order to solve packet contention that originated when parameter information was modified for low priority inside logical queues due to insertion of the high priority information. The priority-switching scheme was simulated and it was observed that the SW-switch architecture was able to provide space priority and time priority for traffic with two priority classes. Performance evaluation of each priority class showed that the packet loss for high priority and low priority packets could be controlled by adjusting the individual threshold on sharing of common memory space for the high priority packets and low priority packets.

REFERENCES

[1] D. Comer, Computer Networks and Internets, Prentice Hall, Forth Edition, 2003

[2] D. Emberley, S. Perrin, and T.S. Valovic. More is not enough: Bandwidth end use forecast and analysis, 2000-2005. Analyst brief, IDC, February 2002

[3] RHK. United States Internet traffic experiences annual growth of 100%, but just 17% revenue growth. Press release #157, RHK, Telecommunication Industry Analysis, May 2002

[4] R. Ramaswami, and K.N. Sivarajan, Optical Networks, A Practical Perspective, Second Edition, M.K., 2002

[5] P. Molinero-Fernandez, Circuit Switching in the Internet, Ph.D. Dissertation, Department of Electrical Engineering, Stanford University, June 2003

[6] Partha P. Mitra, and Jason B. Stark, Nonlinear limits to the information capacity of optical fiber communications, Nature, 411:1027–1030, June 2001

[7] H.J. Chao, C.H. Lam, E.Oki, Broadband Packet Switching Technologies: A Practical Guide to ATM Switches and I Routers, John Wiley & Sons, New York, 2001

[8] Y.-M Joo, and N. McKeown, "Doubling memory bandwidth for network buffers", INFOCOM '98, IEEE Proceedings, April 1998, vol.2, pp:808-815

[9] H.J. Chao, "Next generation routers", IEEE Proceedings, Sept. 2002, vol 90, issue 9, pp 1518-1558

[10] M. Karol, M. Hluchyj, and S. Morgan; "Input versus output queueing on a space division switch," IEEE Trans. Communications, 1987, pp.1347-1356.

[11] M. Chen, and N.D. Georganas, "A fast algorithm for multi-channel/port traffic scheduling," Proceedings of IEEE Supercom/ICC '94, pp.96-100

[12] M. Karol, and M. Hluchyj, "Queueing in high-performance packet-switching," IEEE J. Selected Area Communications, Dec. 1988, vol.6, pp.1587-1597

[13] A. Mekkittikul, Scheduling Non-Uniform Traffic in High Speed Packet Switches and Routers, Ph.D. Dissertation, Department of Electrical Engineering, Stanford University, November 1998

[14] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High speed switch scheduling for local area networks," ACM Transaction. on Computer Systems, Nov. 1993, pp. 319-352

[15] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," Proceedings of INFOCOM, 1996, pp. 296-302

[16] R.Y. Awdeh and H.T. Mouftah, "Survey of ATM switch architectures," Computer Networks & ISDN Systems, 1995, vol. 27, pp. 1567-1613

[17] H. Suzuki, H. Nagano, T. Suzuki, T. Takeuchi, and S. Iwasaki, "Output buffer switch architecture for asynchronous transfer mode," IEEE International Conference on Communications, June 1989, vol 1, pp. 99-103

[18] F.A. Tobagi, "Fast packet switch architectures for broadband integrated services digital networks," IEEE Proceedings, Jan. 1990, vol.78, no.1, pp. 133-167

[19] G. Kesidis, and N. McKeown, "Output-buffer ATM Packet Switching for Integrated-Services Communication Networks," ICC '97, Aug. 1997

[20] G. Kesidis, ATM Network Performance, Kluwer Academic Publishers, Boston, MA, 1996

[21] L. Yashe, and L. Zengji , "Performance analysis of ATM switch fabric with combined-input/output buffering," Communication Technology Proceedings, ICCT'96., 1996, vol.1, pp. 508-512

[22] C. Minkenberg, and T. Engbersen, "A combined input and output queued packet switched system based on PRIZMA switch on a chip technology," Communications Magazine, IEEE, Dec. 2000, vol. 38, issue 12, pp. 70-77

