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

HARDWARE SUPPORT FOR HIGH PRIORITY TRAFFIC IN VLSI COMMUNICATION SWITCHES

Yuval Tamir
Gregory L. Frazier

December 1990
CSD-900051

Hardware Support for High-Priority Traffic in VLSI Communication Switches †

Yuval Tamir and Gregory L. Frazier

Computer Science Department
4731 Boelter Hall
University of California
Los Angeles, California 90024-1596
U.S.A.
Phone: (213)825-4033 E-mail: tamir@cs.ucla.edu

Abstract

Interconnection networks used in large multiprocessors and multicomputers are composed of small $n \times n$ switches. The architecture of these switches is critical for achieving high-bandwidth low-latency interprocessor communication. Low latency communication is particularly important for interprocessor traffic generated for devices that operate under real-time constraints, as well as certain other system activities, such as exception handling. In order to meet this requirement, it may be necessary to provide special support for such high-priority traffic in many multiprocessor and multicomputer systems. We discuss the design of $n \times n$ switches that can be used to construct communication networks which provide low latency communication for high-priority traffic. We focus on the design of the internal buffers, specifically on buffers where storage is statically or dynamically partitioned between normal traffic and high-priority traffic, and where packets can be handled in non-FIFO order. We evaluate alternative designs and configurations in the context of a multistage interconnection network. Our simulations show that a slightly modified version of the recently introduced *dynamically-allocated multi-queue* buffer can provide superior support for high-priority traffic, at a relatively low cost.

Index Terms: Communication coprocessors, communication switches, interconnection networks, multicomputers, multi-stage networks, non-FIFO buffers, real-time systems, VLSI systems.

† This research has been supported by the SDIO Innovative Science and Technology Office, managed by the Office of Naval Research, contracted to the Jet Propulsion Laboratory under task plan #80-2984; and by the State of California MICRO program.

December 1990

I. Introduction

Efficient interprocessor communication is critical to the ability of multiprocessors and multicomputers to achieve high performance by exploiting parallelism. The interconnection networks used for interprocessor communication are designed to achieve the maximum possible throughput (bandwidth) of data through the network and the minimum latency (delay), using cost-effective implementations. Most traffic patterns through the network result in contention for the use of network links. This contention causes the maximum achievable network throughput to be lower than the theoretical maximum throughput of the network. In addition, unless the network is lightly loaded, the average communication latency is significantly higher than the minimum possible latency.

For a given traffic pattern, there can be significant variance in the latencies of different packets through the network. As the network load increases, the variance of packet latencies increases. For large heavily-loaded networks, some packets may require several times the average latency to reach their destination.

In many cases it is particularly important to reduce the latency of a subset of the packets in the system. For example, an input/output device connected to one of the nodes in a multicomputer system may occasionally need particularly fast communication (high "burst throughputs") with other parts of the system in order to meet real time constraints. Another example is broadcasting of certain types of system-wide events, such as the initiation of global system rollback, which may also require preferential handling by the network. Real-time requirements due to interaction with I/O devices and distributed exception handling thus lead to the need to support special-purpose *high-priority* traffic whose maximum latency is significantly lower than the maximum latency for general traffic.

Small $n \times n$ switches (typically, $2 \leq n \leq 12$) are the key component of multistage interconnection networks used in large multiprocessors [2,7] and of communication coprocessors used in multicomputers [1,14,3,11]. Hence, the performance, reliability, and cost of these small $n \times n$ switches are of critical importance to the success of multiprocessor and multicomputer systems. Since many of these $n \times n$ switches are needed in a large system, there is strong motivation to implement each switch as a single VLSI chip. Furthermore, it is desirable to develop switch architectures that will meet the performance requirements using a low-cost low-power implementation technology such as CMOS VLSI.

This paper describes the design and implementation of small $n \times n$ VLSI communication switches with special hardware support for high-priority traffic. Each switch receives packets at its input ports and routes them to its output ports. If only one packet at a time arrives for a given output port, there are no conflicts, and the packets are routed with the minimum latency. If two packets destined for the same output port arrive at different input ports of a switch at approximately the same time, only one of them can be transmitted immediately while the other one must be buffered in the switch for later transmission. The probability of such conflicts increases with increasing network load (throughput). If some packets are marked as "high-priority," the switch can arbitrate conflicts in favor of these packets. The requirements from the switch are thus to minimize the latency for high-priority packets while maximizing total switch throughput.

The research described in this paper is part of the UCLA ComCoBB (Communication Coprocessor Building-Block) project, whose focus is the design and implementation of a single-chip high-performance communication coprocessor for VLSI multicomputers. The ComCoBB chip is, in part, a small $n \times n$ switch, and must therefore include an efficient packet buffering scheme with the capabilities to meet the requirements of high-priority traffic. We have developed a new type of buffer for small $n \times n$ switches, called a *dynamically-allocated multi-queue* (DAMQ) buffer[13, 6], which provides for significantly higher throughput than alternative buffer designs. In this paper we show that the DAMQ buffer can be easily enhanced to support high-priority traffic and provide low-latency transmission for the high-priority packets despite heavy network traffic. DAMQ buffers are equally useful for use in multicomputer communication coprocessors as well as in the switches of multistage interconnection networks. The evaluation of DAMQ buffers in this paper will be in the context of multistage networks.

In the next section we discuss key issues in the design and implementation of $n \times n$ switches for interconnection networks and introduce four alternative switch organizations. We include a brief description of the DAMQ buffer and its use for normal traffic. In Section III we describe the simulation environment and system model used to evaluate the performance of different types of switches. As a reference point for evaluating practical switch architectures, we introduce a theoretical "ideal" switch which achieves many of the desirable characteristics of an $n \times n$ switch by ignoring the constraints of the implementation technology. The need for special support for high-priority traffic is demonstrated in Section IV. In Section V we present several schemes for providing support for high-priority traffic and focus on the use of a slightly modified DAMQ buffer. The effectiveness of the various schemes for supporting high-priority traffic is evaluated using event-driven simulation in the context of a multistage interconnection network.

II. Switches for Packet-Switching Networks

Buffering of packets in a communication switch is necessary if the succeeding node in the network is not ready to receive a packet or when there are two or more packets in the switch destined to the same output port. For high performance, unnecessary blocking should be minimized — the buffers should be organized in such a way that if there is an available output port and there is a packet destined for that port, that packet will be transmitted without having to wait for packets that need to be sent through other ports. In order to efficiently utilize the available communication bandwidth, variable-length packets must be supported. Otherwise, if the fixed packet size is too large, bandwidth will be wasted by "null bytes" when sending messages smaller than the packet size. If the fixed packet size is too small, bandwidth will be wasted for packet header information when sending large messages, which would consist of multiple packets. Efficient buffer storage utilization requires the ability to allocate storage for these variable-length packets while minimizing internal and external fragmentation [13].

An ideal switch has infinite buffer space and buffers (delays) a packet only as long as the output port to which the packet is destined is busy. In a real implementation, buffers are finite and have a finite bandwidth. This can result in conflicts due to attempted simultaneous accesses to shared buffers and in messages that cannot be received due to lack of buffer space. Packets ready to be transmitted may be blocked behind packets waiting for their output port to free up, and significant delays may be introduced by complex memory allocation schemes required to handle variable length packets.

In a previous paper [13] we discussed alternative organizations for the packet buffers in small $n \times n$ switches. The goals were to maximize network throughput and minimize average latency while providing efficient support for virtual cut-through and minimizing the total size of the packet buffers. Two key design decisions were considered: (1) the location of the buffers — a central buffer pool, buffers at the input ports, or buffers at the output ports, and (2) the organization of the buffers — conventional FIFO buffers versus various types of non-FIFO buffers.

Complete sharing of available storage by all communication ports results in more efficient storage utilization than static partitioning of the buffer storage between ports. Hence, ideally it would be best to use central buffers where all the available storage can be dynamically allocated to arriving packets from any input port. We call a switch that uses such buffers a *centrally-buffered, dynamically-allocated* (CBDA) switch. Unfortunately, there are fundamental difficulties in producing an efficient implementation of a switch with shared central buffers. Specifically, in order to support simultaneous multiple reads and writes from, say, four input ports and four output ports, the buffer storage will be slow as well as expensive in terms of chip area. Furthermore, the control must be capable of rapidly (within one or two clock cycles) allocating memory for multiple variable size packets simultaneously while minimizing internal and external fragmentation [13]. As discussed in [13], output buffering is undesirable for similar reasons — the need to store in the same buffer multiple packets which arrive simultaneously and are destined for the same output port as well as the problem of efficient storage allocation for variable-length packets.

The above considerations lead to the choice of buffering at the input ports in order to achieve efficient high performance switch implementations [13]. In most switch designs, the input port buffers are managed as FIFO (first-in-first-out) queues [5, 15]. The disadvantage of FIFO buffers at the input ports is that packets may be blocked unnecessarily — a single packet at the head of the queue whose destination output port is busy can block all other packets in that queue from being transmitted even if their destination output ports are idle. This unnecessary blocking can severely degrade network performance, reducing the maximum throughput and increasing average latency of packets through a network.

The deficiencies of FIFO buffers motivate the design of buffers which allow handling of packets in non-FIFO order [10, 8, 13, 4]. With appropriate interconnection between input and output ports, packets destined for an idle output port are not blocked by packets that have arrived earlier at the same input port and are waiting for an output port that is temporarily busy. We have considered three possible designs of buffers that support non-FIFO handling of packets: *dynamically allocated multi-queue* (DAMQ) buffers, *statically allocated multi-queue* (SAMQ) buffers, and *statically allocated, fully connected* (SAFC) buffers (Figure 1) [13].

