9th Real-Time Systems Symposium
Huntsville, Alabama, pp. 191-200, December 1988.

SUPPORT FOR HIGH-PRIORITY TRAFFIC IN VLSI COMMUNICATION SWITCHES†

Yuval Tamir and Gregory L. Frazier
Computer Science Department
University of California
Los Angeles, California 90024
U.S.A.

Abstract

Both multistage interconnection networks used in multiprocessors and direct networks used in multicomputers are composed of small $n \times n$ switches. The design of these switches is of critical importance for achieving high-bandwidth low-latency interprocessor communication. Interprocessor traffic generated by devices that must meet real-time requirements as well as certain other system activities, such as exception handling, may require particularly low communication latency. Special support for such high-priority traffic may thus be necessary 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 that provide non-FIFO handling of messages. 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.

I. Introduction

High-bandwidth low-latency communication between processors is critical to the ability of multiprocessors and multicomputers to achieve high performance by exploiting parallelism. The desirable characteristics of communication networks used for interprocessor communication include the maximum possible throughput (bandwidth) of data through the network and a minimum latency (delay) when sending data between processors. For any given traffic pattern, the achievable throughput of the network is significantly lower than the maximum possible network throughput and the average latency for packets is higher than the minimum possible latency through the network. Furthermore, not all packets are transmitted through the network with the same latency; in a large network some packets may require several times the average latency to reach their destination.

Multiprocessor and multicomputer systems which are connected to input/output devices that interact with the outside world occasionally require particularly fast communication with different parts of the system. The broadcasting of certain types of system-wide events, such as the initiation of global system rollback, may also require preferential handling by the network. Real-time requirements due to interaction with I/O devices and system-wide 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.

Multiprocessors with a large number of nodes (e.g. greater than 64) use multistage interconnection networks composed of a large number of small $n \times n$ switches (typically, $2 \le n \le 10$) for communication [1, 3]. Similarly, communications through point-to-point dedicated links in multicomputers [7, 10] rely on communication coprocessors with a small number of ports [2, 8] that basically function as small $n \times n$ switches with $n-1$ ports connected to other nodes, and one bidirectional port connected to the local application processor. The design of high-performance small $n \times n$ switches is thus 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.

This paper deals with the design and implementation of small $n \times n$ VLSI communication switches with special support for high-priority traffic. Each switch takes packets arriving at its input ports and routes them to its output ports. As long as 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. As the throughput goes up, so does the probability of conflict. When two packets destined for the same output port arrive at different input ports of a switch at approximately the same time, they cannot both be forwarded immediately. Only one packet can be

† This research is supported by Rockwell International and the State of California MICRO program.

transmitted through an output port at a time, and hence one of the two packets must be stored at the node for later transmission. 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 results reported in this paper were produced as part of the UCLA ComCoBB project. The goal of the ComCoBB (**Com**munication **Co**processor **B**uilding-**B**lock) project is to design and implement a single-chip high-performance communication coprocessor for use in VLSI multicomputer systems. The ComCoBB chip is, in part, a small $n \times n$ switch, and the problem of designing an efficient buffering scheme for it with support for high-priority traffic had to be faced early on in the project. We have developed a new type of buffer for small $n \times n$ switches, called a *dynamically-allocated multi-queue* (DAMQ) buffer [9], 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. While the DAMQ buffer was originally developed for use in a multicomputer communication coprocessor, it is equally useful for multistage networks and it is in that context (of a multistage network) that the buffer will be evaluated.

In the next section we discuss some of the issues in designing $n \times n$ switchs for interconnection networks. We include a brief description of the DAMQ buffer and its use for normal traffic. We then show the need for hardware support for high-priority traffic. In Section III we present several approaches to providing support for high-priority traffic and focus on the use of a slightly modified DAMQ buffer. The use of the DAMQ buffer is evaluated using event-driven simulation to compare its performance with that of several alternative buffer configurations in the context of a multistage interconnection network.

II. The Design of Switches for Packet-Switching

In an $n \times n$ switch, packets that cannot be immediately forwarded to an output port need to be buffered within the switch. The buffers must be able to accept simultaneous arrival of packets from all of the input ports, while at the same time transmitting packets through all of the output ports. 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. Communication efficiency can be significantly increased if *virtual cut-through* [4] is supported so that the switch does not wait for a complete packet to arrive before beginning to forward it. In order to minimize waste of the available communication bandwidth, variable-length packets must be supported. Efficient buffer storage utilization requires the ability to allocate storage for variable-length packets while minimizing internal and external fragmentation [9].

In order to provide low-latency communication for high-priority traffic, every packet must be marked as either *normal* or *high priority*. Ideally, if there is a high-priority packet and a normal packet destined for the same output port, the high-priority packet will always be transmitted first. If this requirement is met and the buffers are infinite, a high-priority packet will be delayed by a normal packet only if when the high-priority packet arrives at the switch, transmission of the normal packet destined to the same output port is already underway. Under these conditions, if the maximum size of packets is relatively small, it is expected that normal traffic load (normal packets) will have minimal effect on network performance for high-priority traffic.

