Virtual-Channel Flow Control[1]

William J. Dally
Artificial Intelligence Laboratory and
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract

Network throughput can be increased by dividing the buffer storage associated with each network channel into several virtual channels [DalSei]. Each physical channel is associated with several small queues, virtual channels, rather than a single deep queue. The virtual channels associated with one physical channel are allocated independently but compete with each other for physical bandwidth. Virtual channels decouple buffer resources from transmission resources. This decoupling allows active messages to pass blocked messages using network bandwidth that would otherwise be left idle. Simulation studies show that, given a fixed amount of buffer storage per link, virtual-channel flow control increases throughput by a factor of 3.5, approaching the capacity of the network.

1 Introduction

Interconnection Networks

The processing nodes of a concurrent computer exchange data and synchronize with one another by passing messages over an interconnection network [AthSei, BBN86, Dally89, Seitz85]. The interconnection network is often the critical component of a large parallel computer because performance is very sensitive to network latency and throughput and because the network accounts for a large fraction of the cost and power dissipation of the machine.

An interconnection network is characterized by its topology, routing, and flow control [Dally89b]. The topology of a network is the arrangement of nodes and channels into a graph. Routing specifies how a packet chooses a path in this graph. Flow control deals with the allocation of channel and buffer resources to a packet as it traverses this path. This paper deals only with flow control. It describes a method for allocating resources to packets using virtual channels [DalSei]. This method can be applied to any topology and routing strategy.

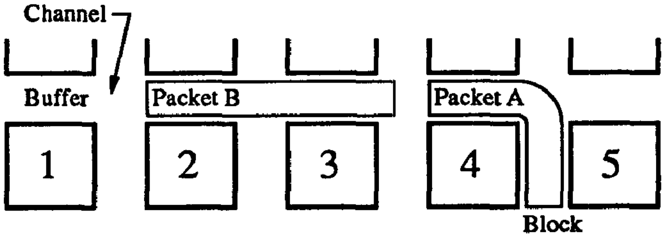


Figure 1: Packet B is blocked behind packet A while all physical channels remain idle.

The Problem

The throughput of interconnection networks is limited to a fraction (typically 20%-50%) of the network's capacity [Dally87] because of coupled resource allocation.

Interconnection networks are composed of two types of resources: buffers and channels. Typically, a single buffer is associated with each channel. Once a packet, A, is allocated the buffer, no other packet, B, can use the associated channel until A releases the buffer. If packet A becomes blocked while holding the buffer, the channel is idled even though there may be other packets in the network that can make productive use of the channel.

This situation is illustrated in Figure 1. In the figure, the network is depicted as a network of streets where each block corresponds to a buffer and each intersection represents one or more channels that connect buffers. Packet A holds buffers 4N (north of block 4) and 4E and is blocked. Packet B is unable to make progress even though all physical channels it requires, (3N to 4N) and (4N to 5N), are idle because Packet A holds

[1] The research described in this paper was supported in part by the Defense Advanced Research Projects Agency under contracts N00014-80-C-0622 and N00014-85-K-0124 and in part by a National Science Foundation Presidential Young Investigator Award with matching funds from General Electric Corporation and IBM Corporation.

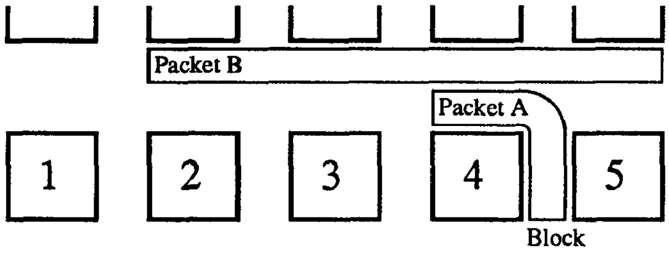


Figure 2: Virtual channels provide additional buffers (lanes) allowing Packet B to pass blocked Packet A.

buffer 4N which is coupled to channel (3N to 4N).

Virtual Channel Flow Control

Virtual channels decouple resource allocation by providing multiple buffers for each channel in the network. If a blocked packet, A, holds a buffer associated with a channel, another buffer is available allowing other packets to pass A. Figure 2 illustrates the addition of virtual channels to the network of Figure 1. Packet A remains blocked holding buffers 4N.1 and 4E.1. In Figure 2, however, Packet B is able to make progress because alternate buffer 4N.2 is available allowing it access to channel (3N to 4N).

