Switch Architecture for Efficient Transfer of High-Volume Data in Distributed Computing Environment

SANJEEV KUMAR, SENIOR MEMBER, IEEE AND ALVARO MUNOZ, STUDENT MEMBER, IEEE [%]
Networking Research Lab, Department of Electrical Engineering
The University of Texas-Pan American, Edinburg, Texas-78541
Ph: 956-381-2401; Email: sanjeevk@utpa.edu

Abstract— Switches are often known to be a perfomance bottleneck in a distributed computing environment that require transfer of high-speed, high-volume data. This is primarily due to buffer overflow in the switches that result in loss of packets. Loss of packets require retransmission of higher layer protocol data segments which floods the network with redundant packets only to be dropped by the TCP layer of the end systems. The redundant packets in the retransmitted segments clog the network switches and thus exagerating the congestion and further packet loss. In distributed computing environment that involves high-volume data transfers, it is very important to minimize packet loss at intermediate switches/routers by maximizing the utilization of memory resources deployed at the switch. Switches deploying shared memory resources are known to incur lowest packet loss compared to other memory allocation schemes. However, the memory being inherently slow, it becomes difficult for a large number of high-speed links to share the common memory space deployed in a shared-memory switch. There have been several attempts made to overcome the memory-speed limit that prevent a shared memory switch from scaling to a larger size. In this paper, we discuss architectures that deploy parallel memory modules to enhance the overall capacity of the shared-memory based switches. We also present performance comparison in terms of memory-speed requirement & packet loss ratio.

Keywords: Distributed Computing, High-Volume Data transfer, Performance evaluation, Shared multibuffer switch, Sliding-window packet switch.

I. INTRODUCTION

High-performance, distributed computing applications in high-energy physics, climate modeling, medical imaging, and earthquake engineering require *efficient transfer* of data in wide area network environment [1]. Switches deployed in data networks are often known to be a perfomance bottleneck in a distributed computing environment that requires transfer of high-speed, high-volume data. This is primarily due to buffer overflow in the switches that result in loss of packets [2]. There are two obvious options to solve this problem. First is to deploy extremly large amount of dedicated buffers at each ports of the switch to minimize the loss of data packets. Second is to use sharing of memory resources in the switch to minimize the loss of data packets at the switches. There are three different strategies commonly used for buffer allocation for switches, namely the dedicated buffers at the input ports, dedicated buffers at the output ports, and shared buffers shared by all the input and output ports. Dedicated buffers are usually very large in size in high-speed switches to keep the packet loss requirements low [2], and they have difficulty in being properly utilized, as at times some ports may be active causing buffer overflow, while others ports may remain idle and underutilized. Furthermore, the dedicated large amount of memory at each port of a high-capacity switch consumes significant amount of circuit space and power. Due to low utilization of dedicated buffers, the concept of sharing the memory resource among all ports of a switch becomes quite attractive. Sharing of memory resources among the ports of a switch architecture is known to provide best utilization of a given memory resource deployed in a switch [3-4]. The measurement results in [3] show the comparative performance of three different types of memory allocations in switches. It is shown in Table-1 that for a given load of 85%, the class of shared memory based switches require the least amount of buffer compared to other dedicated buffer allocation strategies such as input buffer and output buffer to provide a loss of 10^{-9}. Table-1 presents the buffer size required (in ATM cells) to achieve a cell-loss rate of 10^{-9} under 85% traffic load for different size switches with different buffer allocation schemes.

Table-1: Buffer size required for a traffic load of 85% and cell-loss-rate of 10^{-9} [3]

Type	Switch Size	
	16x16	32x32
Shared Buffer	113	199
Input Buffer	320	640
Output Buffer	896	1824

Despite the best memory utilization, the shared memory switches are limited in the number of high-speed links it can support. This is due to the physical limitations imposed on the

[%] Authors are with Networking Research Lab (NRL) in the Electrical Engineering department of UTPA. Work of Dr. Kumar is supported in part by funding from FRC, FDC, CITeC, OBRR/NIH, and digital-X.

memory speed, which in turn limits the scalability of the shared-memory based switches. In order to overcome the limitations imposed by the memory-speed, some architectures deploy parallel memory modules that can be physically separate but logically connected [5-7]. In this paper, we call such architecture to deploy shareable parallel memory modules.