A. Packet Buffer Design for Normal Traffic

In a previous paper [9] we discussed alternative designs 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. The main design decisions include: (1) the location of the buffers — a centralized 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.

The implementation of a centralized buffer pool is difficult since all the input and output ports may need to access the buffer simultaneously. Furthermore, the control must be capable of rapidly (within one or two clock cycles) allocating memory for variable size packets while minimizing internal and external fragmentation [9]. Output buffering is undesirable for similar reasons — the need to store 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 [9].

With input port buffering [2, 6] only one write port is needed since only one packet at a time arrives at the input port. If the buffer is managed as a FIFO (first-in-first-out) queue [2, 6], variable length packets are easily handled, thus avoiding the complex memory allocation problems mentioned above. The problem with FIFO

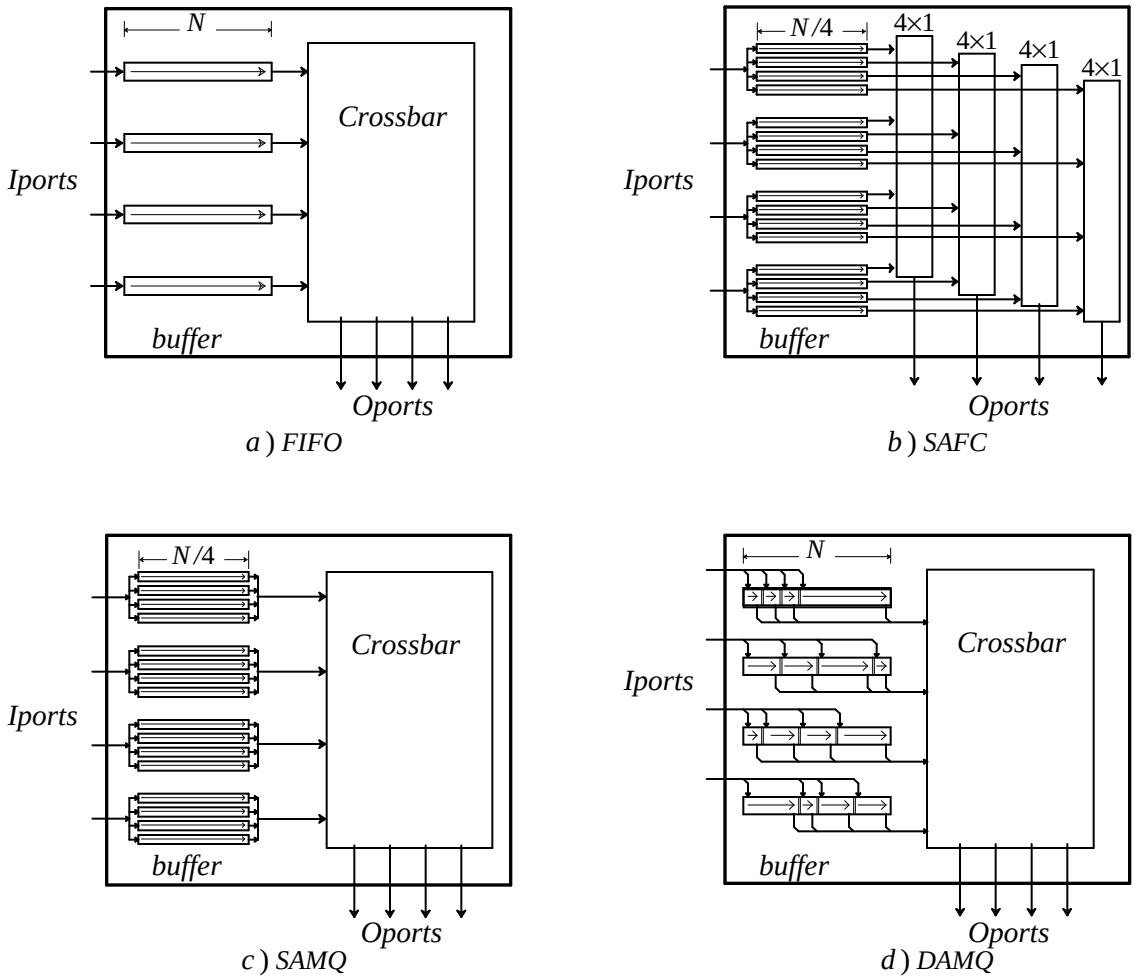

Figure 1: Switches with the Four Buffer Types

buffers at the input ports is that only the packet at the head of the queue can be read from the buffer and transmitted. Thus, if two buffers on a switch have packets at their heads destined for the same output port, one of them will be idle while the other transmits. When such *output port contention* occurs, a packet at the head of a buffer can block all other packets in that buffer from being transmitted, even if the destination output ports of the blocked packets are idle. Since there are the same number of input ports as output ports, whenever two or more packets contend for the same output port, one or more of the output ports is idle until the packets are transmitted. Thus, output port contention can severely degrade the maximum throughput and 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. With such buffers and 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 packet handling: *dynamically allocated multi-queue* (DAMQ) buffers, *statically allocated multi-queue* (SAMQ) buffers, and *statically allocated, fully connected* (SAFC) buffers (see Figure 1) [9].