Adding virtual channels to an interconnection network is analogous to adding lanes to a street network. A network without virtual channels is composed of one-lane streets. In such a network, a single blocked packet blocks all following packets. Adding virtual channels to the network adds lanes to the streets allowing blocked packets to be passed.

In addition to increasing throughput, virtual channels provide an additional degree of freedom in allocating resources to packets in the network. This flexibility permits the use of scheduling strategies, such as routing the oldest packet first, that reduce the variance of network latency.

The most costly resource in an interconnection network is physical channel (wire) bandwidth. The second most costly resource is buffer memory. Adding virtual channel flow control to a network makes more effective use of both of these resources by decoupling their allocation. The only expense is a small amount of additional control logic.

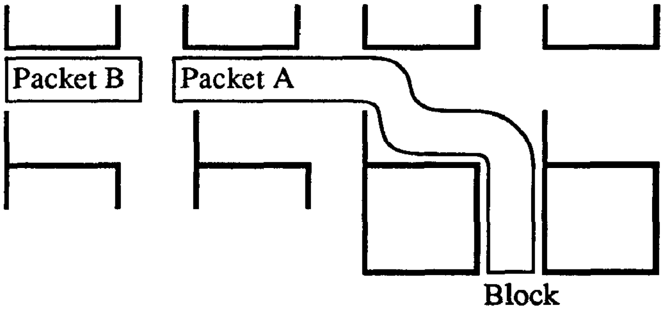


Figure 3: Output queueing or partitioned input queues provide one stage of decoupling. However, long packets (such as Packet A) continue to couple resources and cannot be passed.

Background

The use of virtual channels for flow control builds on previous work in using virtual channels for deadlock avoidance and in using output queueing or split input queues for partial resource decoupling. Virtual channels were introduced in [DalSei] for purposes of deadlock avoidance. A cyclic network can be made deadlock-free by restricting routing so there are no cycles in the channel dependency graph and then adding virtual channels to reconnect the network.

A single stage of resource decoupling is provided by output queueing [KHM87]. By performing the queueing in the output of a switch rather than the input, arriving packets are able to pass blocked messages arriving on the same input. Tamir [Tamir88] has shown how to achieve the same single-stage resource decoupling by partitioning the switch's input queue. This single stage resource decoupling is effective only if an entire packet fits in a single node. As shown in Figure 3 , when a long packet is blocked, it backs up into the output stage of the previous node preventing any following packet from passing it. Extending our roadway analogy, output queueing provides a "turning lane".

Summary

The next section introduces the notation and assumptions that will be used throughout this paper. Section 3 describes virtual channel flow control in detail. The results of simulating networks using virtual channel flow control are described in Section 4.

2 Preliminaries

Topology

An interconnection network consists of a set of **nodes**, N, and a set of **channels**, $C \subseteq N$. Each channel is unidirectional and carries data from a source node to a destination node. A bidirectional network is one where $(n1, n2) \in C \Rightarrow (n2, n1) \in C$.

Routing

A packet is assigned a route through the network according to a **routing relation**, $R \subseteq C \times N \times C$, given the channel occupied by the head of the packet and the destination node of the packet, the routing relation specifies a (possibly singleton) set of channels on which the packet can be routed.

Flow Control

Communication between nodes is performed by sending **messages**. A message may be broken into one or more **packets** for transmission. A packet is the smallest unit of information that contains routing and sequencing information. A packet contains one or more flow control digits or **flits**. A flit is the smallest unit on which flow control is performed. Information is transferred over physical channels in physical transfer units or **phits**. A phit is usually the same size or smaller than a flit.

The flow control protocol of a network determines (1) how resources (buffers and channel bandwidth) are allocated and (2) how packet collisions over resources are resolved. A resource collision occurs when a packet, P, is unable to proceed because some resource it needs (usually a buffer) is held by another packet. Collisions may be resolved by (1) blocking P in place, (2) buffering P in a node prior to where the collision occurs, (3) dropping P , or (4) misrouting P to a channel other than the one it requires. The technique described in this paper is applicable to all of these flow control strategies but is most appropriate for networks that use blocking or limited buffering to resolve collisions.

The flow control strategy allocates buffers and channel bandwidth to flits. Because flits have no routing or sequencing information, the allocation must be done in a manner that keeps the flits associated with a particular packet together. This may be done by associating a set of buffers and some control state together into a virtual channel. A virtual channel or lane is allocated to a packet and the buffers of the lane are allocated in a FIFO manner to the flits of that packet.