With SAMQ and SAFC buffers the storage of each input port buffer of an $n \times n$ switch is statically partitioned into n FIFO queues, one for each output port [10]. The first packet in any one of the n queues can be accessed regardless of the number of packets in the other queues. Thus, if a packet that is supposed to be sent from a particular input port is blocked, it may be possible to send another packet from the same input port destined to a different output port. While only one packet at a time may be sent from a SAMQ buffer, the SAFC buffer allows multiple packets, destined to different output ports, to be sent

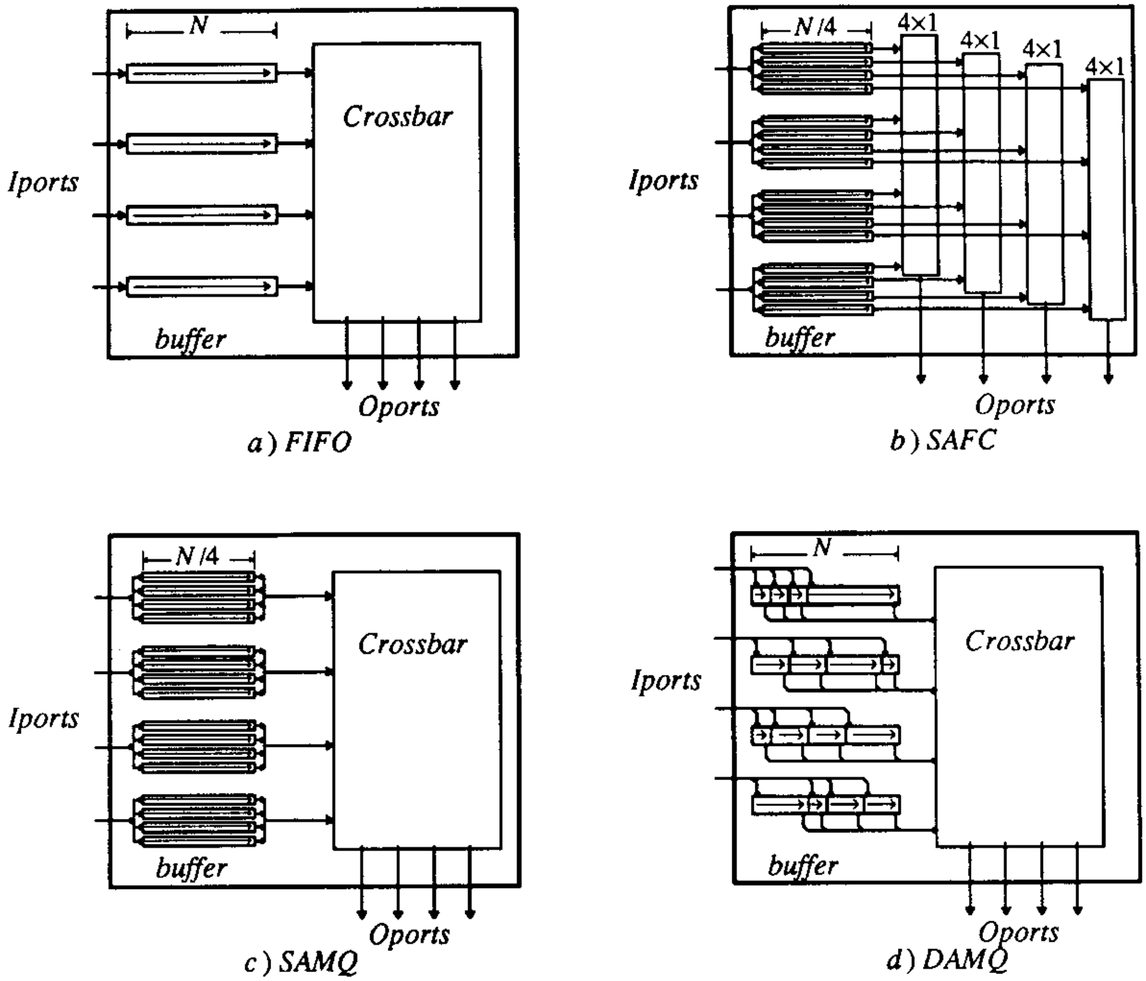


Figure 1: Alternative practical buffer organizations for communication switches.

simultaneously from a single input port. The cost of the increased capability of the SAFC buffer is a more complex switch which includes four 4×1 switches instead of the single 4×4 crossbar switch used with the other buffer types.

The DAMQ buffer is a new type of input buffer that we have developed [13, 6]. In this buffer the available storage is *dynamically* partitioned between *n* FIFO queues, one for each output port. This dynamic partitioning leads to more effective use of the available storage for variable size packets and for handling variations in traffic patterns that may temporarily "skew" the required buffering capacity for different output ports. Multiple queues of packets are maintained within a DAMQ buffer in linked lists. In order to handle variable size packets, the buffer is partitioned into eight-byte blocks. With our design for a maximum packet size of 32 bytes, each packet occupies from one to four blocks (the set of blocks which hold a packet is referred to as that packet's *slot* within the buffer). The linked lists of blocks are managed by dedicated finite state machines (FSMs) [6].

In previous publications [13, 6] we have presented detailed discussions of the advantages of the DAMQ buffer over FIFO, SAMQ, and SAFC buffers, based on implementation considerations as well as performance for normal traffic. It was shown that the control of DAMQ buffers is more complex than the control of simple FIFO buffers [6]. Specifically, there are additional *head*, *tail*, and *link* registers for the linked lists and larger FSMs are needed. Based on our implementation, the additional circuitry requires approximately the same area as 32 bytes of storage in the buffer (one packet slot). If the buffer size is increased, the only changes in the control are additional bits for the *head*, *tail*, and *link* registers, needed to address a larger number of blocks. Hence, the size of the control is nearly independent of the size of the buffers.

III. Performance Evaluation of the Packet Switches

In order to evaluate the performance of the different types of communication switches, we have simulated communication on a 64×64 Omega network [9] constructed from three stages of 4×4 switches. The network connects sixty-four processors (message senders) to sixty-four memory modules (message receivers). The minimum delay per stage is called a *stage cycle* [15]. A stage cycle is the latency of a packet per stage in an empty network, where there is no contention for buffer space or output ports. Thus, the smallest latency for a packet across the three-stage network is three stage cycles. The network operates synchronously, where at the beginning of each stage cycle each sender or switch either sends a packet or waits for the next stage cycle before sending a packet. All our simulations are for *blocking* switches, where a packet is transmitted only if there is a free packet slot in the buffer of the destination switch.

At the senders, the interval between packet creation follows a geometric distribution. If the buffer in the switch connected to the sender is full, the packet is queued at the sender and transmission is re-tried in the next stage cycle. The packet latency we report is measured in stage cycles from the time of packet creation. The interval to the next packet creation is calculated from the time the current packet leaves the sender. The throughput is reported as the fraction of the maximum network throughput that would be achieved if all the network links were transmitting at all times (this would be possible if there were no conflicts in any of the switches). The destination of packets is uniformly distributed so packets have an equal probability of being sent to any one of the possible sixty-four receivers.

Most simulations were run with each sender generating approximately one thousand packets. In order to eliminate start-up effects, statistics were gathered only after the first ten percent of the packets were received. Simulation was terminated once there was one sender that had completed sending its one thousand packets. In order to verify the validity of the results, for each rate of generating messages at the sources, multiple runs were performed with different random number seeds. At least three runs were performed for each data point reported in this paper. The throughput and latency numbers used are the averages of results from these multiple runs. The throughputs for individual simulation runs were all within 3% of the throughput numbers reported. The average latencies reported were within 6% of the average latencies obtained in simulations.

Simulations were run using four practical switch types: FIFO, SAMQ, SAFC, and DAMQ. In addition, we have run simulations of networks composed of CBDA switches — switches with shared central buffers. As discussed in Section II, there are fundamental difficulties in implementing CBDA switches. We include these switches in our evaluation studies not as a practical implementation option but as hypothetical "idealized" switches whose performance is a yardstick against which the performance of practical switches can be measured. The idealized CBDA switch maintains in its central buffer a separate FIFO queue for each output port and can simultaneously receive packets from all input ports while transmitting packets through all the output ports. In this context it should be noted that an additional significant implementation problem with shared central buffers is the need for complex flow control. In our simulations, when there are fewer free buffer slots than inputs ports (so that not all the input ports can receive packets), the input ports which do not have packets pending for them are blocked first, and the rest are prioritized according to the length of time the packets had spent in the preceding switch. In an actual network implementation, the information about pending packets in other switches would not be available to the switch. However, this is not of concern to us since the CBDA switch is not being considered as an implementable switch.