In both the SAMQ and the 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. 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 simultaneously. The

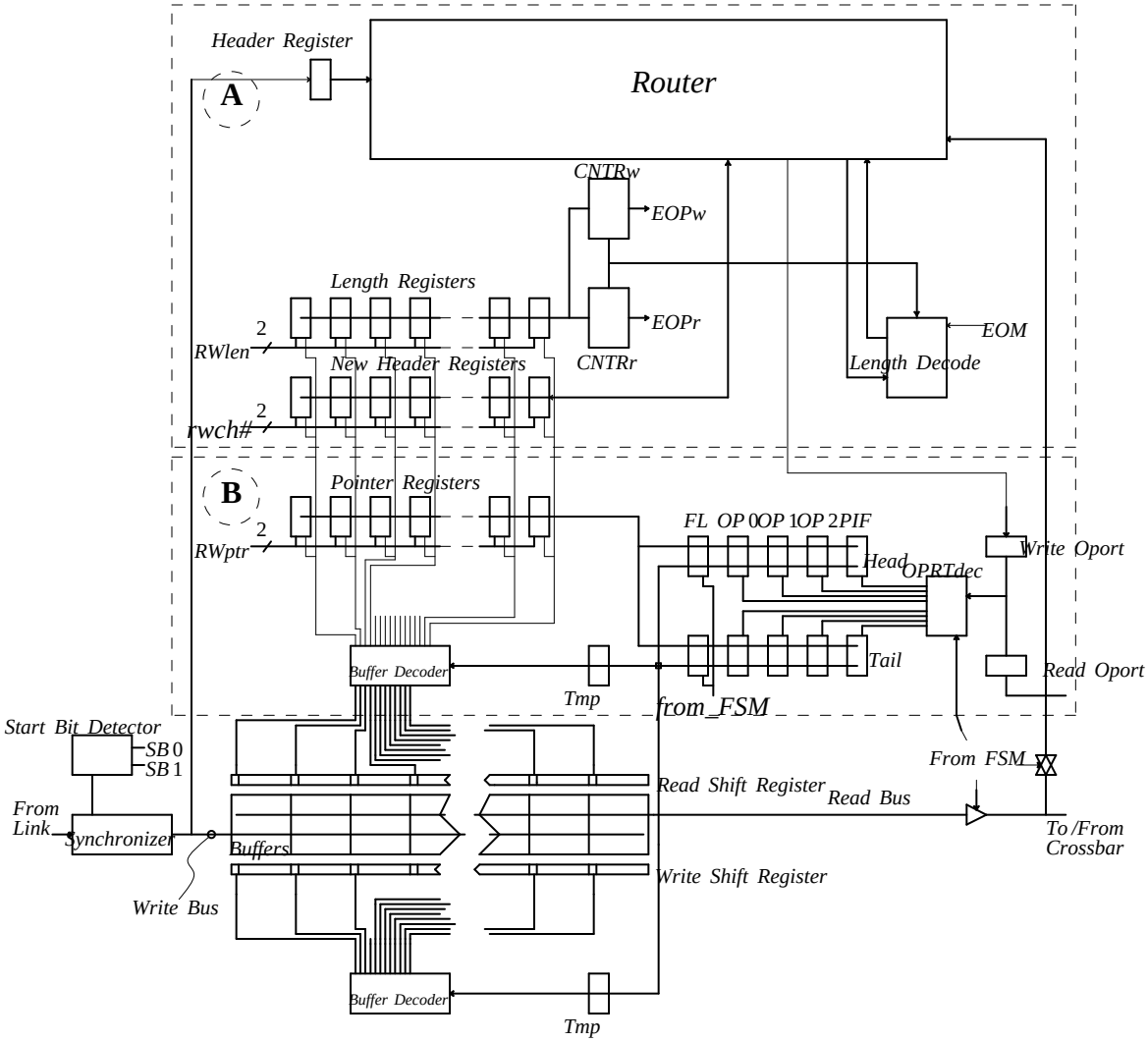

Figure 2: A block diagram of the DAMQ buffer.

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 recently developed [9]. In this buffer the available storage is *dynamically* partitioned between n FIFO queues, one for each output port. One of the main advantages of the DAMQ buffer relative to the SAMQ (and SAFC) buffer is the fact that the dynamic partitioning allows 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.

A detailed discussion of the advantages of the DAMQ buffer over FIFO, SAMQ, and SAFC buffers, based on implementation considerations as well as performance for normal traffic, has been presented elsewhere [9]. A block diagram of the DAMQ buffer, in the context of the ComCoBB chip, is shown in Figure 2. Multiple queues of packets are maintained within a DAMQ buffer in linked lists. In order to manage linked lists of variable size packets, the buffer is partitioned into eight-byte blocks. 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). For each buffer block there is a pointer register, which points to the next block in the list (*Pointer Registers*, in Figure 2). The pointers for the linked list are stored in a separate storage array so that they can be accessed simultaneously with accessing the ‘‘data’’ in the buffer. For an $n \times n$ switch, there are $n+1$ linked lists in each buffer: a list of packets destined for each output port and a list of free (currently unused) buffer blocks.