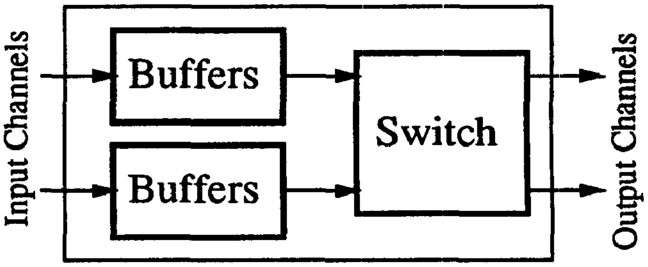


Figure 4: Node organization. Each network node contains a set of buffers for each input channel and a switch.

Most networks associate only a single lane with each channel. This paper describes a method for improving the performance of networks by associating several lanes with each channel. This method makes no assumptions about how wires are allocated.

Wormhole Routing

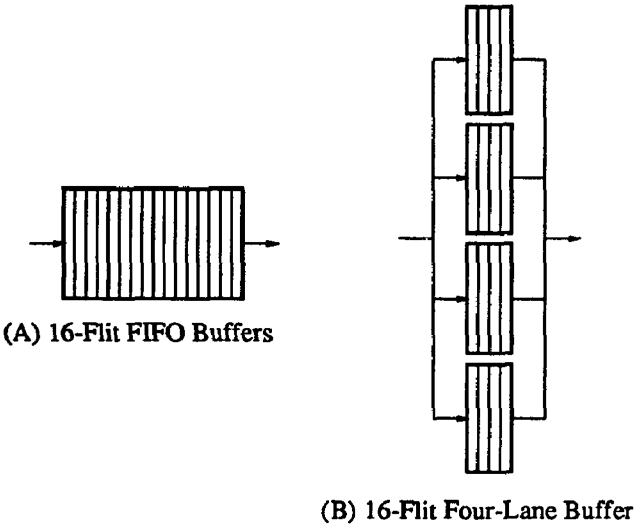
The technique described here is particularly suitable for use in networks that use wormhole routing [Dally87]. Wormhole routing refers to a flow-control protocol that advances each flit of a packet as soon as it arrives at a node (pipelining) and blocks packets in place when required resources are unavailable. Wormhole routing is attractive in that (1) it reduces the latency of message delivery compared to store and forward routing, and (2) it requires only a few flit buffers per node. Wormhole routing differs from virtual cut-through routing [KerKle] in that with wormhole routing it is not necessary for a node to allocate an entire packet buffer before accepting each packet. This distinction reduces the amount of buffering required on each node making it possible to build fast, inexpensive routers.

3 Virtual Channel Flow Control

Structure

Each node of an interconnection network contains a set of buffers and a switch[2] In this paper, we assume that the buffers are partitioned into sets associated with each input channel, an input-buffered node, as shown in Figure 4. An output-buffered switch [KHM87, Tamir88]

[2] Each node also contains driver and receiver circuits to communicate across the physical wires and control logic.



(A) 16-Flit FIFO Buffers

(B) 16-Flit Four-Lane Buffer

Figure 5: (A) Conventional nodes organize their buffers into FIFO queues restricting routing. (B) A network using virtual-channel flow control organizes its buffers into several independent lanes.

can be considered to be an input buffered switch with a non- blocking first stage by associating the buffers on the output of each stage with the inputs of the next stage.

A conventional network organizes the flit buffers associated with each channel into a first-in, first-out (FIFO) queue as shown in Figure 5A. This organization restricts allocation so that each flit buffer can contain only flits from a single packet. If this packet becomes blocked, the physical channel is idled because no other packet is able to acquire the buffer resources needed to access the channel.

A network using virtual channel flow control organizes the flit buffers associated with each channel into several lanes as shown in Figure 5B. The buffers in each lane can be allocated independently of the buffers in any other lane. This added allocation flexibility increases channel utilization and thus throughput. A blocked message, even one that extends through several nodes, holds only a single lane idle and can be passed using any of the remaining lanes.

Operation

In a network using virtual channel flow control, flow control is performed at two levels. At the packet level packets are assigned to virtual channels or lanes. At the flit level channel bandwidth, switch bandwidth, and individual buffers are allocated to flits.

Lane assignment is performed by the node (node A) at the transmitting end of the physical channel. For each of its output channels, this node keeps track of the state of each lane buffer at the opposite end of the channel. For each lane, the state information includes whether the lane is assigned, and if assigned, how many empty buffers it contains. A packet in an input buffer on node A selects a particular output channel based on its destination and the routing algorithm in use. The flow-control logic then assigns this packet to any free lane of the selected channel. If all lanes are in use, the packet is blocked.