Switching Architectures deploying shareable parallel memory modules are quite versatile in their ability to scale to higher capacity while retaining the advantage of sharing its entire memory resource among all input and output ports. There are only two main classes of such architectures, namely the Shared Multibuffer (SMB) based switch [5-6] and the Sliding-Window (SW) based packet switch [7], that deploy parallel shareable memory modules. Despite their similarity in regards to using shareable parallel memory modules, they differ in switching control & scheduling of packets to parallel memory modules. SMB switch uses centralized control where as the SW switch uses a decentralized control for switching operations. However, in this paper, we discuss operation of packet-scheduling schemes used in these switches. In this paper, we propose a new scheme for assignment of parallel memory modules for the SW-switch, and compare its performance with the scheme used for packet scheduling in the SMB switch (Fig.1) under conditions of identical resource and traffic type.

The performance of these two classes of switching systems is compared in terms of memory-speed required by these systems and the packet loss incurred for a given memory speed. In order to assign incoming packets to parallel memory modules, there are more than one scheme that can be used for Sliding-Window switch [7-8]. In this paper, we use a new memory assignment scheme for the Sliding-Window (SW) switch for assigning packets to parallel memory modules that maximizes the parallel storage of packets to multiple memory modules. The flow-chart for this assignment scheme is shown in Fig.3. We compare the performance of a sliding-window switch deploying this memory assignment scheme (Fig.3) with that of a SMB switch architecture under conditions of identical traffic type and memory resources deployed. Section II of this paper presents the switching scheme for the class of shared multi-buffer (SMB) switch architecture [5-6], which is used primarily for comparison purposes. The sliding-window switch architecture [7] with a new memory assignment scheme is presented in section III. Section IV presents the bursty traffic model that can be used to depict high-volume, data-intensive traffic. A bursty traffic of average burst length=8 is used to evaluate the performance of the sliding-window switch and that of SMB switch. Performance results are discussed in Section V, and Section VI concludes the paper.

II. THE SHARED MULTI-BUFFER (SMB) SWITCH

The Shared-Multibuffer (SMB) switch architecture is discussed in detail in [5-6]. Multiple memory-modules are shared among all the input/output ports through cross-point switches. The control for the SMB-based switching system is centralized and maintains a buffer address queue for each output port and the idle-addresses pool to store the vacant addresses. For complete sharing of memory-modules, the idle buffer and the address-queue for each output port in SMB switch need 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.

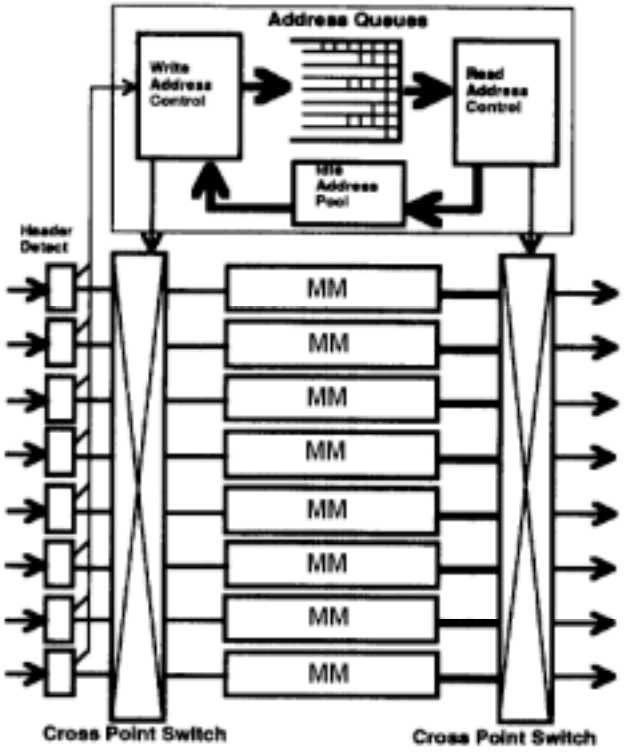

Fig.1: Shared-Multi Buffer (SMB) switch architecture [5-6].