When a packet arrives at an input port, a block is removed from the free list and used to store the first eight bytes of the packet. The block is then linked to the rear of the list for the output port to which the packet is routed. When a packet is transmitted through the crossbar switch from the input buffer to an output port, the blocks it occupies are returned to the free list so that they can be used again. To manage the linked lists, each buffer has $n+1$ *head* and *tail* registers, as is shown in Figure 2. The head register points to the first block of the first packet of its linked list, and the tail register points to the last block of the last packet in the list. All the hardware required to implement dynamic buffer allocation and multiple queues as described above, except for the buffer storage array and the control finite state machine, is contained within the box marked 'B' in Figure 2. The functional blocks within 'A' are necessary for low latency handling of the ComCoBB's message transport mechanism (virtual circuits) and variable length packets. The hardware in 'A' is independent of the DAMQ buffer, and would accompany any other buffer configuration.

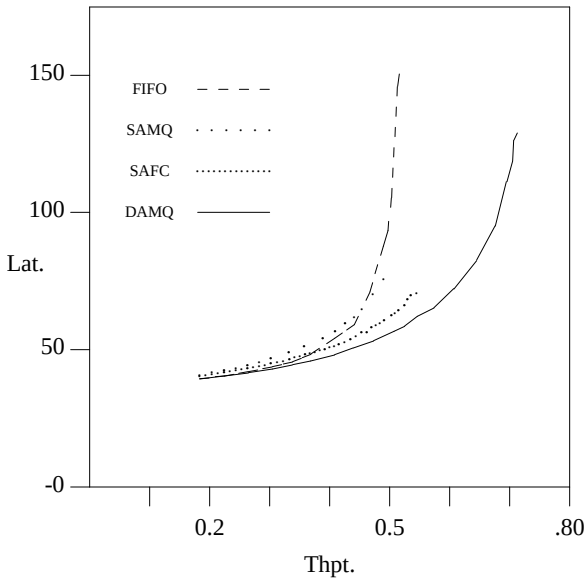

Figure 3: Average latency vs. throughput. Blocking switches. Four packet slots per buffer. Uniform normal traffic.

The performance of the four buffer types (Figure 1) has been evaluated using an event-driven simulation of a 64×64 Omega network [5] composed of three stages of 4×4 switches [9]. The network transmits the packets synchronously, with a delay of twelve time units through each stage, so the minimum network delay is thirty-six time unis. The delay of each packet is measured from the time at which its *sender* creates it, as opposed to the time at which it enters the network. The sender creates a packet, waits until the network accepts the packet (which may be immediately), and then waits a time interval which varies exponentially before generating the next packet. Some of the advantages of the DAMQ buffer are demonstrated by the results shown in Figure 3. This graph shows the average latency versus throughput of networks composed of the four buffer types with four packet slots per buffer and uniform traffic (i.e. each packet's destination is selected from a uniform distribution of all of the possible destinations) [9]. All four networks were simulated with a wide range of throughputs: from lightly loaded (< 20%) to saturation — the maximum throughput achievable with each network (note that the average latency in the SAMQ and SAFC networks could not be ''pushed'' beyond the values shown in the figure). The network composed of DAMQ switches was able to handle significantly higher throughput at lower average latency compared to networks composed of the three other types of switches.

B. The Need for Special Support for High-Priority

The ''performance'' of a network is often 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 be guaranteed faster ''service'' than normal traffic. Thus, in many systems a third important characteristic of the communication network is the *maximum* latency through the network. In our investigation of support for high-priority traffic we have focused on a 64×64 Omega network composed of three stages of 4×4 switches. However, the need for supporting high-priority traffic exists in many other types of systems (such as multicomputers) and we expect that the solutions we present are applicable to a wide variety of systems which use small $n \times n$ switches as the building blocks of their communication network.

The problem of very high worst-case latency through a network is demonstrated by the results of simulations of an Omega network composed of switches with FIFO input port buffers. Each buffer can hold a maximum of four packets. As shown in Figure 4, the maximum latency through the network may be many times larger than the average latency. In order to consider a measure of ''worst case'' performance that is less susceptible to statistical anomalies of the simulation, we have also measured the ''99^{th} percentile latency'' which is the minimum of the latencies of the 1% of the packets that received the poorest ''service'' (longest

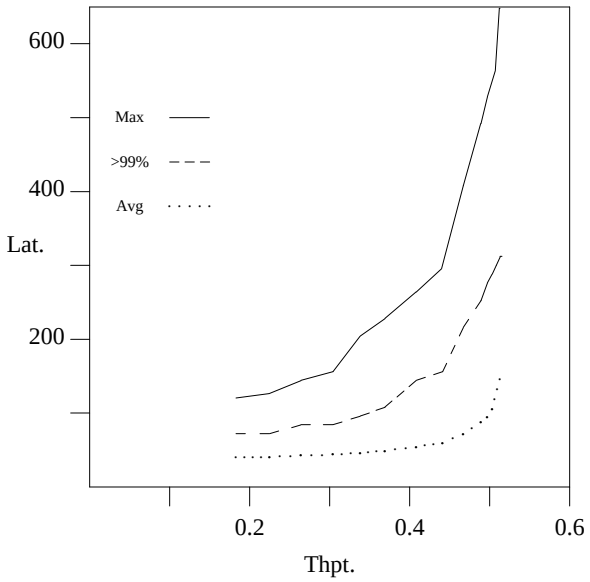

Figure 4: Maximum, 99th percentile, and average packet latency vs. network throughput. FIFO buffers. Four packet slots per buffer.

latencies) from the network. As shown in Figure 4, this 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).