Maintaining lane state information on the transmitting end of the channel allows lane assignment to be performed on a single node. No additional internode communication is required to maintain this information as it is already required for flit-level flow control.

Once a lane is assigned to a packet, flit-level flow control is used to advance the packet across the switch and physical channel. To advance from the input buffer on the transmitting node (node A) to the input buffer on the receiving node (node B), a flit must gain access to (1) a path through the switch to reach the output of node A, and (2) the physical channel to reach the input of node B. Typically either the switch is non-blocking, and thus always available, or a few flits of buffering are provided at the output of node A so that switch and channel resources do not have to be allocated simultaneously.

When the last flit of a message (the tail flit) leaves a node the lane assigned to that packet is deallocated and may be reassigned to another packet.

Allocation Policies

Flit-level flow control across the physical channel involves allocating channel bandwidth among lanes that (1) have a flit ready to transmit and (2) have space for this flit at the receiving end. Any arbitration algorithm can be used to allocate this resource including random, round-robin, or priority.

Deadline scheduling [LiuLey] can be implemented by allocating channel bandwidth based on a packet's deadline or age – earliest deadline or oldest age first. Scheduling packets by age reduces the variance of message latency. Deadline scheduling provides several classes of delivery service and reduces the variance within each class.

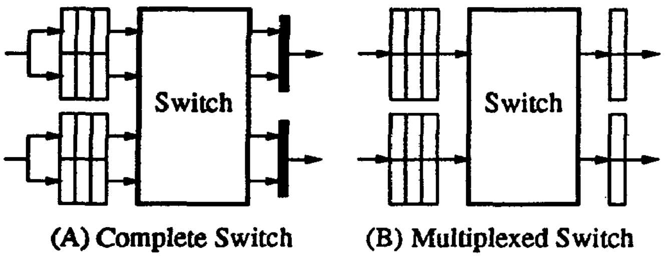


Figure 6: (A) Adding virtual channels increases switch complexity if a complete switch is used. (B) using a multiplexed switch leaves switch complexity unchanged.

Implementation Issues

Virtual channel flow control can be integrated into existing switch designs by replacing FIFO buffers with multi-lane buffers. When this replacement is made, however, the switch must be modified to deal with a larger number of inputs and outputs, and the flow control protocol between nodes must be modified to identify lanes.

Increasing the number of virtual channels multiplexed on each physical channel increases the number of inputs and outputs that must be switched at each node. If the switch handles each of these inputs and outputs separately as shown in Figure 6(A), the switch complexity will increase significantly. Increasing the switch complexity is not required, however. The average data rate out of the set of lanes associated with a given physical channel is limited to the bandwidth of the channel. Thus it is sufficient to provide a single switch input for each physical input and output channel as shown in Figure 6(B). With this organization, a small (one or two flit) output buffer is desirable to decouple switch allocation from physical channel allocation. Individual lanes are multiplexed onto the single path through the switch in the same manner that they are multiplexed over the single physical channel between the nodes.

Any network that uses blocking or buffering flow control must, for each channel, send information in the reverse direction to indicate the availability of buffering on the receiving node. These acknowledgment signals can be transmitted on separate wires [DalSon] or, in a bidirectional network, they can be transmitted out-of-band on a channel in the opposite direction [Inmos].

In a network using multi-lane buffers, two effects increase the acknowledgment traffic. First, a few bits must be added to each acknowledgment signal to identify the lane being acknowledged. Second, because a lane buffer is typically smaller than a channel FIFO, the use of block acknowledgments to amortize the cost of the signal over several flits is restricted[3].

Even with these effects, acknowledgment signal bandwidth is still a small fraction of forward bandwidth. In a network with 32-bit flits, 15 lanes per channel, and no block acknowledgment, 4 bits must be sent along the reverse channel for each flit transmitted along the forward channel, a 12.5% overhead. An additional 12.5% overhead is required to identify the lane associated with each flit sent in the forward direction. Such a scheme could be realized by a physical channel consisting of an 9-bit forward path (8-bit phits) and a 1-bit reverse path. Every four channel cycles a 32-bit flit is transmitted over the forward path along with its 4-bit lane identifier and a 4-bit acknowledgment code is transmitted over the reverse path. Efficiency can be improved further by increasing the flit size or using block acknowledgments.

4 Experimental Results

To measure the effect of virtual channel flow control on network performance (throughput and latency) we have simulated a number of networks holding the total buffer storage per node constant and varying the number of lanes per channel. If lanes are added, the depth of each lane is proportionally reduced.