Some of the advantages of the DAMQ buffer are demonstrated by the simulation results shown in Figure 2. This graph shows the average latency versus throughput of networks composed of the five buffer types with four packet slots per input port and uniform traffic. All five networks were simulated under a wide range of loads: from lightly loaded (a throughput of 0.10) to *saturation*, where saturation is the maximum throughput achievable with each network. The network composed of DAMQ switches was able to handle significantly higher throughput at lower average latency compared to networks composed

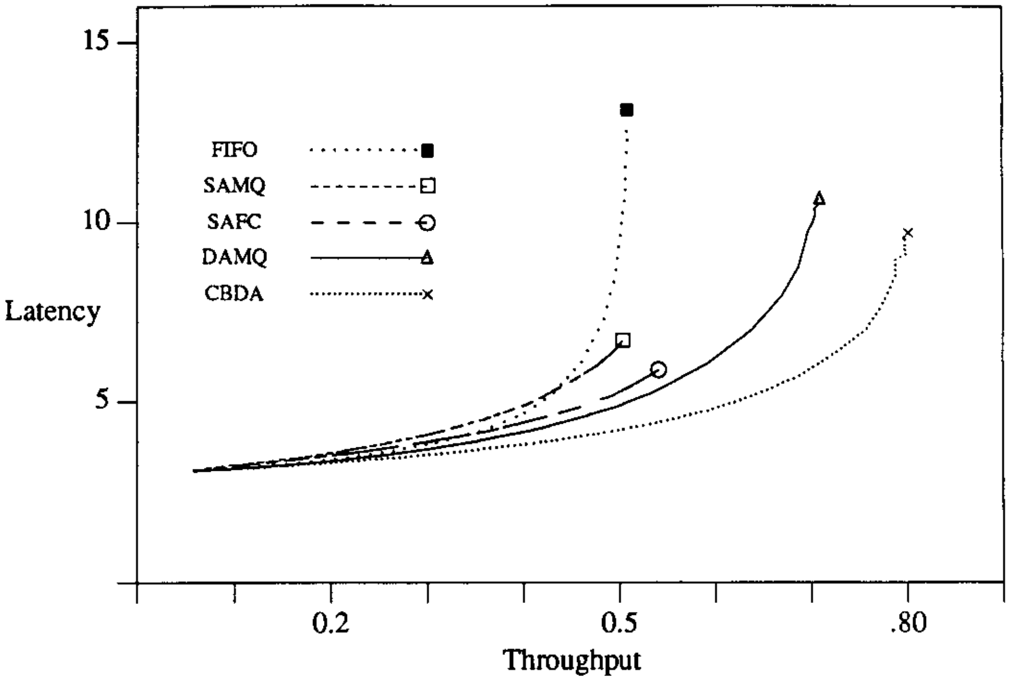


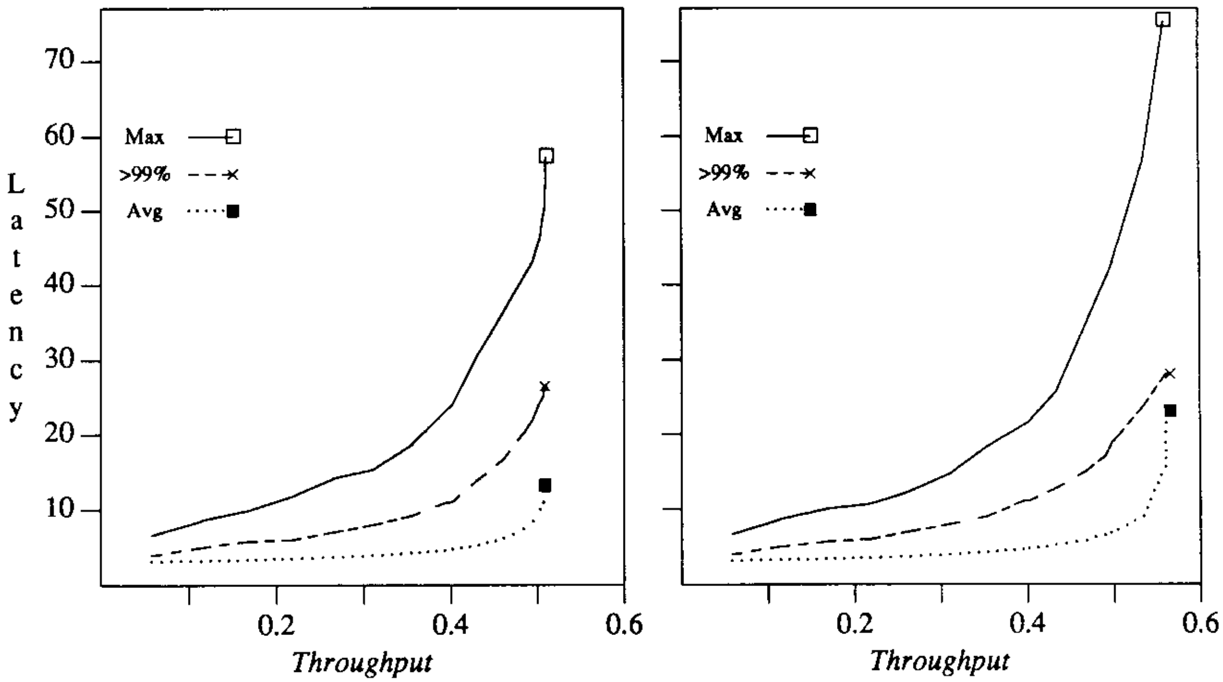
Figure 2: Average latency vs. throughput. Blocking switches. Four packet slots per input port (sixteen total buffer slots in the CBDA switch).

of the three other practical switches. In particular, while the networks composed of FIFO, SAMQ, and SAFC switches saturate at approximately 0.50 throughput, the network composed of DAMQ switches could achieve a throughput of 0.71. The theoretical advantages of a central buffer pool are demonstrated by the superior performance of the network composed of CBDA switches.

We have previously shown that, with 4×4 switches, increasing the size of the FIFO buffers to eight packet slots results in only a small increase in the maximum throughout that the network can support [13]. In particular, even with eight slot buffers, the maximum throughput in the FIFO network is only 80% of the maximum throughput of the DAMQ network with four slot buffers [13]. Hence, maximum network throughput cannot be achieved by simply using larger FIFO buffers; multi-queue buffers, such as DAMQ buffers, must be used.

IV. The Need for Special Support for High-Priority Traffic

The performance of a network can be characterized by the maximum throughput it can support as well as the average packet latency. As mentioned earlier, real-time requirements due to interaction with I/O devices and system-wide exception handling may require that a certain percentage of the traffic will receive faster "service" than normal traffic. Thus, a third important characteristic of the communication network is the *maximum* latency through the network.



4 Packet Slots per Buffer

8 Packet Slots per Buffer

Figure 3: Maximum, 99th percentile, and average packet latency vs. network throughput. FIFO input port buffers.

The problem of high worst-case latency through a network is demonstrated by the results of simulations of a 64×64 Omega network composed of switches with FIFO input port buffers. As shown in Figure 3, the maximum latency through the network may be many times larger than the average latency. Furthermore, if the buffer size is increased from four packets slots to eight packet slots, the ratio of maximum latency to average latency is not improved. Hence, increasing the size of the FIFO buffers is not a solution to the problem of very large worst-case latency through the network.

The simulated maximum latency is a poor measure of "worst case" performance since it is susceptible to statistical anomalies of the simulation. In particular, the simulated maximum latency is sensitive to the initial random number seeds and increases as the number of packets transmitted during the simulation is increased. As a measure of "worst case" performance that is less susceptible to statistical anomalies of the simulation, we have used the "99th percentile latency" which is the minimum of the latencies of the 1% of the packets that received the poorest service (longest latencies) from the network. For the 99th percentile latencies, the result from each simulation run is an integer number of stage cycles. Hence, for low throughputs (and latencies), percentage variations in some 99th percentile latency results were larger than the percentage variations in average latencies. However, the results of the simulation runs were either within one stage cycle or 6% of the 99th percentile latencies reported.

As shown in Figure 3, the 99th percentile latency may be more than twice the average latency, implying that 1% of the traffic will be delayed by more than twice the average network latency (even when the network is *not* in saturation). As discussed above, this situation is not improved by increasing the size of the buffers. As system load (throughput) increases, the average packet latency increases. This degradation in network performance is much more pronounced with respect to the maximum and 99th percentile packet latencies (Figure 3). Network performance in terms of average packet latency begins to significantly deteriorate only when the throughput is increased beyond 0.45, for four slot buffers, and 0.50, for eight slot buffers. However, performance as measured by maximum (or 99th percentile) latency begins to deteriorate with just 0.25 or 0.30 throughput.

	Buffer Type	Throughput									
		0.10		0.20		0.30		0.40		0.50	
>99% max avg	FIFO	4.75	8.10	5.95	11.11	7.78	15.03	10.97	23.77	23.48	45.08
		3.14		3.38		3.79		4.65		9.34	
	SAMQ	5.76	9.80	6.75	13.88	9.00	19.79	12.00	26.74	17.88	39.25
		3.24		3.58		4.09		4.90		6.57	
	SAFC	5.38	10.10	6.73	13.78	8.16	20.06	11.00	21.60	14.38	32.83
		3.22		3.50		3.88		4.42		5.28	
	DAMQ	4.76	7.76	5.67	10.24	7.00	13.74	8.88	16.87	11.11	22.27
		3.14		3.36		3.68		4.16		4.91	
	CBDA	4.39	6.43	5.00	8.11	6.00	9.60	7.00	11.75	8.00	14.40
		2.76		3.29		3.50		3.80		4.19	

Table I: The 99th percentile, maximum, and average latencies vs. network throughput. Four slots per input port (sixteen total slots for CBDA switch). Normal uniform traffic.