	Buffer Type	Throughput							
		18%		30%		41%		50%	
>99% max	FIFO	72	120	84	156	144	288	288	588
		39.49		43.83		54.13		93.48	
	DAMQ	60	96	84	144	96	192	132	420
		39.31		42.96		48.14		56.70	
	SAMQ	72	132	108	264	144	264	216	420
		40.82		46.81		56.71		77.45	
	SAFC	72	144	96	180	120	240	168	336
		40.47		44.88		51.86		62.40	
avg									

Table 1: The 99th percentile, maximum, and average latencies vs. throughput. Four slots per buffer. Normal uniform traffic.

As system load (throughput) increases, the average packet latency increases. This effect is much more pronounced with respect to maximum (and 99th percentile) packet latencies (see Figure 4). While network performance in terms of average packet latency begins to significantly deteriorate only when the throughput is increased beyond 45%, performance as measured by maximum (or 99th percentile) latency begins to deteriorate with just 25% or 30% throughput.

We have previously shown [9] that networks composed of DAMQ, SAMQ, or SAFC switches perform significantly better than networks composed of FIFO switches (see Figure 3). As shown in Table 1, although all three of these switch types perform better than FIFO switches in terms of worst-case packet latency under heavy load (throughput), the maximum (and 99th percentile) latencies for all three of these buffer types is quite poor. Hence, using the original buffer types without special support for high-priority traffic does not solve the problem.

III. Hardware Support for High-Priority Traffic

As discussed above, the provision of non-FIFO packet handling 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. In our simulations a small percentage (1%, 5%, or 10%) of the packets are ''randomly'' marked as ''high priority'' packets.

	Buffer Type	Throughput			
		18%	30%	41%	50%
>99% avg	FIFO	48	72	108	288
		36.58	38.15	43.99	88.59
	DAMQ	48	48	60	84
		36.24	36.57	37.76	39.62
	SAMQ	60	72	108	180
		37.29	38.98	43.61	56.08
	SAFC	60	72	84	132
		37.32	38.79	41.66	48.05

Table 2: The 99th percentile and average latencies of the high-priority traffic vs. total network throughput. Four slots per buffer. 5% high-priority packets. The only support for high-priority traffic is in the crossbar arbiter (there are *no* separate queues for high-priority packets).

One approach to providing support for high-priority packets is to modify the crossbar arbiter — the controller that determines which input buffer is connected to which output port. The arbiter can be modified so that when determining the crossbar configuration for each cycle it gives priority to those

queues where the head of the queue contains a high-priority packet. Specifically, at each cycle, queues with normal packets have a chance to transmit a packet only after an attempt is made to route all high-priority packets at the heads of all the queues.

Table 2 shows the 99th percentile and average latencies of high-priority packets versus total network throughput for a system in which 5% of all of the packets generated are high-priority. The results in Table 2 indicate that, especially for networks with non-FIFO buffers, these modifications significantly improve high-priority performance over a system with no special support (see Table 1). However, the 99th percentile latencies are still much worse than the average latencies. This is a consequence of the fact that the high-priority packets spend time in queues behind low-priority packets and receive preferential treatment for only the brief time they spend at the heads of queues.

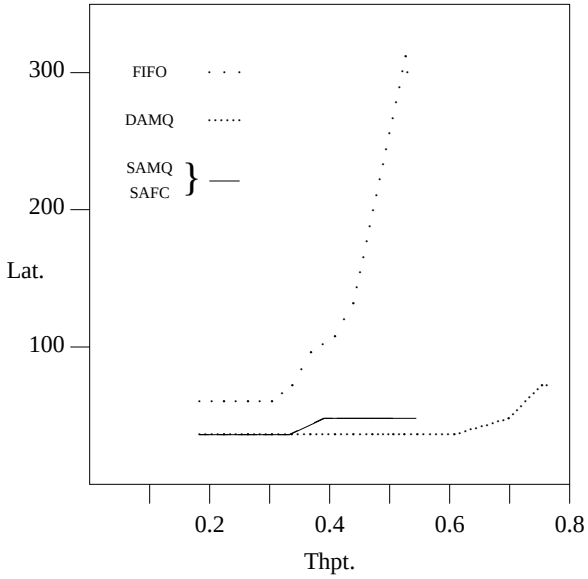

Figure 5: The 99th percentile latencies of high-priority packets vs. throughput. Switches with five slots per input port. 1% of the packets high-priority. Priority queue implementation.