[23] H.J. Chao, L.S. Chen; "Delay-bound guarantee in combined input-output buffered switches," Global Telecommunications Conference, GLOBECOM '00, IEEE, 2000, vol. 1, pp. 515-524

[24] J. Garcia-Haro, and A. Jajszczyk, "ATM shared-memory switching architectures," IEEE Network, Jul/Aug 1994, vol.8, issue 4, pp.18-26

[25] N. Endo, T. Kozaki, T. Ohuchi, H. Kuwahara, S. Gohara, "Shared buffer memory switch for an ATM exchange," IEEE Transactions on Communications, Jan. 1993, vol.41, no.1, pp. 237-245

[26] T. Kozaki, N. Endo, Y. Sakurai, O. Matsubara, M. Mizukami, and K. Asano, "32x32 shared buffer type ATM switch VLSI's for B-ISDN's," IEEE J. Select. Areas Commun., Oct. 1991, vol. 9, pp. 1239–1247

[27] M. Arpaci, and J.A. Copeland, "Buffer management for shared-memory ATM switches", IEEE Communications Surveys & Tutorials, vol. 3, no. 1, First Quarter 2000 pp. 2-10

[28] M.I. Irland, "Buffer management in a packet switch," IEEE Transactions on Communications, 1978, vol.26, pp. 328-337

[29] F. Kamoun, and L. Kleinrock, "Analysis of shared finite storage in a computer node environment under general traffic conditions," IEEE Transactions on Communications, 1980, vol.28, pp.992-1003

[30] A.K. Choudhury, and E.L. Hahne, "Dynamic Queue Length Thresholds for Shared-Memory ATM Switch," IEEE/ACM Transactions of Networking, 1998, vol.6, no.2, pp.130-140

[31] M. Pustisek, A. Kos, and J. Bester, "Buffer management in packet switching networks", EUROCON 2003, Sept. 2003, vol.1, pp. 276-280

[32] S. Kumar, A. Munoz, and T. Doganer, "Performance Comparison of Memory-Sharing Schemes for Internet Switching Architecture," The Proceedings of IEEE International Conference on Networking, March 2004

[33] A. Munoz, and S. Kumar, "Effect of Unbalanced Bursty Traffic on Memory-Sharing Schemes for Internet Switching Architecture," The Proceedings of International Conference on Networking, April 2005

[34] S. Iyer, and N. McKeown "Making Parallel Packet Switches Practical," IEEE INFOCOM, March 2001, vol.3, pp.1680-87

[35] S. Iyer, and N. McKeown "Analysis of the Parallel Packet Switch Architecture," IEEE/ACM Transactions on Networking, April 2003, pp.314-324

[36] Y. Sakurai, N. Ido, S. Gohara, and N. Endo, "Large-scale ATM multistage switching network with shared buffer memory switches," Communications Magazine, IEEE, Jan. 1991, pp. 90-96, vol. 29, issue 1

[37] D. Basak, A.K. Choudhury, and E.L. Hahne, "Sharing memory in banyan-based ATM switches," IEEE Journal on Selected Areas in Communications, June 1997, vol.15, issue 5, pp.881-891

[38] H. Yamanaka, H. Saito, H. Yamada, M. Tsuzuki, S. Kohama, H. Ueda, H. Kondoh, Y. Matsuda, and K. Oshima "Scalable Shared buffering ATM Switch with a Versatile Searchable Queue," IEEE Journal on Selected Areas in communications, June 1997, vol.15, no.5

[39] K. Oshima et al., "A new ATM switch architecture based on STS-type shared buffering and its implementation," in Proc. ISS 1992, Yokohama, Japan, 1992, pp. 359–363.