In SMB architecture [3-4], an incoming packet is assigned to the least occupied memory-module first. The least occupied memory-module is given highest priority for an incoming packet to be written to. Based on the occupancy of the memory modules, it is possible to assign different memory modules to different packets input in a given switch cycle. This results in multiple packets stored in different memory modules in parallel, and hence requiring no increase in the memory bandwidth during the WRITE cycle. However, during the READ cycle, it is possible for multiple packets of the same memory module to be scheduled to go to different output ports during the same switch cycle. This requires multiple packets to be read out of the same memory modules in the same switch cycle, and hence this increases the memory-bandwidth requirement of the SMB switch during the READ cycle. Since memory speed-up is required only during the READ cycle, the memory-bandwidth requirement for a SMB switch can be defined as the number of memory READ cycles needed per switch cycle to output the scheduled packets from the memory modules.

III. THE SLIDING-WINDOW (SW) SWITCH

The class of sliding-window (SW) switch is discussed in detail in [7]. The class of the sliding-window switch is characterized by parallel memory modules and decentralized control. The self-routing parameter assignment circuit computes the self-routing parameters (*i*,*j*,*k*) to be attached to the incoming packets. The parameter *j* in the self-routing tag designates the j^{th} output slot vector (OSV), which represents a packet's location in i^{th} memory module, and parameter *k* value determines a packets turn to go out. The parameter assignment circuit first determines the *j* and *k* parameters, and then uses the *j* and *k* values to determine the value of the i^{th} parameter i.e. the memory module (*i*) where an incoming packet is stored [7].

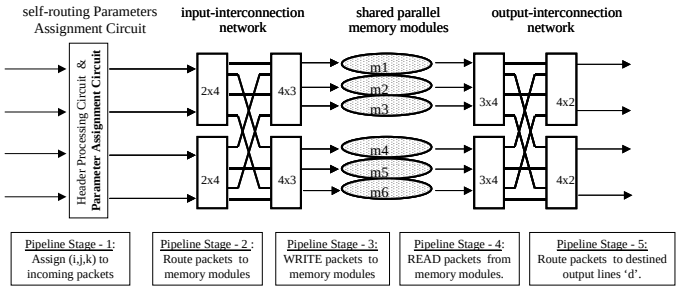

Fig.2: A 4x4 Sliding-Window switch architecture [7].

The packets input in the same switch cycle are assigned different values of *i* (i.e. the i^{th} memory module) in an increasing order. However, in the switching scheme given in [7], it is possible for two or more incoming packets of a switch cycle belonging to different OSVs(j) to be scheduled to be written to same memory module. This requires the switching scheme in [7] to speed-up the memory-modules to enable multiple-write operations in a given switch cycle. This in turn requires an increase in the memory-bandwidth requirement of a SW-switch during the WRITE cycle. The read operation of the SW-switch is such [7] that no more than one packet is output from one memory module, each output ports gets a packet each switch cycle (if available in the memory) and re-ordering of packets is not required. This results in only one read operation per memory module per switch cycle. Hence in the sliding-window switch architecture, the memory speed-up is needed only during the WRITE cycle but not during the READ cycle. The memory-bandwidth requirement for a sliding-window switch is simply the number of memory WRITE cycles needed per switch cycle to store incoming packets to memory modules.

Due to physical limitations, the speed of memory modules can be increased only so far. It is important to design a memory assignment scheme that maximizes the parallel write of packets to different memory modules in a given switch cycle without requiring an increase in memory speed-up.

A. A New Memory Assignment Scheme for SW switch

There are more than one ways for the packets of a switch cycle to be assigned parallel memory modules in the sliding-window switch [7-8]. A new scheme for memory-assignment in SW-switch is presented in this section. The main goal of a memory assignment scheme is to maximize the parallel write of packets to multiple memory modules so that the speed-up requirement of memory modules can be reduced. According to this scheme (Fig.3), an additional array is used called Temp[*i*] for *i*=1 to *m*, where *m* is the number of memory modules deployed in the switch. The Temp[*i*] is used to keep track of the assignment of memory-modules in a given switch cycle (Fig.4). Each switch cycle, the Temp array is initialized to 0 to indicate availability of all memory modules for packet assignment. According to this scheme, first the best option is considered i.e. a packet is assigned to a memory module *i* if the i^{th} slot is available in both the j^{th} OSV as well as the Temp[*i*]. This condition is depicted by steps 306 and 308 of the flow-chart in Fig.3. Availability of i^{th} slot in both arrays, OSV-*j* (i.e. j^{th} OSV column in the scan-table, ST(i,j) in [7]) and the Temp [*i*] is to allow the packets of a switch cycle to be assigned to different memory modules and hence maximize the parallel write of packets to different memory modules.