A. Separate High-Priority Queues

The DAMQ, SAMQ and SAFC switches all achieve low latency 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. In our simulation studies we compared the performance of networks of DAMQ, SAMQ, and SAFC switches with dedicated high-priority queues to the performance of networks with FIFO switches that use the priority arbitration scheme discussed above.

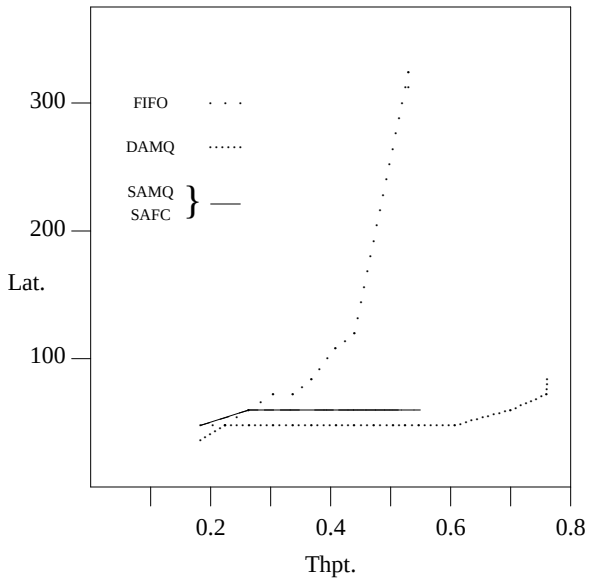

Figure 6: The 99th percentile latencies of high-priority packets vs. throughput. Switches with five slots per input port. 5% of the packets high-priority. Priority queue implementation.

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 has 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 may be needed to manage an additional linked list. For a DAMQ buffer as used in the ComCoBB chip, two additional bits per buffer block (a total of sixteen bits for a buffer of eight eight-byte blocks) may be needed to store the destination output port number for each packet in the high-priority queue. In order to present a fair comparison of the performance of networks with the four buffer types, many of the results presented in this paper were obtained for five slot buffers for all four types.

The 99th percentile latencies of the four buffer

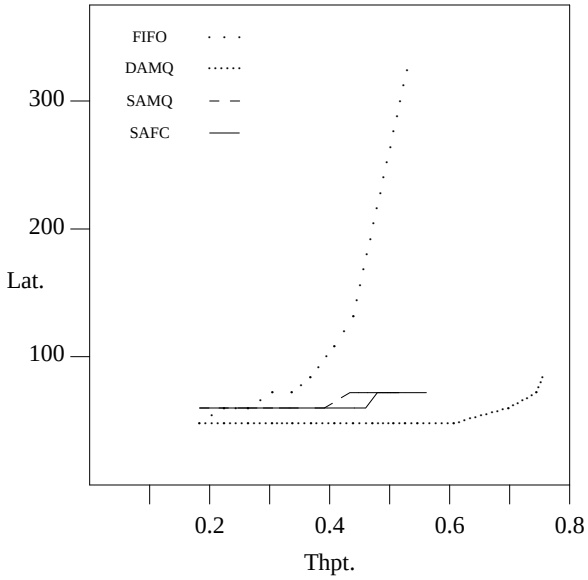

Figure 7: The 99th percentile latencies of high-priority packets vs. throughput. Switches with five slots per input port. 10% of the packets high-priority. Priority queue implementation.

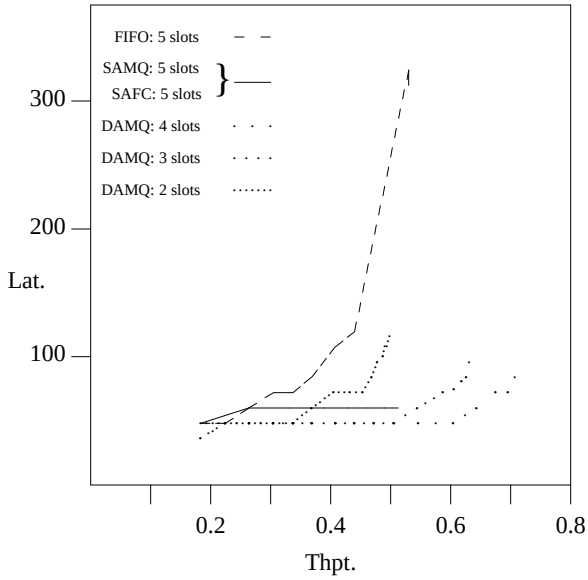

Figure 8: The 99th percentile high-priority latency vs. throughput. FIFO buffer with five packet slots per input port, DAMQ buffers with two, three and four packet slots per input port, SAMQ and SAFC buffers with five packet slots per input port. 5% high-priority traffic. Priority queue implementation.

types with 1%, 5% and 10% high-priority traffic are shown in Figures 5, 6, and 7, respectively. As we have previously discovered with respect to average latencies [9], switches with DAMQ buffers outperform the competition. It appears that the DAMQ buffer 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 are available to any single packet arriving at an input port (one fifth, in this implementation), the entire buffer is available to packets arriving at the DAMQ switch. Thus, the DAMQ switch utilizes the available space better, blocks incoming packets less often, is capable of handling higher throughputs, and delivers packets with lower latencies.