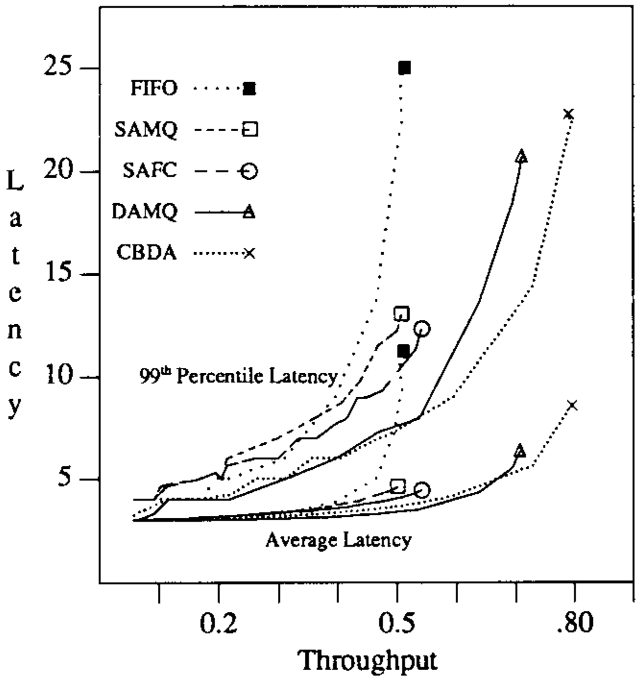
We have previously shown [13] that networks composed of DAMQ, SAMQ, or SAFC switches perform significantly better than networks composed of FIFO switches (Figure 2). As shown in Table I, these three practical switches as well as the CBDA switch perform better than FIFO switches in terms of worst-case packet latency under heavy load (throughput). However, even for these "better" switches the maximum (and 99th percentile) latencies are several times larger than their average latency. Even for the CBDA switch, which attempts to achieve fairness by arbitrating between incoming packets based upon the amount of time spent in the previous switch, the 99th percentile latency is double the average latency at 50% throughput, and the maximum latency is double that. These results indicate that the problem of large variations in the latency of packets, some of which may be "high priority," cannot be solved by using different buffer organizations without special support for the high-priority packets.

V. Hardware Support for High-Priority Traffic

As discussed above, support for non-FIFO handling of packets is not sufficient to significantly reduce the ratio of worst-case to average latencies. In order to obtain better worst-case performance for critical high-priority traffic, such traffic must be so identified, thus allowing the communication hardware to give it priority over normal traffic. We assume that the sender of a packet can mark it "high-priority" by setting a dedicated bit in the packet's header byte. The goal of the system will thus be to continue to provide low average latency for all traffic while also providing relatively low worst-case latencies to a small percentage of high-priority packets [12].

In our simulations, a small percentage (1%, 5%, or 10%) of the packets are randomly marked as "high priority" packets. In addition to collecting measurements of the normal traffic, we collected measurements of the 99th percentile and average latencies of the high-priority packets versus total network throughput. The 99th percentile latency in this case is relative to the high priority traffic only. The success of schemes for supporting high priority packets can be measured by the extent to which the 99th percentile latency of these packets can be kept low, even when the network is heavily loaded.

One way to support high-priority packets is to modify the crossbar arbiter — the controller that determines which input buffer is connected to which output port. The purpose of such a modification is to provide preferential arbitration to high-priority packets. Specifically, when determining the crossbar configuration, the modified arbiter will give priority to those queues which have a high-priority packet at their head. At each cycle, the arbiter first attempts to connect queues with high-priority packets at their heads to output ports, and then connects queues with normal packets at their heads to any remaining available output ports. For the CBDA switch, support for high-priority traffic in the arbiter means that if multiple packets destined for the same output port arrive simultaneously, the high-priority packets will be placed in the queue for that output port ahead of any normal packets. Furthermore, if the number of available slots in the buffer of the CBDA switch is less than the number of senders to the switch, senders with high-priority packets at the head of their queue will be given priority.



	Buffer Type	Throughput				
		0.10	0.20	0.30	0.40	0.50
>99% Latency *pref arbit.* *normal switch*	FIFO	4.00	5.00	5.89	9.34	21.08
		4.75	5.95	7.78	10.97	23.46
	SAMQ	4.55	5.13	7.05	8.65	12.25
		5.76	6.75	9.00	12.00	17.88
	SAFC	4.27	5.15	6.06	7.78	10.21
		5.38	6.73	8.16	11.00	14.38
	DAMQ	3.59	4.00	4.89	6.09	7.63
		4.76	5.67	7.00	8.88	11.11
	CBDA	3.81	4.13	5.00	6.00	7.37
		4.37	5.00	6.00	7.00	8.00

a) The latency of the 5% high priority packets with preferential arbitration.

b) The impact of preferential arbitration — comparison with the 99th percentile latency in normal switches.

Figure 4: Preferential arbitration for high priority packets. Four slots per input port. 5% high-priority packets. Throughputs shown are *total* network throughputs.

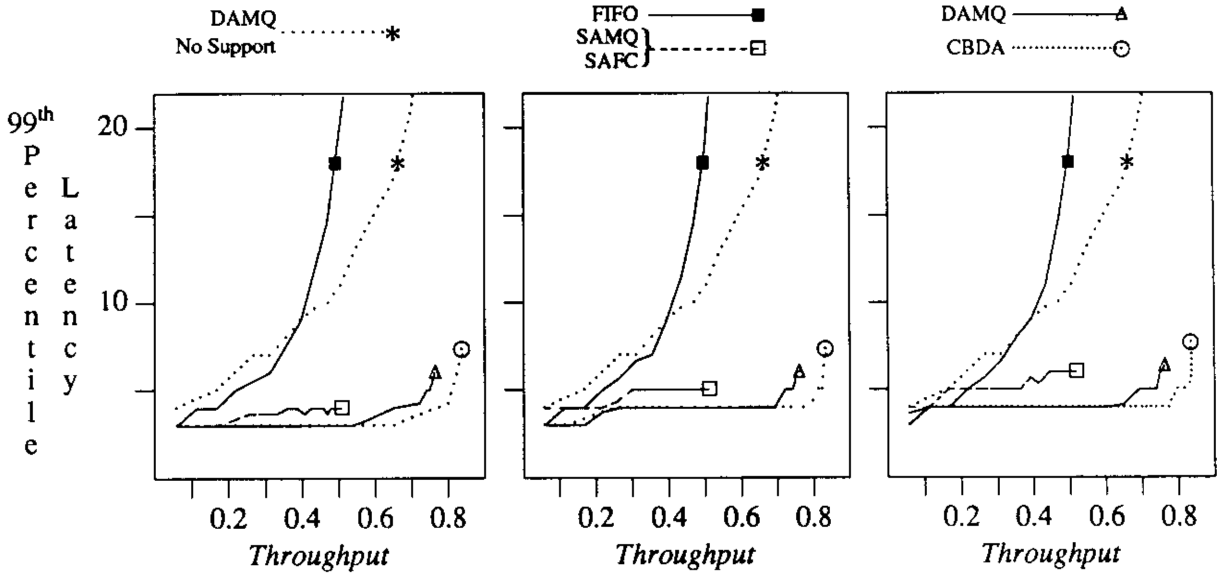
In order to evaluate the effectiveness of the above scheme we ran simulations of the 64×64 Omega network where a randomly selected 5% of all the packets generated were marked as high priority. Figure 4.a compares the 99th percentile latency and the average latency of the high-priority packets only. Figure 4.b compares the 99th percentile latency of a network where the switches provide the preferential arbitration for high-priority traffic to the 99th percentile latency in a network of normal switches. The results in Figure 4.b indicate that, especially for networks with non-FIFO buffers, the modified switches improve high-priority performance over a system with no special support. However, as shown in Figure 4.a, the 99th percentile latencies are still much worse than the average latencies, especially under high network load. In general, support for high-priority packets in the arbiter is not sufficient since the high-priority packets often wait in queues behind low-priority packets and receive preferential treatment for only the brief time they spend at the heads of queues.

In the previous section we have shown that there is a need for special support for high-priority packets. In this section, we have already shown that under moderate and high throughputs, even with preferential arbitration of high-priority packets in the switch, the 99th percentile latencies of these packets are still several times higher than the average latencies. In the rest of this section we investigate switch designs that provide better support for high-priority packets. Alternatives designs are presented in the order of increasing complexity and hardware cost. Subsection A introduces the use of dedicated queues for high-priority packets within the buffers used for all packets. Multiple dedicated queues for high-priority packets, one queue per output port, are discussed in Subsection B. Subsection C evaluates the use of separate dedicated buffers for the high-priority packets.

A. Dedicated Queues for High-Priority Traffic

The SAMQ, SAFC, DAMQ, and CBDA switches achieve low average latencies by employing separate queues to avoid having packets which are destined for different output ports block each other. A possible extension of this idea is to use an additional queue at each input port dedicated to high-priority packets. This should allow high-priority packets to be routed through the switch as soon as they arrive in preference to any normal packets waiting to be transmitted out of the same output port. This idea cannot be applied to a FIFO buffer since it has only a single queue; FIFO switches with the priority arbitration scheme discussed above are used in the following comparisons.