The simulator is a 3000 line C program that simulates interconnection networks at the flit-level. A flit transfer between two nodes is assumed to take place in one time unit. The network is simulated synchronously moving all flits that have been granted channels in one time step and then advancing time to the next step. The simulator is programmable as to topology, routing algorithm, and traffic pattern.

Throughput is measured by applying to each network input a saturation source that injects a new packet into the network whenever a lane is available on its input channel. Throughput is given as a fraction of network capacity. A uniformly loaded network is operating at capacity if the most heavily loaded channel is used 100% of the time.

Latency is measured by applying a constant rate source to each input and measuring the time from packet creation until the last flit of the packet is accepted at the destination. Source queueing time is included in the

[3] A block acknowledgments signals the availability of a block (several flits) of storage in a single action rather than signalling each flit separately

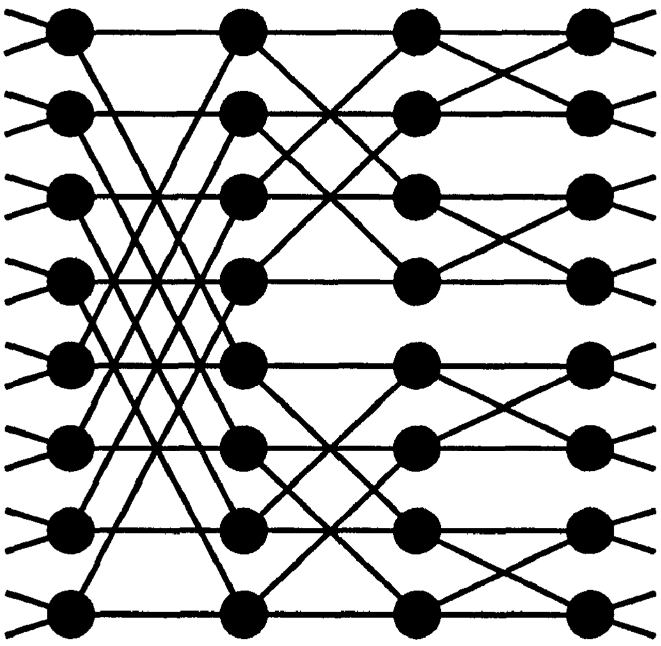


Figure 7: A 2-ary 4-fly network.

latency measurement.

Multistage Networks

Multistage (k-ary n-fly) networks have kn inputs connected to kn outputs by n-stages of kn-1 k x k -switches. For example, a 2-ary 4-fly is shown in Figure 7. Because of their simplicity, simulations of multistage networks were used to evaluate virtual channel flow control. The method is in no way specific to multistage networks. It is equally applicable to other topologies including direct networks such as k-ary n- cubes, and trees.

Throughput

Figure 8 shows the saturation throughput versus the number of lanes per channel for radix-2 multistage networks (2-ary n- flys). Data is shown for networks with dimensions of 4,6,8, and 10. Each network simulated has 16 flits of storage per channel. The number of lanes per channel was varied from 1 (conventional network) to 16 in powers of two.

The simulations were run with packet length fixed at 20 flits and uniformly distributed random packet destinations. Channel bandwidth was allocated randomly to lanes.

The results show that adding lanes to a network greatly increases its throughput, particularly for large networks. The radix 10 (1024-input) network shows a throughput gain of 250%, throughput more than tripled, with the addition of lanes. The first few lanes results in most of the improvement with diminishing returns for larger numbers of lanes. Increasing from 8 to 16 lanes gives only a 14% improvement for the radix 10 network. This suggests that 4 to 8 lanes per channel is adequate for most networks.

Adding lanes holding the total storage constant gives a far greater throughput improvement than does increasing the total amount of buffering with a single lane (see [Mailhot]).

Latency

Figure 9 shows the average packet latency versus offered traffic for radix-2 dimension- 8 multistage networks (2-ary 8-flys). As above, each network simulated has 16 flits of storage per node, and the number of lanes per node was varied from 1 to 16 in powers of two.

The results show that adding lanes has little effect on latency below the saturation throughput of a conventional network. The curves lie on top of each other below a traffic of 0.2. Above this point the latency of a conventional network is infinite (no data for 1 lane) and the addition of lanes extends the latency curve smoothly for each network simulated until it reaches its saturation throughput.

Each network's departure from the smooth latency curve as saturation is reached is smoother than shown in this figure. The abrupt transition shown in the figure is due to the coarse spacing of data points at 10% increments in network load.

