JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING **52**, 82–95 (1998)
ARTICLE NO. PC981466

# Adaptive Virtual Cut-Through as a Viable Routing Method[1]

Ho Won Kim,[†] Hyun Suk Lee,[†] Sunggu Lee,[†]and Jong Kim[‡]

[†]*Department of Electrical Engineering and* [‡]*Department of Computer Science and Engineering, Pohang University of Science and Technology* (*POSTECH*), *San 31 Hyoja Dong*, *Pohang 790-784*, *South Korea*

Received February 14, 1997; revised May 11, 1998; accepted May 11, 1998

Adaptive *virtual cut-through* is considered as a viable alternative to *wormhole switching* for fast and hardware-efficient interprocessor communication in multicomputers. Computer simulations are used to show that our implementation of a minimal-path fully-adaptive virtual cut-through algorithm outperforms both deterministic and adaptive wormhole switching methods under both uniform random message distributions and clustered distributions such as the matrix transpose. A hardware-efficient implementation of adaptive virtual cut-through has been implemented using a semi-custom-designed router chip that requires only 2.3% more area than a comparable deterministic wormhole router chip. A network interface controller chip, which is crucial to our adaptive virtual cut-through method, has also been designed and is under fabrication. © 1998 Academic Press, Inc.

*Key Words:* Virtual cut-through, wormhole switching, circuit switching, multicomputer, direct interconnection network, VLSI design.

## 1. INTRODUCTION

The interconnection network is a critical component of a massively parallel computer system because information must be communicated quickly in order for the processing nodes of the parallel computer to cooperate in solving a given problem. Information may be sent as messages between the processing nodes of a distributed-memory multicomputer or as data between the processors and memories of a shared-memory multiprocessor

For fast inter-processor communications, dedicated hardware devices, termed *routers*, can be used to perform flow control without the intervention of the computation processor. Two popular hardware-supported switching methods are

[1] This paper has been significantly modified from a preliminary version of this work presented at the "24th International Conference on Parallel Processing" held in August 1995. This work was supported in part by the Korea Science and Engineering Foundation (KOSEF) under Grant 96-0101-09-01-3.

*wormhole* (*WH*) *switching* [1] and *virtual cut-through* (*VCT*) *switching* [1]. In WH switching (also known as WH routing) and VCT switching, messages arriving at an intermediate node are immediately forwarded to the next node on the path without buffering, provided that a channel to the next node is available. This is referred to as a *cut-through* operation. A cut-through may be performed at an intermediate node as soon as the header of the message arrives with the destination information. If cut-throughs are established through all intermediate nodes, a *circuit* is established to the destination, and the switching method resembles traditional *circuit switching*. If the message header is *blocked* at an intermediate node because the requested outgoing channel is unavailable, the message is kept in the network (i.e., in the on-line flit buffers at each node along the path up to the current node) in WH and completely buffered at the current node in VCT.

Wormhole switching is the most popular communication method used in current commercial multicomputers due to its low hardware overhead (requiring only a single flit buffer per incoming channel) and high performance. However, VCT can typically achieve higher performance than WH for the following reason: when a packet cannot cut-through an intermediate node, the packet is buffered at that node instead of being kept in the network as in WH. This effect becomes particularly evident with heavy network traffic. The main reason that VCT has not been as popular as WH is that the required buffering capability of the former method has been seen as a large hardware overhead.

Due to its relative unpopularity, VCT has not been as heavily researched as WH. However, this paper will show that VCT, in particular adaptive VCT, has significant performance benefits over WH and can be implemented with minimal hardware overhead. It is shown that adaptive VCT has several attractive features when compared to WH: negligible overhead for avoiding deadlock (a situation in which no packet can proceed toward its destination), use of 100% of the free channels, good performance with heavy network traffic, and good performance with random or arbitrary locations of destination nodes. The hardware complexity issue is addressed by showing that VCT can be implemented with little hardware overhead (in some cases, less than WH), by exploiting the source buffer in the network interface to the router.