Implementing a dedicated queue for high-priority packets in the SAMQ and SAFC buffers involves adding additional storage (at least one packet slot) as well as a slight increase in the complexity of the arbitration circuitry. Since each queue in a SAMQ or SAFC buffer has at least one packet slot, the minimum size SAMQ or SAFC buffer in a 4×4 switch with support for high-priority traffic is five packet slots. On the other hand, adding the dedicated queue to the DAMQ buffer does not require any additional buffer storage — the available storage is dynamically shared between queues. A few (two or three) more control registers are needed to manage an additional linked list. In order to present a fair comparison of the performance of networks with the different buffer types, many of the results presented in this paper were obtained for a total buffer size of five packet slots per input port. For the CBDA switch, special queues for high-priority traffic mean that the switch maintains, in its central buffer, a separate FIFO queue for each priority level at each output port. For each output port, packets are sent from the queue of normal packets only if the high-priority queue is empty. Furthermore, if the number of available slots in the buffer of the CBDA switch is less than the number of senders to the switch, senders with a non-empty high-priority queue are given priority.



a) 1% Packets High-Priority b) 5% Packets High-Priority c) 10% Packets High-Priority

Figure 5: The impact of a dedicated queue for high-priority packets at each input port. Shown are the 99th percentile latencies of high-priority packets vs. total throughput. Five slots per input port (a total of twenty slots in the CBDA switch).

We have simulated 64×64 Omega networks constructed out of switches which include a dedicated queue for the high-priority packets at each input port. The 99th percentile latencies of the high-priority packets for networks with 1%, 5% and 10% high-priority traffic are shown in Figure 5. For comparison, the figure also shows the 99th percentile latencies of a network with five-slot DAMQ buffers with no support for high-priority packets. The DAMQ buffer was chosen since, as shown in Table I, this buffer results lower 99th percentile latencies than the other practical switches. These simulation results demonstrate the benefits of dedicated high-priority queues. Networks composed of the three practical switches which can support a separate high-priority queue achieved significantly lower 99th percentile latencies compared to those networks without separate high-priority queues. Furthermore, the 99th percentile latencies for high-priority packets are lower than the <u>average</u> latencies with switches that do not have support for high priority traffic (Figure 2).

As we have previously discovered with respect to the average latencies of general network traffic [13], switches with DAMQ buffers outperform the SAMQ and SAFC switches by achieving lower

99th percentile latencies for high-priority traffic. The DAMQ switch is better at handling high-priority packets for the same reason it provides lower average latency for normal traffic — while only a fraction of the buffer space of the SAMQ and SAFC buffers is available to any single packet arriving at an input port (one fifth, in this case), the entire buffer is available to packets arriving at the DAMQ switch. Thus, the DAMQ switch better utilizes the available space, blocks incoming packets less often, is capable of handling higher throughputs, and delivers packets with lower latencies.

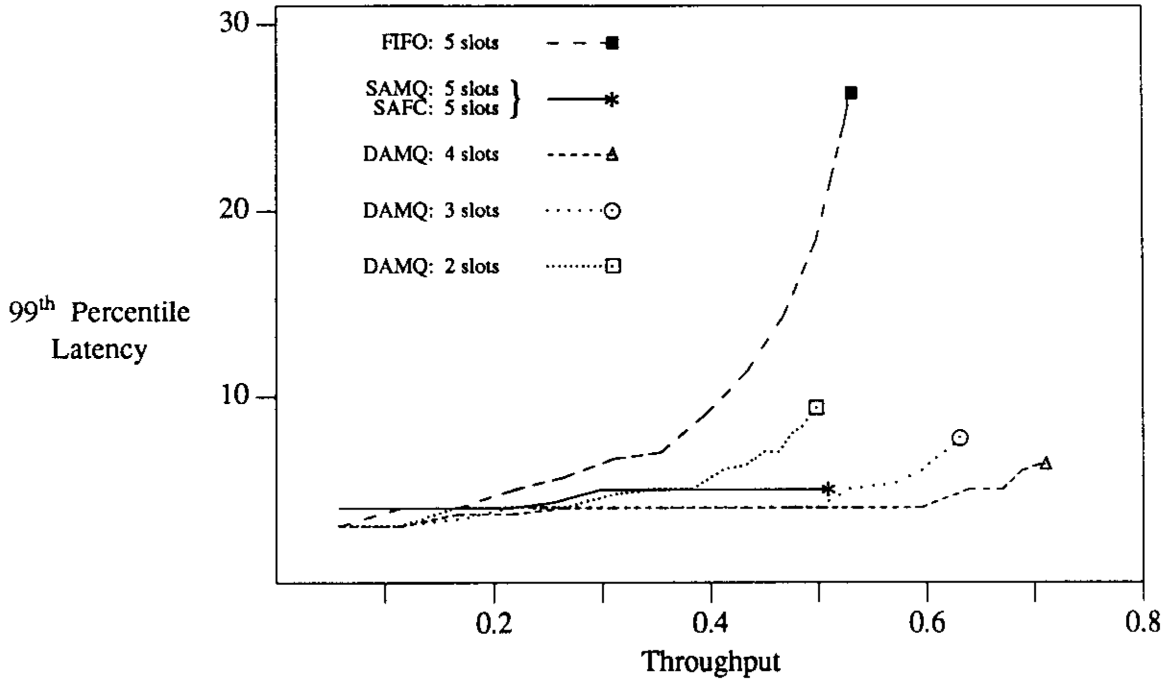


Figure 6: The 99th percentile latency of high-priority traffic vs. total network throughput. FIFO, SAMQ, and SAFC buffers with five packet slots per input port, DAMQ buffers with two, three and four packet slots per input port. 5% high-priority traffic. Dedicated queues for high-priority packets.

The advantages of the DAMQ buffer are further demonstrated by the results shown in Figure 6. The 99th percentile latencies of a network composed of DAMQ switches with two, three, and four slots are compared with the 99th percentile latencies for networks composed of FIFO, SAMQ, and SAFC switches with five slots. The DAMQ switch with only two slots significantly outperforms the FIFO switch with five slots. The DAMQ switch with three slots outperforms the SAMQ and SAFC switches with five buffer slots. We have now shown that DAMQ buffers achieve higher performance than other practical buffers for both normal traffic [13] and high-priority traffic. Thus, for the rest of this paper, DAMQ buffers with dedicated queues for high-priority traffic, will serve as a benchmark against which alternative designs will be compared.

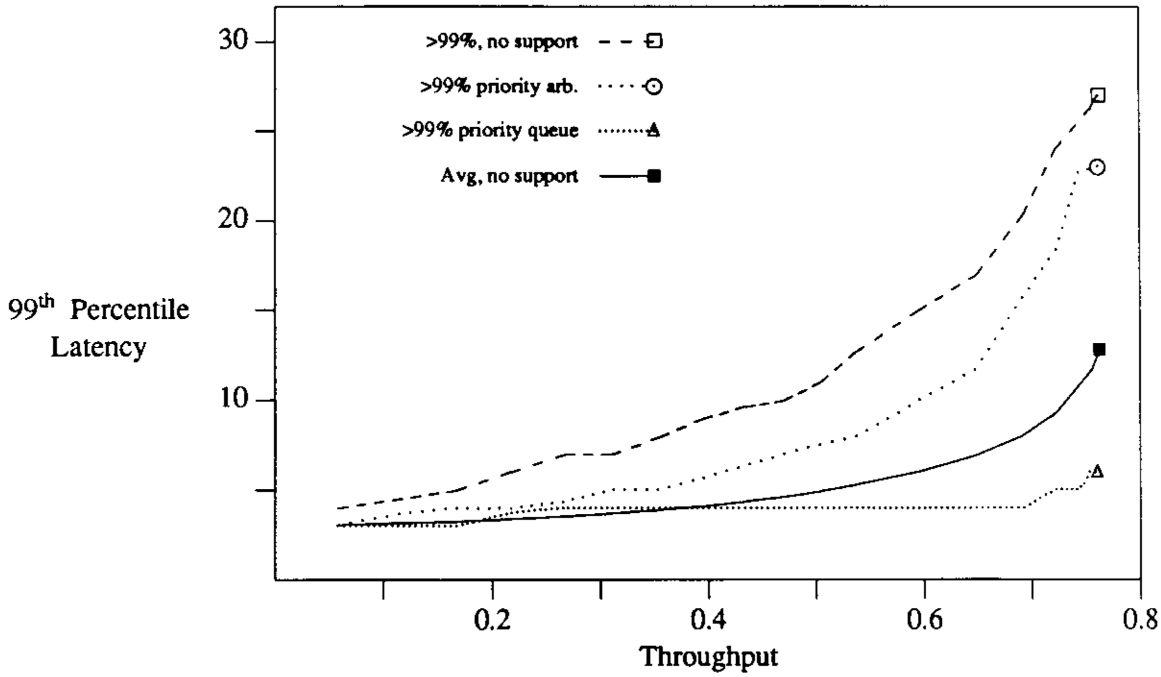
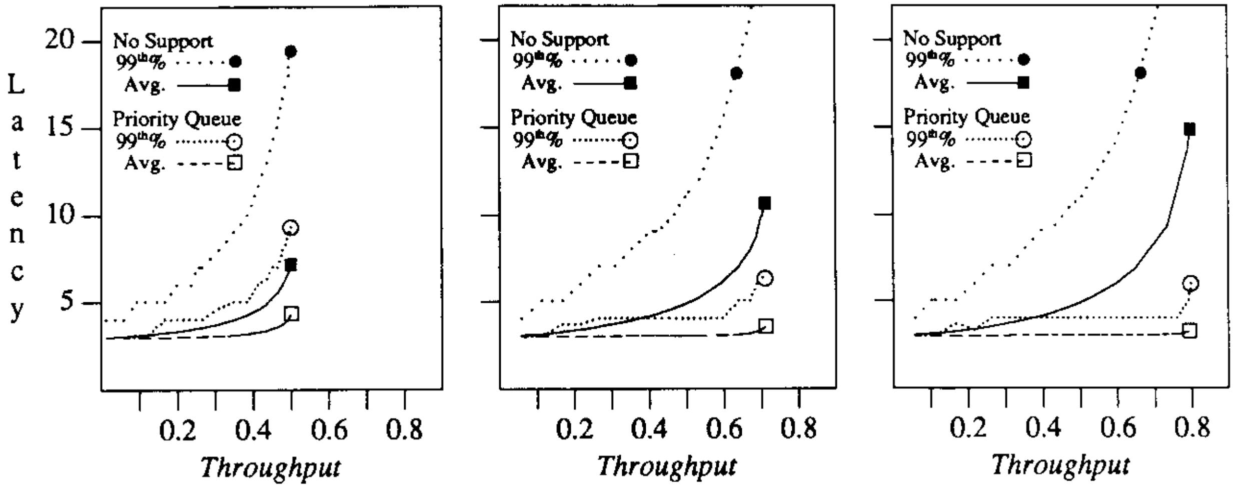