Scheduling Algorithm

Figure 10 shows the effect of the channel scheduling algorithm on latency. The figure shows two latency histograms, one for a random assignment of channel bandwidth to packets, and the other for oldest-packet-first channel bandwidth allocation (deadline scheduling). Both curves are for 2-ary 6- fly networks with random traffic operating at 50% capacity. The histograms have been truncated at 128 cycles latency.

The use of deadline scheduling reduced the average latency slightly, from 74.4 cycles to 71.8 cycles and dra-

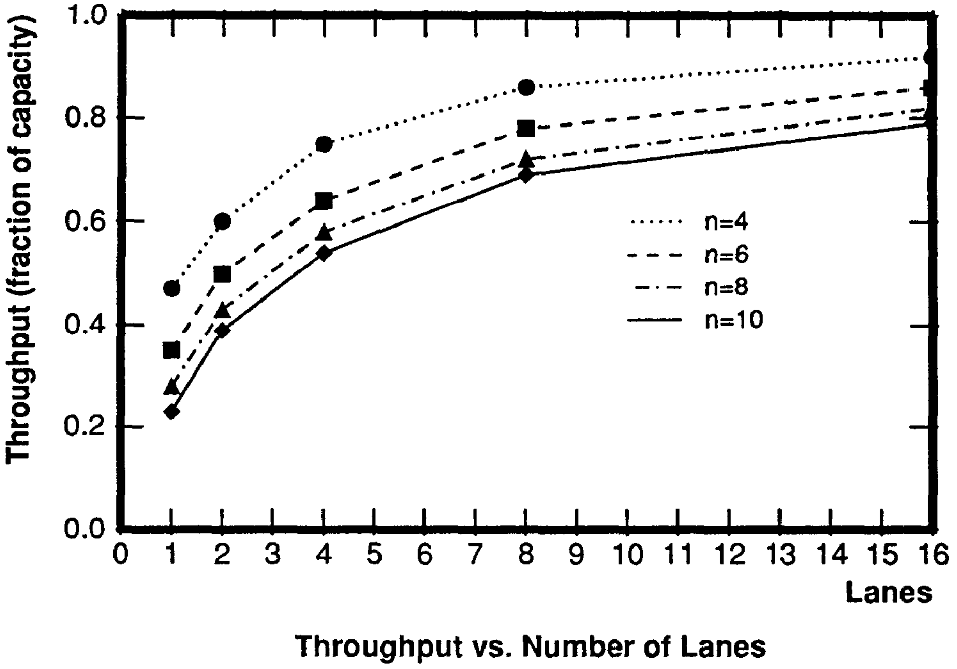


Figure 8: Throughput vs. number of lanes for radix-2 multistage networks under random traffic. Throughput increases rapidly as lanes are added with diminishing returns for larger numbers of lanes. For the dimension-10 network throughput with 16 lanes is 3.5 times the throughput with a single lane.

matically reduced the variation in latency. With deadline scheduling 4619 packets, over one quarter of all packets delivered during the simulation, had a latency of 23 cycles, the minimum possible for this network. The deadline curve also shows smaller peaks of 949, 277, and 93 packets at 43, 63, and 83 cycles that are due to packets that had to wait one or more entire packet delays (about 20 cycles) before being able to proceed. These peaks stand out from the background level that slopes from about 100 packets per cycle at 24 cycles latency to about 30 packets per cycle at 128 cycles latency.

In contrast to the sharp peaks of the deadline curve, the random curve shows broad peaks at 43 and 83 cycles. The random curve also has a higher background level except in the region from 24 cycles to 36 cycles. The two curves cross over at 36 cycles. The data suggest that deadline scheduling can be useful in reducing average message latency and in making message latency more predictable.

The experiment shown here was run with all packets having the same deadline (their birth time). The same scheduling algorithm can be used to provide different classes of service by giving some packets tighter deadlines than others. This would be useful, for example, in a switch that handles both voice and data where the voice traffic has a tight deadline and should be dropped if it cannot make its deadline.

5 Conclusion

The performance of interconnection networks can be improved by organizing the buffers associated with each network channel into several lanes or virtual channels rather than a single FIFO queue. Associating several lanes with each physical channel decouples the allocation of virtual channels to packets from the allocation of physical channel bandwidth to flits. This decoupling allows active messages to pass blocked messages dramatically improving network throughput.