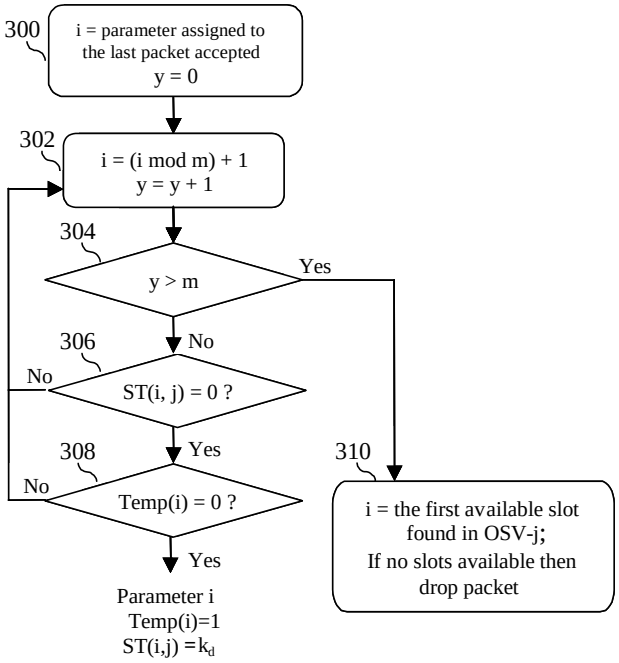

Fig.3: An assignment scheme for parallel memory modules in sliding-window switch architecture

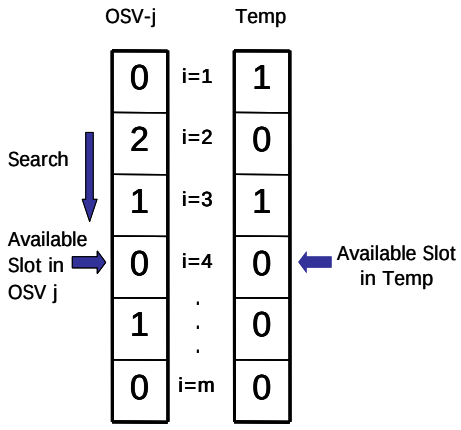

Fig.4: Use of Temp-Array in assignment of memory modules.

Secondly, if an i^{th} slot is available only in OSV-j but not available in the Temp Array then the search defaults to step 310 (Fig.3) that assigns an available space in OSV-j, and ignores the temp array. It can be noticed that first the attempt is made to find the best slot that is available in both the OSV-j and the Temp Array. When that is not possible, then the scheme tries to find the slot only in OSV-j, which is same as the scheme used in [7]. The packet is dropped only when there are no slots available in OSV-j. This means that in a given switch cycle there may be more than one incoming packets that might need to be written to the same memory module. Under this situation, the memory speed will need to be increased compared to the line-speed, for the memory modules to accommodate multiple memory access for writing multiple packets of a switch cycle. According to this scheme, the memory speed-up is needed only at the memory-write but not during the memory-read phase of the switching. In the section below, we perform simulations to compare the memory-speed requirement and Packet-loss performance of the SMB switch and the Sliding-Window switch.

IV. PERFORMANCE EVALUATION

A. Bursty Traffic model

Bulk-data transfer applications transmit data in bursts. To study comparative performance of these switching systems, a bursty traffic is generated using a two state ON-OFF model 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 (Fig. 2) [7].

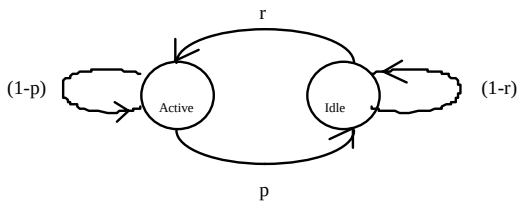

Fig. 5. A two-state ON-OFF model.