Figure 7: The benefits of increasing levels of support for high-priority traffic. The average latency for normal traffic and the 99th percentile latency for high-priority traffic without support, with priority arbitration and with priority queues vs. total network throughput. DAMQ buffers with five slots per input port. 5% high-priority traffic.

Figure 7 shows the 99th percentile latency of high-priority packets in a DAMQ network with high-priority queues, with priority arbitration, and with no support for high-priority traffic. The average latency in a DAMQ network with no high-priority support is also shown. In all cases the buffers have five packet slots and 5% of the packets are high priority. With dedicated queues for high-priority traffic, the 99th percentile latency for this traffic is stable at near the minimum network latency for a wide range of throughputs. With the dedicated queues, at medium and high throughputs, the 99th percentile latency of the high-priority traffic is actually less than the average latency for normal traffic. An interconnection network with switches using DAMQ buffers with a high-priority queue can thus provide low worst-case latencies for high priority packets even under network loads approaching saturation.

Figure 8 shows the impact of buffer size on the effectiveness of using dedicated queues for high-priority traffic with DAMQ buffers. For all buffer sizes, the dedicated queues result in significantly better performance for the high-priority packets than in a network where there is no special support for such packets. With very small buffers (two packet slots), even with dedicated queues, the 99th percentile latency of the high-priority packets increase significantly as the total network throughput increases



a) 2 Slots per Buffer

b) 4 Slots per Buffer

c) 6 Slots per Buffer

Figure 8: The impact of varying the buffer size on the effectiveness of dedicated queues for high-priority traffic with DAMQ buffers. For varying total network throughput we show the average and 99th percentile latencies of normal traffic in a network without support for high-priority packets. We also show the average and 99th percentile latencies for the 5% high priority traffic in a network with dedicated queues for high-priority packets.

above 0.3. Furthermore, this 99th percentile latency of the high-priority traffic is always higher than the average latency in a network without the dedicated queues. The reason for this rather poor performance with two slot buffers is that at most two queues can exist in the buffer at any instant.

The maximum *total* throughput of the network increases significantly as the buffer size is increased from two to four slots and then six slots (Figure 8). With the larger buffers, more queues can exist in the buffer simultaneously and, at any point in time, the probability of a full buffer is lower than with two slot buffers. Hence, with dedicated queues for high-priority packets, it is much less likely that such a packet will be blocked due to a full buffer in the next stage. As a result, with four slot and six slot buffers with dedicated queues for high-priority traffic, the 99th percentile latency of the high-priority packets remains very low (4 stage cycles) even as the network throughput increases up to 0.6. Furthermore, this 99th percentile latency of the high-priority traffic is always lower than the average latency in a network without the dedicated queues.

B. Multiple Dedicated Queues for High-Priority Traffic

The previous subsection proposes support for high-priority traffic using, at each input port, a DAMQ buffer with a separate queue for high-priority packets. For normal traffic, DAMQ buffers achieve superior performance to FIFO buffers by providing a separate queue at each input port for each output port [13]. Hence, it should be possible to achieve better performance for high-priority traffic by providing, at each input port, a dedicated queue for high-priority traffic for each output port, in addition to the separate queues for normal traffic. With this scheme, for a 4×4 switch, each buffer will contain *eight queues* instead of the five queues proposed in the previous subsection. The additional hardware required by this scheme may be worthwhile for applications where minimizing worst-case latency is particularly critical. SAMQ and SAFC switches with multiple dedicated queues for high-priority traffic require even more hardware since additional buffer storage is needed for each additional queue. Since, in addition, DAMQ switches with high-priority queues have already been shown to achieve superior performance compared to SAMQ and SAFC switches, only DAMQ switches are considered in this subsection.

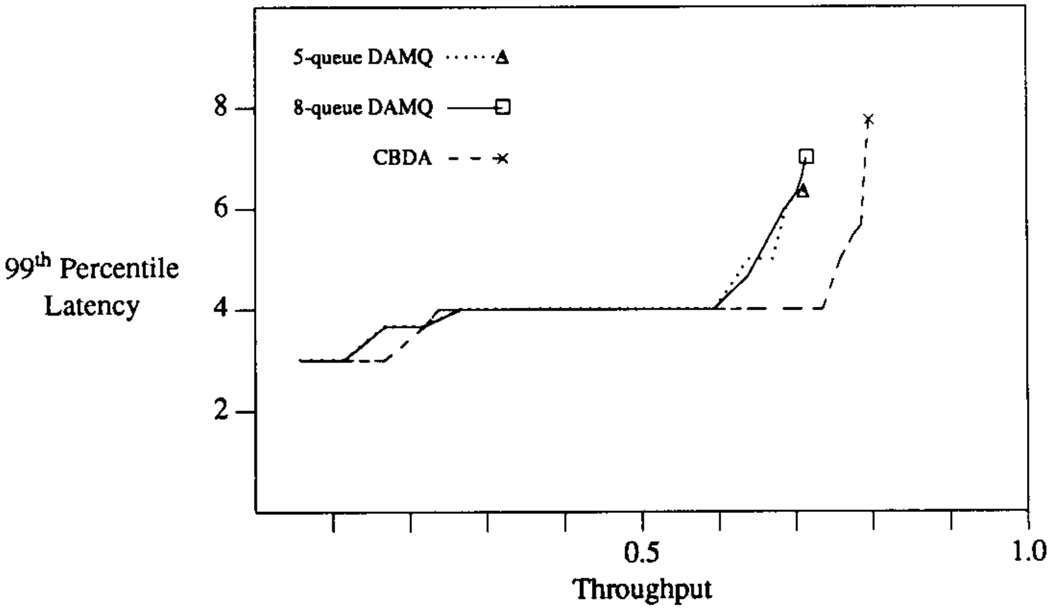


Figure 9: Five-queue DAMQ switches versus eight-queue DAMQ switches. The 99th percentile latency of high-priority traffic vs. the total network throughput. Four packet slots per input port. 5% high-priority traffic.

Figure 9 compares the 99th percentile latencies of the high-priority traffic with eight-queue and five-queue DAMQ switches, varying the total throughput and keeping the percentage of high-priority packets constant at 5%. The performance of a network with CBDA switches is shown as a reference point for the level of performance that can be achieved. Under these conditions, the two DAMQ switches behave almost identically. The reason for this is that multiple queues are beneficial only when there are more than one packet in the buffer simultaneously. With only 5% high-priority packets it is rare that more than one high-priority packet is in a buffer at any instant.

In a network constructed from switches that support high-priority traffic, it is useful to know up to what level (proportion) of high-priority traffic will the separate queues be able to maintain good worst-case performance, i.e., low 99th percentile latency, for the high priority packets. To answer this question we have simulated the 64×64 Omega network of four-slot DAMQ buffers under a constant total network throughput of 50% with varying proportions of high-priority traffic. It should be noted that without support for high-priority traffic, such a network of DAMQ switches achieves an average latency of 4.9 stage cycles and a 99th percentile latency of 11.1 stage cycles (Table I).