The advantages of the DAMQ buffer are further demonstrated by the results shown in Figure 8. In this figure the 99th percentile latencies of DAMQ switches with two, three, and four slots are compared with those 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.

Figure 9 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. These measurements show that with the DAMQ buffer with a high-priority queue the 99th percentile latency for high-priority traffic is stable for a wide range of throughputs, and actually becomes less than the average latency for normal traffic at medium and high throughputs. An interconnection network with switches using DAMQ buffers with a high-priority queue can thus provide very low worst-case latencies for high priority packets even under network loads approaching saturation.

B. Multiple Buffers

A design option that might be considered for supporting high-priority traffic is to use two independent buffers at each input port — one for normal packets and one for high-priority packets. Possible examples of such a scheme would be 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 would be to completely isolate the high-priority buffering from the normal packet buffering to prevent a situation where a high-priority packet cannot be forwarded along its path due to the buffer in the next stage being full with normal packets.

We considered using input ports with DAMQ

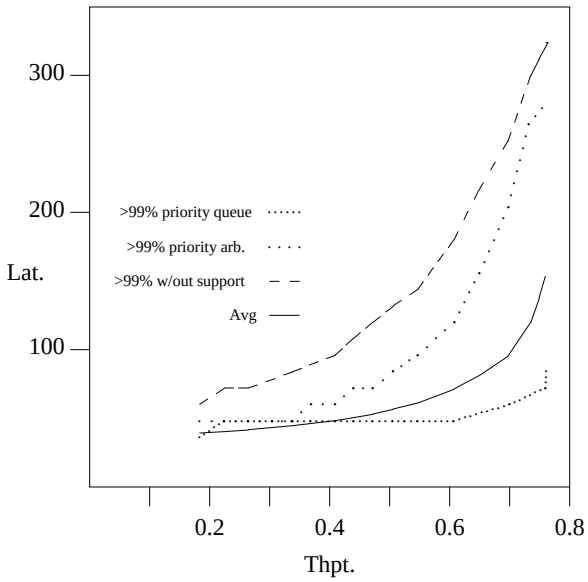

Figure 9: The 99th percentile high-priority latency with priority queues, the 99th percentile high-priority latency with priority arbitration, the 99th percentile high-priority latency with no high-priority support, and the average latency vs. throughput. DAMQ buffers. Five slots per input port. 5% high-priority traffic.

buffers for normal packets and a single-slot FIFO buffer for high-priority packets. In such a scheme the high-priority traffic will behave as though it were in a completely separate network. As discussed above, the high-priority traffic will not have to contend with the low-priority traffic for buffer space. Furthermore, the high-priority traffic does not compete for link bandwidth with the low-priority traffic since whenever there is a conflict, it is always given priority over the low-priority traffic. Thus, in order to evaluate this mechanism we can examine the results of running 100% normal traffic through a FIFO network with buffers of length one, and compare the 99th percentile latency to that of the high-priority packets in a DAMQ network.

	Throughput				
	0.05	0.10	0.14	0.18	0.24
>99% lat.	60	84	108	132	264

Table 3: 99th percentile latency vs. throughput. FIFO network with one slot per buffer. 100% normal traffic.

Table 3 shows the 99th percentile latency of a FIFO network with buffers of length one and throughput from 5% to 24% (saturation). Table 4 shows the 99th percentile latency of the high-priority traffic of a DAMQ network with buffers of length four, total throughput of

% Packets High Priority	1	10	20	30	40	50
>99% Lat.	36	48	60	60	72	84

Table 4: High-priority traffic 99th percentile latency versus the percentage of traffic which is high-priority. This was done on a DAMQ network with four slots per buffer whose throughput was at 50%.

50%, and a percentage of high-priority packets that varies from 1% to 50%. Note that with 10% of the total traffic being high-priority packets, the absolute high-priority throughput is 5%. As can be seen from the tables, the DAMQ buffer alone delivers high-priority packets with a lower 99th percentile latency than if it were paired with a high-priority single-slot FIFO buffer. These results, and similar related results that we have obtained, indicate that a single DAMQ buffer at each input port performs better than the combination DAMQ and a FIFO buffer.

Another possible design would be to partition the input buffer intro two DAMQ buffers: one for normal packets and one for high-priority packets. Instead of using a single four-slot DAMQ buffer at each input port, we consider using two two-slot DAMQ buffers. As shown in Table 5, one still derives considerable benefit from using DAMQ buffers instead of FIFO when there are only two buffer slots. Once again, the high-priority traffic behaves as though the low-priority traffic were not there. Thus we can evaluate the performance of this implementation by examining a DAMQ network with two buffer slots per input port transmitting normal traffic.