According to traffic model used in [7], if p and r characterize the duration of the active and idle period respectively, then the offered load L is given by

$$L = \frac{E_B[i]}{\left(E_I[i] + E_B[j]\right)} = \frac{r}{r + p - rp}.$$

B. Simulation Setup

The *measures of interest* considered in the simulation studies are the *offered load for a bursty traffic* of a given *average burst length (ABL)*, number of memory-cycles needed per switch cycle in the worst-case traffic load of 100%, and *packet-loss ratio (PLR)* of the switch. The simulation experiment started out with empty memory modules and the incoming bursts of packets were uniformly distributed to the output ports. Depending on the offered load, a maximum of 32 million packets were generated for evaluation of performance parameters of 32x32 SMB and Sliding-Window switches. Bulk data transfer applications usually have higher burst length at the source, however, by the time multiple source traffic are multiplexed and rate controlled on a given high speed line, its burst-length might become smaller when it reaches the switch ports. In this paper, we consider a bursty traffic with an average burst length (ABL) of 8 packets just for comparative evaluation of two architectures, even though bulk data transfer in cluster interconnects may have an ABL greater than 8 packets.

C. Switch configuration for SMB and SW Switch

Both switches were configured for comparison study under conditions of identical resources and traffic type. The switch size of 32x32 was considered for both the SMB switch and the SW switch for comparative evaluation. The total shared-memory deployed in both the switches = 2048 packets, the minimum number of memory modules required according to requirements given in [7] = m = 2N = 64, and the packet-size of a memory module = σ = 32 packets. For efficient sharing of common memory space among the output ports of these switches, the dynamic threshold scheme is used as given in [9] with $\alpha = 1$.

V. PERFORMANCE RESULTS & DISCUSSION

In this paper, we measure the worst-case memory-bandwidth requirement, and packet loss ratio (PLR) for the same-size SMB based switch and sliding-window based switch. It is observed in Table-2, the worst-case memory bandwidth requirement in terms of the memory-cycles required to write/read packets to/from the memory in a given switch cycle. The first column of the Table-2 shows the number of memory-cycles used for READ/WRITE operations for packets belonging to a given switch cycle. The second column of the Table-2 shows the percentage of the switch cycles that need a given number of memory-cycles (indicated in the first column of the table) for READ/WRITE operations. It is observed that for the given switch-size of 32x32 used in this simulation, the

SMB switch required a maximum (worst case) of 7 memory-cycles per switch cycle. On the other hand, the sliding-window switch with identical resource and the switching scheme of Fig.3 required a maximum (worst case) of 2 memory-cycles per switch cycle for READ/WRITE operations for packets input in a given switch cycle.

Table-2: Worst-case scenario for # of memory-cycles required per switch cycle for packets' READ/WRITE memory operations in SMB, and SW switch with the memory assignment scheme of Fig.3.

Number of Memory-Cycles used for READ/WRITE operations.	Number of Switch cycles in Shared Multibuffer (SMB) based Switch [2]	Number of Switch cycles in Sliding-Window (SW) Switch with new scheme
1 Cycle	63.65900%	99.99993%
2 Cycles	29.16323%	0.00007%
3 Cycles	6.25252%	0%
4 Cycles	0.84300%	0%
5 Cycles	0.07670%	0%
6 Cycles	0.00543%	0%
7 Cycles	0.00012%	0%

As the size of the switch increases, the memory-bandwidth requirement will also increase, which in turn imposes a physical limitation on the switch's scalability. Since the memory bandwidth can't be increased beyond a certain point, the switches will have to operate with a fixed memory speed. Since the memory-bandwidth is usually fixed due to physical constraints, the performance of the SW switch with switching scheme of Fig.3 is compared with that of SMB switch under conditions of a fixed memory bandwidth. For comparison, we limit the switches' memory-bandwidth to one (i.e. the memory-speed = line speed). We then compare the packet loss ratio (Fig.6) for the SW switch with an assignment scheme given in Fig.3, and the SMB switch under condition of a fixed memory bandwidth (MB) =1 (i.e. without any memory speed-up)

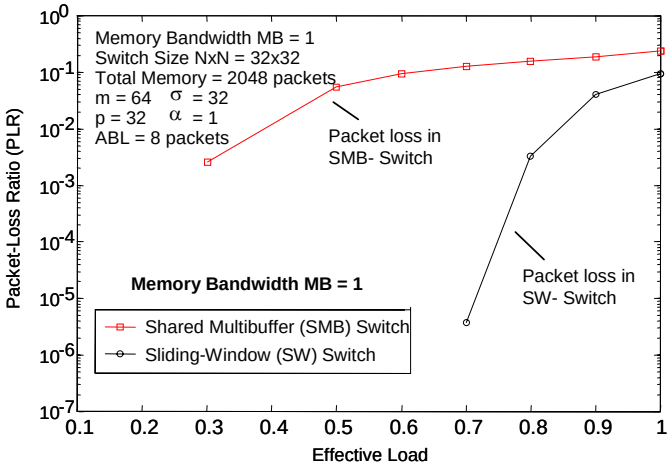

Fig.6. Packet-loss ratio for SMB switch, and SW switch with the assignment scheme of Fig.3, and with memory-bandwidth (MB) = 1

Fig.6 shows the steady-state packet loss ratio (PLR) of the sliding-window switch with switching scheme of Fig.3 to be much reduced compared to that of the same size SMB-switch under conditions of identical memory size, memory bandwidth and the traffic type.