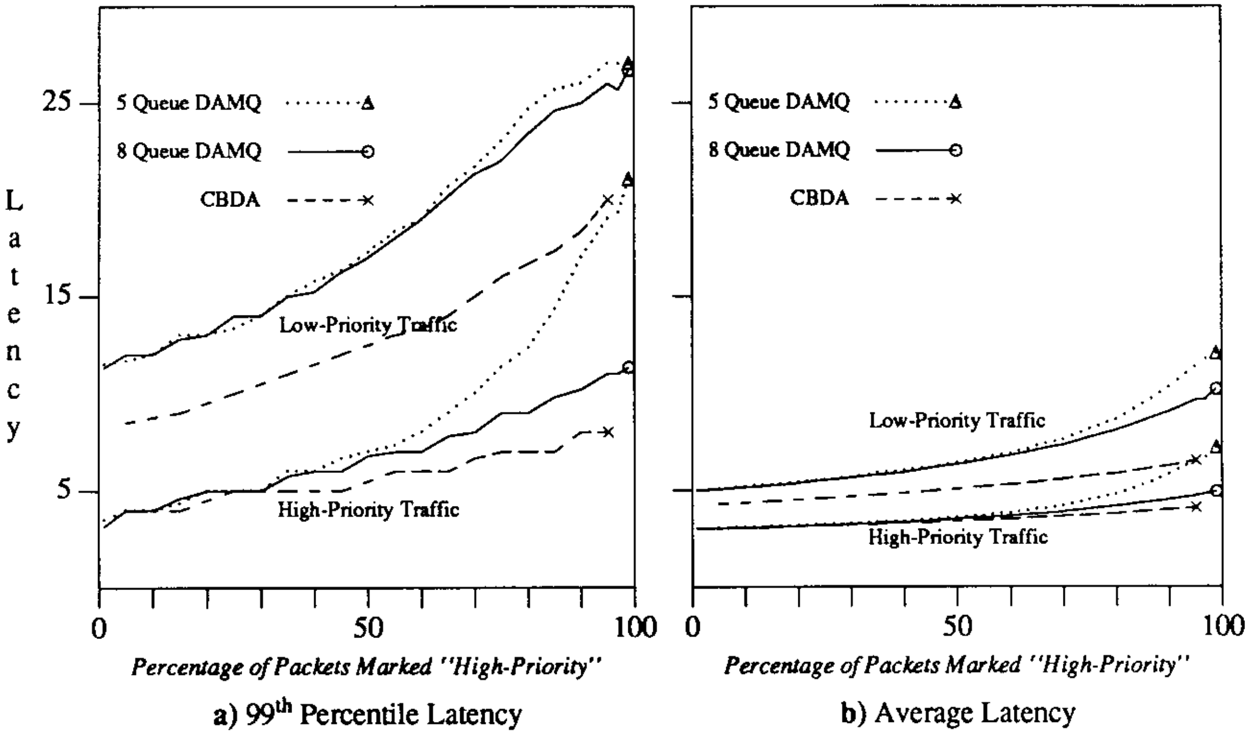


Figure 10: The impact of varying the percentage of high-priority packets. Shown are 99th percentile latencies and average latencies of high-priority and low-priority traffic for a fixed total network throughput of 50%. Four packet slots per input port. Five-queue DAMQ switches, eight-queue DAMQ and CBDA switches.

Figure 10 shows the 99th percentile and average latencies of both the high-priority and low-priority (normal) traffic for a varying percentage of high-priority traffic. For a low percentage of high-priority packets, the five-queue DAMQ buffers perform well, i.e., result in low 99th percentile latency for the high priority traffic. With up to 18% high priority packets, the 99th percentile latency for such packets is below the overall average latency for a standard network using switches with DAMQ buffers (Table I).

The average latencies and 99th percentile latencies of the low priority packets (Figure 10) indicate that, for low percentages of high-priority packets, the support for high-priority traffic has little impact on network performance with respect to normal traffic. Specifically, as the percentage of high-priority traffic varies from 0 to 10%, the change in average latency of the low priority traffic is negligible. For 5% high-priority packets the increase in the 99th percentile latency of low-priority packets is only 4%, while for 10% high-priority packet the increase is 8%.

As the proportion of high-priority packets increases beyond 75%, the 99th percentile latency of a network with the five-queue DAMQ buffers is *larger* than 11.1 stage cycles, i.e., larger than the 99th percentile latency for the network composed of switches without support for high-priority traffic (Table I). The reason for this behavior is that the DAMQ switch has four queues at each input port for normal traffic, but only a single queue for high-priority traffic. Hence, with respect to high-priority traffic, the DAMQ switch with a high-priority queue performs as a FIFO switch. As shown in Figure 3.a, at a throughput of 0.37, the 99th percentile latency of a network with FIFO switches is 10 stage cycles. With the network of DAMQ switches with high priority queues, 0.50 network throughput with 75% high priority traffic, the absolute throughput of high priority traffic is 0.375. Since part of each input buffer is occupied by normal packets, it is not surprising that the high priority traffic receives worse "service" (higher 99th percentile latency) than the packets in the FIFO network with a throughput of 0.37.

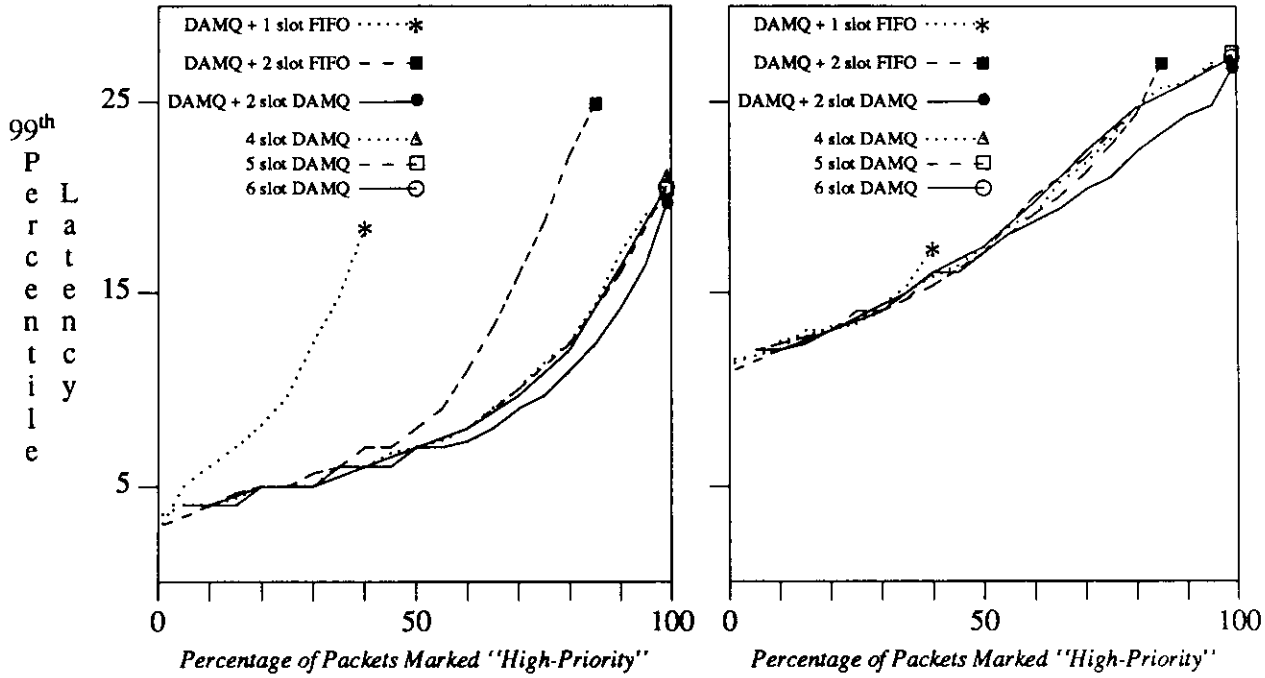
Figure 10 shows that when more than 50% of the packets are high-priority, the eight-queue DAMQ switch performs significantly better than the five-queue DAMQ buffer, maintaining a relatively low 99th percentile latency for the high-priority traffic. Furthermore, the improved performance with respect to high-priority packets does *not* come at the expense of the performance for low-priority packets. If the high-priority traffic is less than 50% of the total network load, there are no benefits to using the eight-queue DAMQ buffers. Eight-queue DAMQ buffers should be considered only if the *majority* of the packets are high priority. In this latter case, a better solution might be to use five-queue DAMQ buffers with a single queue dedicated to the low-priority traffic and four queues dedicated to high-priority traffic.

C. Dedicated Buffers for High-Priority Traffic

In the previous two subsections we have considered the use of one or more dedicated queues for high-priority packets and demonstrated the advantages of using DAMQ buffers that include such queues. With the DAMQ buffers, the normal packets and high-priority packets share the available buffer storage, even though they are organized in separate queues. Another design option for supporting high-priority traffic is the use of two independent buffers at each input port — one for normal packets and one for high-priority packets. Examples of such a scheme include the use of an *n* slot DAMQ buffer for normal packets together with an *m* slot FIFO buffer for high-priority packets, or an *n* slot DAMQ buffer for normal packets with an *m* slot DAMQ buffer for high-priority packets. The motivation for such a scheme is to isolate the high-priority buffering from the normal packet buffering. This is done to prevent a situation where a high-priority packet cannot be forwarded to the next stage due to a full buffer containing some normal packets. As with high-priority queues, for a synchronous network (Section III), the high-priority traffic does not compete for link bandwidth with the low-priority traffic since, whenever there is a conflict, the high-priority traffic is always given priority over the low-priority traffic. Thus, with dedicated buffers for high-priority traffic, network performance with respect to such traffic will be independent of low priority traffic.

We have evaluated schemes for using dedicated buffers for high-priority packets by comparing their performance, with respect to high-priority traffic, to the performance of a network of DAMQ switches where the buffer storage is ***shared*** between the high-priority traffic and the normal traffic (Subsection V.A). For varying percentage of high-priority traffic, the comparisons were done for a fixed total network throughput of 50% (at which, as shown in Figure 2, a network of switches with four-slot FIFO buffers is already in saturation). We have previously shown that the five-queue DAMQ buffer with the dedicated high-priority queue provides good support at a low implementation cost for high-priority traffic. At 50% total network throughput, the network composed of switches using these buffers is quite heavily loaded, so contention for buffer storage between the high-priority and low-priority traffic might be expected. Hence, this network serves as a good benchmark, relative to which dedicated buffer schemes can be compared.