[40] J. Lee, J. Sohn, and M. Lee, "Design of a shared multi-buffer ATM switch with enhanced throughput in multicast environments," ATM Workshop, IEEE Proceedings, May 1999 pp.455-461

[41] S. Kumar and D.P. Agrawal, "The sliding-window approach to high performance ATM switching for broadband networks," in Proc. IEEE GLOBECOM, London, U.K., Dec. 1996, pp. 772–777

[42] S. Kumar, "The Sliding-Window Packet Switch: A new class of packet switch architecture with plural memory modules and decentralized control," IEEE Journal on Selected Areas in Communications, May 2003, vol. 21, No. 4, pp. 656-673

[43] A. Munoz, and S. Kumar, "On Design of Internet Routers/Switches Deploying Parallel Memory Modules," International Conference on Computing, Communications and Control, August 2004

[44] S. Kumar, T. Doganer, and A. Munoz, "Effect of Traffic Burstiness on Memory-Bandwidth of the Sliding-Window Switch Architecture," The Proceedings of International Conference on Networking, March 2004

[45] S. Kumar., and T. Doganer, "Memory-Bandwidth Performance of the Sliding-Window based Routers/Switches" The proceedings of the IEEE Local and Metropolitan Area Networks, April 2004

[46] T. Doganer, Design and Evaluation of High-Performance Packet Switching Schemes, M.S. Thesis, Department of Electrical Engineering, University of Texas-Pan American, April 2004

[47] L. Burchard, "Analysis of Data Structures for Admission Control of Advance Reservation Requests," IEEE Transactions on Knowledge and Data Engineering, March 2005, vol.17, issue 3, pp.413-424

[48] Sheau-Ru Tong; Yuan-Tse Yu; Chung-Ming Huang; Lin, M.-H., "Efficient resource reservation based on communication paradigms for multicast multimedia applications," IEEE Transactions on Broadcasting, Sept. 2004, vol.50, issue 3, pp.260-278

[49] M. Menth, S. Kopf, and J. Milbrandt, "A performance evaluation framework for network admission control methods," IEEE/IFIP Network Operations and Management Symposium, April 2004, vol.1, pp.307-320

[50] K. Mase, "Toward scalable admission control for VoIP networks," IEEE Communications Magazine, July 2004, vol.42, issue 7, pp.42-47

[51] R. Sivakumar, T. Kim, N.Venkitaraman, and V. Bharghavan, "Achieving per-flow weighted rate fairness in a core stateless network," IEEE Proc. Int. Conf. Distributed Computing Systems, April 2000, pp.188-196

[52] R.R.-F Liao, and A.T. Campbell, "Dynamic core provisioning for quantitative differentiated services," IEEE/ACM Transactions on Networking, June 2004, vol.12, issue 3, pp.429-442

[53] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," IEEE/ACM Transactions on Networking, April 1999, vol.7, issue 2, pp.188-201

[54] Yiping Gong; and Bin Liu, "Performance evaluation of a parallel-poll virtual output queued switch with two priority levels," IEEE International Conference on

Communications, Circuits and Systems and West Sino Expositions, June 2002, vol.1, pp.669-674

[55] A.K. Choudhury, and E.L Hahne, "Space priority management in a shared memory ATM switch," IEEE Global Telecommunications Conference, GLOBECOM '93, Nov. 1993, vol.3, pp.1375-1383

[56] R. Roy, and S.S. Panwar, "Efficient buffer sharing in shared memory ATM systems with space priority traffic," IEEE Communications Letters, April 2002, vol.6, issue: 4, pp.162-164

[57] E.L. Hahne, and A.K. Choudhury, "Dynamic queue length thresholds for multiple loss priorities," IEEE/ACM Transactions on Networking, June 2002, vol.10, issue 3, pp.368-380

VITA

Alvaro Munoz was born in Distrito Federal, Mexico on February 5, 1974 as the son of Brigido Munoz and Beatriz Vargas. He completed his undergraduate B.S. degree in Electrical Engineering at the Instituto Tecnologico de Matamoros, Mexico, in 1998. From 1999 to 2002, he worked with Delphi Deltronicos and Teccor Electronics, Mexico as a Process Engineer. In 2002 he entered the Graduate School of the University of Texas-Pan American. Subsequently he received the M.S. degree in Electrical Engineering on May 2005.