The use of virtual channel flow control also allows flexibility in allocating physical channel bandwidth. By decoupling resource allocation, a channel's bandwidth need not be allocated to the "next packet in line". Instead, this bandwidth may be allocated on the basis of packet type, age, or deadline. The use of deadline scheduling may be particularly important in networks where one class of packets must be delivered in a bounded amount of time.

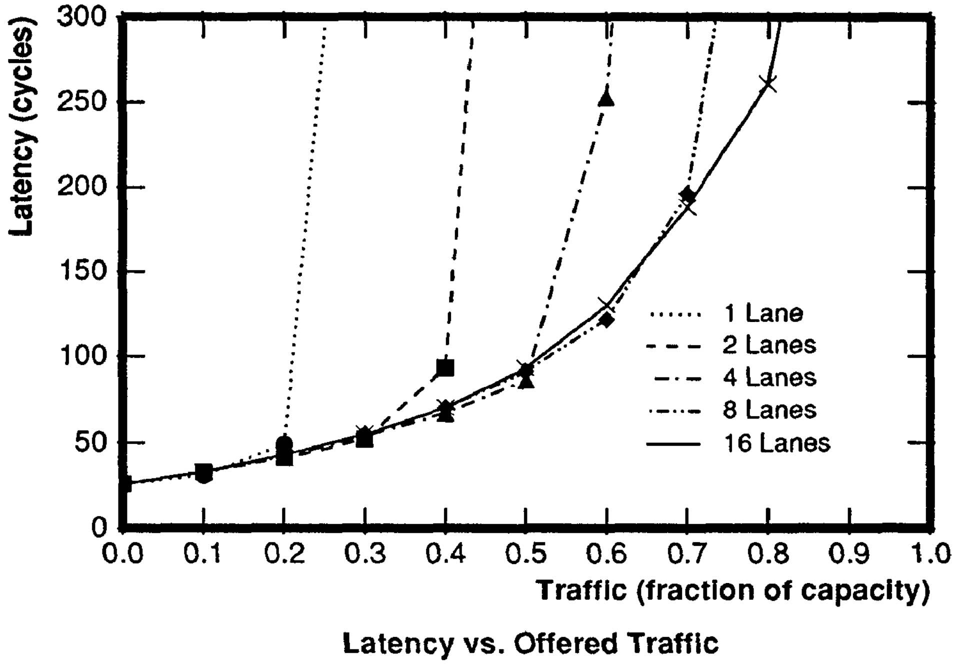


Figure 9: Latency vs. offered traffic for 2-ary 8-flys under random traffic. Adding lanes to the network has little effect on latency below saturation throughput for a single lane. Above this point, the latency curves for multiple lanes are extended smoothly until they reach their saturation throughput levels.

Several indirect (2-ary n-fly) networks have been simulated to measure the performance of virtual channel flow control. These simulations show that with the total amount of buffer storage per node held constant, adding lanes increased throughput by a significant factor. For a 2-ary 10-fly network (1024 input butterfly), the throughput with 16 lanes per channel is 3.5 times the throughput with a single lane.

Simulations also indicate that adding lanes has little effect on latency below saturation throughput and extend the latency curve smoothly as throughput is improved. The use of deadline scheduling reduces average latency by a small amount and makes latency much more predictable.

The critical resources in an interconnection network are wire bandwidth and buffer memory. Virtual channel flow control is a method for allocating these critical resources in a more efficient manner. With network switches constructed using VLSI circuits, the cost of adding the small amount of control state and logic required to implement multiple lanes per channel is well worth the cost.

Acknowledgment

I thank Steve Ward, Anant Agarwal, Tom Knight, and Charles Leiserson for many discussions about interconnection networks and their analysis. Appreciation is due to the referees for many helpful comments and suggestions. Finally, I thank all the members of the MIT Concurrent VLSI Architecture group and especially Scott Wills, Dave Chaiken, and Waldemar Horwat for their help with and contributions to this paper.

References

[AthSei] Athas, W.C., and Seitz, C.L., "Multicomputers: Message-Passing Concurrent Computers," IEEE Computer, Vol 21, No 8, August 1988, pp. 9-24.

[BBN86] BBN Advanced Computers, Inc., Butterfly Parallel Processor Overview, BBN Report No 6148, March 1986.