We have considered several input port organizations based on a four-slot four-queue DAMQ buffer for normal packets and a dedicated buffer for high-priority packets. DAMQ buffers for normal packets are assumed since they have been shown to have significant performance benefits for normal traffic [13]. In choosing the buffer for high-priority traffic, we must consider the hardware cost for additional storage and control, as well as performance. Single-slot FIFO buffers are a low-cost choice for the dedicated high-priority buffers. Multi-slot high-priority buffers can improve the performance of dedicated high-priority buffer schemes, but require more hardware. In particular, we have considered the use of dedicated two-slot FIFO buffers and two-slot DAMQ buffers for high-priority traffic.



a) High Priority Packets

b) Normal Packets

Figure 11: The impact of dedicated buffers for high-priority packets. The 99th percentile latency of high-priority and of normal packets versus the percentage of high-priority traffic. The total network throughput is 0.5. The shared buffers are five-queue DAMQ buffers with four, five, and six packet slots. The switches with dedicated buffers for high-priority traffic consists of four-queue four packet slot buffers for normal traffic and either a one slot FIFO, a two slot FIFO, or a two slot DAMQ buffer for high-priority traffic.

Figure 11 shows the performance of networks with the dedicated high-priority buffer schemes described above as well as the performance of networks with DAMQ buffers which include dedicated queues for high-priority traffic but where storage is shared by all traffic. The simulation results presented are for a fixed total network throughput of 50%. For varying percentage of high-priority traffic, Figure 11.a shows the 99th percentile latency of high-priority packets, while Figure 11.b shows the 99th percentile latency of normal packets. Since none of the three shared DAMQ buffers is in its saturation region, they all result in identical performance. With a one slot dedicated buffer, handling of high-priority packets is poor, despite using more total buffer space than the shared four-slot DAMQ buffers. Performance with respect to the normal packets is identical with all the schemes for the proportion of high-priority packets of interest (under 30%).

At a low percentage of high-priority traffic (under 30%), the shared DAMQ buffers deliver equal performance to the performance of the two slot dedicated buffer schemes, despite the fact that the shared buffers are also supporting the total network throughput of 0.5. Since DAMQ buffers are already used to achieve high performance for normal traffic, very little additional hardware is required to support dedicated queues. Dedicated two-slot buffers require significantly more hardware for both storage and control. In addition, the two buffers schemes are less likely to adapt well to a variety of conditions (such as very high throughput with *no* high-priority packets) due to the *static* partitioning of the available storage. Thus, dedicated buffers for high-priority traffic should be considered only if there is some reason why DAMQ buffers cannot be used. Under these conditions, it is useful to know that similar performance, with respect to high-priority traffic, can be achieved using this alternative approach.

VI. Summary and Conclusions

In order to use large multiprocessors and multicomputers for real-time applications, it must be possible to guarantee, with a high degree of confidence, that high-priority traffic can be transmitted through the network with low specified latency. With conventional interconnection networks used in multiprocessors and multicomputers, the worst-case latency of traffic through the network can increase dramatically as the load on the network increases. This may prevent the use of these interconnection networks for critical real-time applications or force their use with very low utilization (and thus high cost/performance ratio) in order to guarantee low maximum latency.

For interconnection networks composed of small $n \times n$ switches, we have shown that simply increasing the size of conventional buffers in the switches does not result in improved performance for high priority packets. There is thus a fundamental need for new buffer organizations which support high-priority packets. We have developed a technique for efficiently supporting high-priority traffic, while maintaining good performance for normal traffic. Our scheme is based on a small modification of the *dynamically allocated multi-queue* (DAMQ) buffer[13]. This buffer provides high performance for normal traffic due to its ability to forward packets in non-FIFO order and handle variable-length packets.

The DAMQ buffer is amenable to efficient VLSI implementation [6] and provides higher saturation throughput and lower average latencies than several common buffer organizations. The modifications to the DAMQ buffer in order to support high-priority traffic involve a few additional control registers and somewhat more complex arbitration of the crossbar switch. Overall these modifications are expected to require only a small percentage increase in total buffer area.

We have evaluated alternative approaches to providing support for high-priority packets in $n \times n$ switches. This evaluation was based on implementation complexity and simulation studies of a multistage interconnection network transmitting packets with two levels of priority. Our simulations have demonstrated five key points. 1) In a conventional network with a single priority level for all packets, worst-case latency for packets can be several times higher than average latency. Hence, there is a need to identify and provide preferential treatment to those packets for which fast service is particularly important. 2) Using the priority of packets as the determining factor in arbitrating contention within each switch does not provide sufficient support for high-priority traffic. Such arbitration does not reduce the 99th percentile latency of the high-priority packets to the level of average normal packet latency, even under moderate network load. 3) As long as the proportion of high-priority traffic is low, dedicated queues for high priority traffic reduce the 99th percentile latency of the high-priority packets to the level of the average latency for normal traffic under a wide range of network throughputs. 4) When the proportion of high-priority traffic is large and the network is heavily loaded, multiple dedicated queues for high-priority traffic can reduce the 99th percentile latency relative to the single dedicated queue approach. However, this performance advantage is marginal and the associated implementation cost is very high. 5) Dedicated buffers for the high-priority traffic can provide the same performance advantage as dedicated queues in shared DAMQ buffers. However, since DAMQ buffers are needed to maximize performance for normal traffic, the implementation cost for dedicated buffers is much higher than the cost of adding dedicated queues to DAMQ buffers. Hence, there is no reason to use dedicated buffers.

Our results indicate that the DAMQ buffer with a single dedicated queue for high-priority packets provides support for high-priority traffic which is superior to the support provided by alternate switch designs based on a dedicated high-priority queue. This resulting network performance, with respect to high-priority traffic, is very close to the performance of a network based on "ideal" switches for which a practical implementation is not feasible. Hence, given the low hardware overhead of the scheme based on a DAMQ buffer with a single high-priority queue, it is clearly the preferable option.

Acknowledgements

Our work was aided and enhanced by other members of the ComCoBB group: Hsin-Chou Chi, Tiffany Frazier, and Yoshio Turner.

References

1. Athas, W. C. and Seitz, C. L., ''Multicomputers: Message-Passing Concurrent Computers,'' *Computer* **21**(8), pp. 9-24 (August 1988).

2. Crowther, W., Goodhue, J., Gurwitz, R., Rettberg, R., and Thomas, R., ''The Butterfly Parallel Processor,'' *IEEE Computer Architecture Newsletter*, pp. 18-45 (September/December 1985).

3. Dally, W. J. and Seitz, C. L., ''The Torus Routing Chip,'' *Distributed Computing* **1**(4), pp. 187-196 (October 1986).

4. Dally, W. J., ''Virtual-Channel Flow Control,'' *17th Annual International Symposium on Computer Architecture*, Seattle, WA (May 1990).

5. Dias, D. M. and Jump, J. R., ''Packet Switching Interconnection Networks for Modular Systems,'' *Computer* **14**(12), pp. 43-53 (December 1981).

6. Frazier, G. L. and Tamir, Y., ''The Design and Implementation of a Multi-Queue Buffer for VLSI Communication Switches,'' *International Conference on Computer Design*, Cambridge, MA, pp. 466-471 (October 1989).

7. Gottlieb, A., Grishman, R., Kruskal, C. P., McAuliffe, K. P., Rudolph, L., and Snir, M., ''The NYU Ultracomputer - Designing an MIMD Shared Memory Parallel Computer,'' *IEEE Transactions on Computers* **C-32**(2), pp. 175-189 (February 1983).

8. Kumar, M. and Jump, J. R., ''Performance Enhancement in Buffered Delta Networks Using Crossbar Switches and Multiple Links,'' *Journal of Parallel and Distributed Computing* **1**(1), pp. 81-103 (1984).

9. Lawrie, D. H., ''Access and Alignment of Data in an Array Processor,'' *IEEE Transactions on Computers* **C-24**(12), pp. 1145-1155 (December 1975).

10. McMillen, R. J. and Siegel, H. J., ''The Hybrid Cube Network,'' *Distributed Data Acquisition, Computing, and Control Symposium*, pp. 11-22 (December 1980).

11. Stevens, K. S., Robinson, S. V., and Davis, A. L., ''The Post Office - Communication Support for Distributed Ensemble Architectures,'' *The 6th International Conference on Distributed Computing Systems*, Cambridge, MA, pp. 160-166 (May 1986).

12. Tamir, Y. and Frazier, G. L., ''Support for High-Priority Traffic in VLSI Communication Switches,'' *9th Real-Time Systems Symposium*, Huntsville, AL, pp. 191-200 (December 1988).

13. Tamir, Y. and Frazier, G. L., ''High-Performance Multi-Queue Buffers for VLSI Communication Switches,'' *15th Annual International Symposium on Computer Architecture*, Honolulu, Hawaii, pp. 343-354 (May 1988).

14. Whitby-Strevens, C., ''The Transputer,'' *12th Annual Symposium on Computer Architecture*, Boston, MA, pp. 292-300 (June 1985).

15. Yoon, H., Lee, K. Y., and Liu, M. T., ''Performance Analysis of Multibuffered Packet-Switching Networks in Multiprocessor Systems,'' *IEEE Transactions on Computers* **39**(3), pp. 319-327 (March 1990).