Thpt.	>99% Latency	
	FIFO	DAMQ
0.18	72	60
0.26	84	72
0.34	120	96
0.43	204	144
0.48	Sat.	192

Table 5: 99th percentile latency versus throughput for the FIFO and DAMQ switches with two slots per buffer, 100% normal traffic.

We compared the 99th percentile latency in a network of switches with two-slot DAMQ buffers (Table 5) to the 99th percentile latency of the high-priority packets in a network of switches with four-slot DAMQ buffers supporting 50% total traffic with varying percentages of high-priority packets (Table 4). This comparison shows that the configuration with separate DAMQ buffers yields slightly better performance than the single large DAMQ buffer configuration. However,

two two-slot DAMQ buffers require twice the chip area for control compared to the single four-slot DAMQ buffer configuration. The two buffers configuration also requires that twice the amount of flow-control information be passed between switches. In addition, the two buffers scheme is 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 (note that the DAMQ buffer with two buffer slots saturates at a throughput of less than 50%, while the switches with four slots saturate at over 70%). Once again, we conclude that a single DAMQ buffer with a dedicated high-priority queue is the best choice.

IV. Summary and Conclusions

The potential of large multiprocessors and multicomputers to be used for real-time applications depends on the ability to guarantee, with a high degree of confidence, that high-priority traffic can be transmitted through the network with very low, specified, latency. Interconnection networks used to connect the nodes in multiprocessors and multicomputers have the property that 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.

We have developed a method for supporting high-priority traffic in the large set of interconnection networks composed of small $n \times n$ switches. Our scheme is based on a small modification of the *dynamically allocated multi-queue* (DAMQ) buffer that we have recently developed [9]. This buffer provides efficient handling of variable-length packets and forwarding of packets in non-FIFO order. The DAMQ buffer is amenable to efficient VLSI implementation and provides higher saturation throughput and lower average latencies than several common buffer organizations. Our modifications to the DAMQ buffer 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 explored several alternative approaches to providing support for high-priority packets in $n \times n$ switches. We have evaluated the solution based on the modified DAMQ buffer by comparing its performance with that of three alternative buffers and several possible configurations in the context of a multistage interconnection network transmitting packets with two levels of priority. Our simulations show that the DAMQ buffer results in significantly lower latencies and higher maximal throughput than other designs with the same total buffer storage capacity. In addition, the DAMQ maintains a relatively constant 99^{th} percentile high-priority packet latency for a given percentage of high-priority packets, even when the total network throughput is approaching saturation. In our simulations we have not considered variable length packets for which the DAMQ buffer is specifically designed. We believe that the DAMQ buffer will outperform its competition by an even wider margin for the more realistic case of variable length packets which arrive at the inputs of the switch asynchronously.

Acknowledgements

Discussions with T. Lang throughout this project have been extremely helpful. The SIMON simulator was provided by R. Fujimoto. Our simulation studies using SIMON were possible due to the work of T. Frazier, M. Huguet, and L. Kurisaki.

References

1. W. Crowther, J. Goodhue, R. Gurwitz, R. Rettberg, and R. Thomas, ‘‘The Butterfly Parallel Processor,’’ *IEEE Computer Architecture Newsletter*, pp. 18-45 (September/December 1985).
2. W. J. Dally and C. L. Seitz, ‘‘The Torus Routing Chip,’’ *Distributed Computing* **1**(4), pp. 187-196 (October 1986).
3. A. Gottlieb, R. Grishman, C. P. Kruskal, K. P. McAuliffe, L. Rudolph, and M. Snir, ‘‘The NYU Ultracomputer - Designing an MIMD Shared Memory Parallel Computer,’’ *IEEE Transactions on Computers* **C-32**(2), pp. 175-189 (February 1983).
4. P. Kermani and L. Kleinrock, ‘‘Virtual Cut Through: A New Computer Communication Switching Technique,’’ *Computer Networks* **3**(4), pp. 267-286 (September 1979).
5. D. H. Lawrie, ‘‘Access and Alignment of Data in an Array Processor,’’ *IEEE Transactions on Computers* **C-24**(12), pp. 1145-1155 (December 1975).
6. Y. Rimoni, I. Zisman, R. Ginosar, and U. Weiser, ‘‘Communication Element for the Versatile MultiComputer,’’ *15th IEEE Conference in Israel* (April 1987).
7. C. L. Seitz, ‘‘The Cosmic Cube,’’ *Communications of the ACM* **28**(1), pp. 22-33 (January 1985).
8. K. S. Stevens, S. V. Robinson, and A. L. Davis, ‘‘The Post Office - Communication Support for Distributed Ensemble Architectures,’’ *The 6th International Conference on Distributed Computing Systems*, Cambridge, MA, pp. 160-166 (May 1986).
9. Yuval Tamir and Gregory L. Frazier, ‘‘High-Performance Multi-Queue Buffers for VLSI Communication Switches,’’ *15th Annual International Symposium on Computer Architecture*, Honolulu, Hawaii, pp. 343-354 (May 1988).
10. C. Whitby-Strevens, ‘‘The Transputer,’’ *12th Annual Symposium on Computer Architecture*, Boston, MA, pp. 292-300 (June 1985).