Smaller packet loss for the sliding-window switch means there will be a reduced number of end-to-end retransmission of TCP layer payload data in the network. Use of the sliding window switch can directly lead to a faster end-to-end transfer of high-volume application data in distributed computing and cluster interconnect environment.

VI. CONCLUSION

In this paper, we present a class of switch architecture with parallel and shareable memory modules that can be used for efficient transfer of high-volume data in distributed computing and cluster interconnect environment. There are two known switches that belong to this type of architecture, namely the shared multi-buffer (SMB) switch and the sliding-window (SW) switch. In this paper, we propose a new packet-scheduling scheme for the sliding-window (SW) switch architecture and compare its performance with that of the packet-scheduling scheme of another well-known switch architecture namely the SMB switch. Performance and scalability of these switches are constrained due to the limited memory-speed of the memory modules. Hence, it is very important to have a switch architecture that provides high performance with limited memory-bandwidth. We measure the worst-case memory-bandwidth requirement and packet-loss performance of the sliding-window (SW) switch (using the proposed memory assignment scheme) against the SMB switch under conditions of identical switch size, memory size, memory-speed and traffic type. It is observed that under conditions of identical memory-resource and traffic type, the class of sliding-window (SW) switch with its new packet-scheduling scheme, has much reduced memory-bandwidth requirement compared to that of its SMB-based switch counterpart. Furthermore, for a fixed value of memory-bandwidth and memory-resource deployed in the switch, the class of the sliding-window switch achieves smaller packet-loss when compared with that of a same size SMB-based switch. Smaller packet loss incurred by the sliding-window (SW) switch can directly lead to a faster end-to-end transfer of high-volume application data in a distributed computing and cluster interconnect environment.

ACKNOWLEDGEMENT

Work of Dr. Kumar is supported in part by funding from Faculty Research Council, FDC, CITeC, OBRR/NIH, digital-x Inc.

Authors would like to thank all anonymous reviewers for their useful comments regarding bulk data transfer applications in cluster interconnect environment.

REFERENCES

[1] B. Allcock et al “Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing,” www.isi.edu/~annc/papers/msc01paperann.pdf

[2] B.L. Tierney, “Distributed Storage Caches and Distributed System Performance Analysis Tools,” Supercomputing’98 Distributed Storage Caches and Distributed System Performance Analysis Tools Data Intensive Computing Tutorial

[3] K. A. Lutz, "Considerations on ATM Switching Techniques," *International Journal of Digital and Analog Cabled Systems*, vol. 1, no. 4, October 1988, pp. 237-243.

[4] M. I. Irland, “Buffer management in a packet switch,” *IEEE Transactions of Communications*, vol.26, pp. 328-337, 1978.

[5] K. Oshima, H. Yamanaka, H. Saito, H. Yamada, S. Kohama, H. Kondoh, and Y. Matsuda, “A new ATM switch architecture based on STS-type shared buffering and its implementation,” in *Proc. XIV Int. Switching Symp. (ISS’92)*, vol. 1, Oct. 1992, pp. 359–363.

[6] H. Yamanaka et al, “Scalable Shared-Buffering ATM Switch with a Versatile Searchable Queue,” *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 5, pp. 773-784, June 1997.

[7] 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*, vol. 21, No. 4, pp. 656-673, May 2003.

[8] S. Kumar and T. Doganer, “Memory-Bandwidth Performance of the Sliding-Window based Internet Routers/Switches,” *IEEE Workshop on Local and Metropolitan Area Networks*, San Francisco, CA, April 2004, pp. 205-210.

[9] Choudhury, A.K. and Hahne, E.L., “Dynamic queue length thresholds for shared-memory packet switches,” *IEEE/ACM Transactions of Networking*, vol.6, no.2, pp.130-140, 1998.