[Dally87] Dally, W.J. "Wire-Efficient VLSI Multiprocessor Communication Networks," Proceedings of the Stanford Conference on Advanced Research in VLSI, Paul Losleben, ed., MIT Press, March 1987, pp. 391-

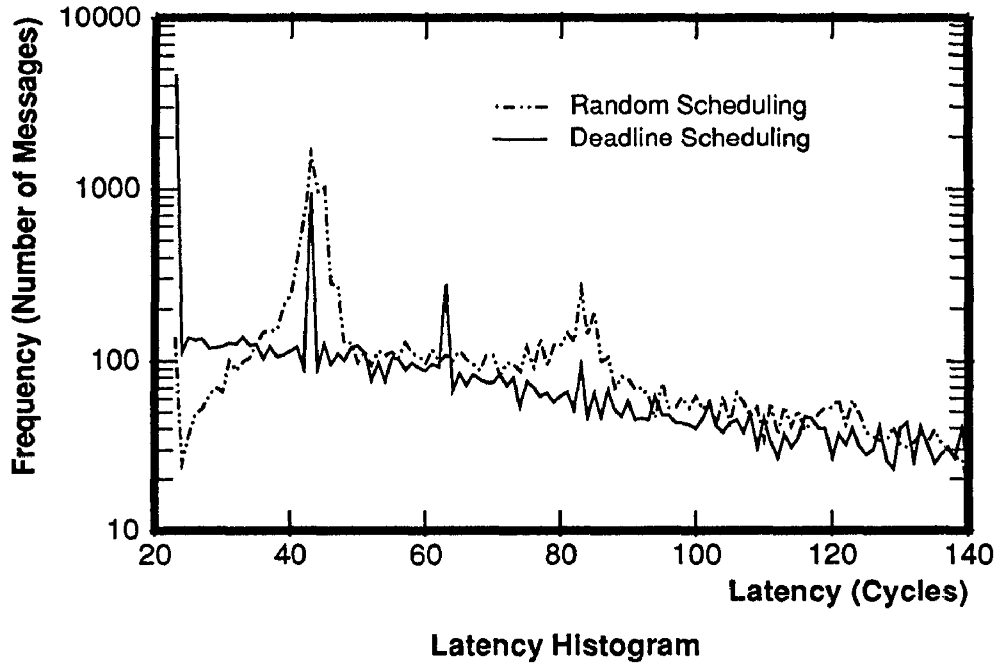


Figure 10: Latency histogram for a 2-ary 6-fly network with random traffic at 50% capacity using deadline (oldest first) and random scheduling of physical channels. The histogram for deadline scheduling has a sharp peak at 23 and smaller peaks at 43, 63, and 83. The curve for random scheduling shows a broad peak at 43.

415.

[Dally89a] Dally, W.J., et. al., "The J- Machine: a Fine-Grain Concurrent Computer," Information Processing 89, Elsevier North Holland, 1989.

[Dally89b] Dally, W.J., "Network and Processor Architecture for Message-Driven Computing," in VLSI and Parallel Processing, R. Suaya and G. Birtwistle eds., Morgan Kaufmann, to appear 1989.

[DalSei] Dally, W.J. and Seitz, C.L., "Deadlock Free Message Routing in Multiprocessor Interconnection Networks," IEEE Transactions on Computers, Vol C-36, No 5, May 1987, pp. 547-553.

[DalSon] Dally, W.J., and Song, P., "Design of a Self-Timed VLSI Multicomputer Communication Controller", Proceedings IEEE International Conference on Computer Design, ICCD-87, October 1987, pp 230-234.

[Inmos] Inmos Limited, IMS T424 Reference Manual, Order No 72 TRN 006 00, Bristol, UK, November 1984.

[KerKle] Kermani, P., and Kleinrock, L., "Virtual Cut-Through: A New Computer Communication Switching Technique," Computer Networks, Vol 3, 1979, pp. 267-286.

[KHM87] Karol, M.J., Hluchyj, M.G., and Morgan, S.P., "Input Versus Output Queueing on a Space-Division Packet Switch," IEEE Transactions on Communications, Vol COM- 35, No 12, December 1987, pp. 1347- 1356.

[LiuLey] Liu and Leyland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," Journal of the ACM, Vol 20, No 1, January 1973, pp. 46- 61.

[Mailhot] Mailhot, J.N., A Comparative Study of Routing and Flow-Control Strategies in k- ary n-cube Networks, Massachusetts Institute of Technology, SB Thesis, May 1988.

[Seitz85] Seitz, C.L., "The Cosmic Cube," Communications of the ACM, Vol 28, No 1, January 1985, pp. 22-33.

[Tamir88] Tamir, Y., and Frazier, G.L., "High-Performance Multi-Queue Buffers for VLSI Communication Switches," 15th annual ACM/IEEE Symposium on Computer Architecture, June 1988, pp. 343-354.