## 2. BACKGROUND

The type of parallel computer assumed will be a multicomputer with a *direct interconnection network*, in which a communication link can only be used by the two end-nodes that it is connected to. Adjacent processing nodes are assumed to be connected by two directed communication links in opposite directions (although undirected links are also possible, directed links are assumed for simplicity). A communication link consists of one or more data lines and several control lines. One or more *physical channels* may occupy a single communication link by using time multiplexing or by partitioning the set of data lines. One or more *virtual channels* may occupy a single physical channel by using separate flit buffers and control lines for each virtual channel. When two or more virtual channels are active over a single physical channel, they must be time-multiplexed; however, when only one of the virtual channels is active, it can utilize the full bandwidth of the physical channel. It is also assumed that messages are sent in fixed-size packets.

All communication methods must deal with the problem of *deadlock*, which occurs when there is a cycle of packets in which no packet can progress toward its destination because it is blocked by some other packet. WH switching is particularly susceptible to deadlock because blocked packets remain in the network, thereby reducing the number of free channels. Although much less likely, deadlock is also possible with VCT as the packet buffers at intermediate nodes can become full. The use of virtual channels was introduced by Dally and Seitz [1] as a means of guaranteeing *deadlock-free* WH switching. The WH routing algorithms introduced in [1] are *deterministic* algorithms, in which the routing path used is completely determined by the source and destination node addresses. In [3], Dally showed that virtual channels could also be used with a deterministic WH routing algorithm to significantly reduce the probability of blocking, and thus increase the throughput of the network.

Realizing the need for adaptability in routing around congested or faulty nodes and links, several researchers [4, 5] used virtual channels to produce *adaptive* deadlock-free WH routing algorithms, in which a packet is permitted to change its routing path if a blocked channel is encountered. Adaptive algorithms can be classified into *minimal* [6] and *nonminimal* [4, 7] algorithms, where a minimal algorithm always uses the shortest-length paths to each destination, and *fully adaptive* [4] and *partially adaptive* [6, 7] algorithms, where the former differs from the latter in that only a subset of the possible paths may be used.

## 3. ADAPTIVE VIRTUAL CUT-THROUGH

Wormhole switching methods have several deficiencies when compared to VCT. First, WH in general has a lower saturation throughput than a comparable VCT method as blocked packets contribute to network congestion in WH; however, this problem is significantly alleviated by the use of multiple-flit flit buffers and/or multiple virtual channels. Second, in WH based on deadlock avoidance, even with the use of virtual channels, 100% of the free channels (and flit-buffers) are not available for use by a given packet as some of the channels have to be reserved in order to prevent deadlock. Also, in WH based on deadlock recovery, extra hardware overhead (deadlock buffer, timer, and crossbar) is necessary to detect and recover from deadlock [8–10]. Third, fully adaptive routing in torus networks is difficult and requires extra virtual channels to be used for deadlock avoidance. Although virtual channels do not waste channel bandwidth, virtual channels do occupy valuable hardware resources because each virtual channel requires a separate on-line flit buffer and control line and contributes to the complexity of the arbitration logic within the router (refer to Section 4).

For the above reasons, adaptive VCT is proposed as an alternative to WH. When originally proposed in [2], VCT was described as a deterministic routing algorithm. In deterministic VCT, when the head of a packet is received at an incoming channel, the header is decoded to determine the "one" outgoing channel to connect to. If that outgoing channel is busy, then the packet is buffered at the current node. This method is analogous to deterministic WH as originally described by Dally and Seitz [1]. Thus, it is natural to consider an adaptive version of virtual cut-through in order to be able to navigate around congested or faulty nodes/links.

In order to facilitate discussion of the many possible forms of adaptive VCT, a general switching model, shown in Fig. 1, is proposed. When the head of a packet is received at an incoming channel, the destination node information is extracted from the header. A connection is then attempted to the first-choice outgoing channel. If successful, then a cut-through is established through the current node as denoted by (0) in Fig. 1. However, if the requested outgoing channel is blocked (1) because it is being used by another packet, then a connection is attempted to the second-choice outgoing channel (2). This process is repeated until a cut-through is achieved or all outgoing channels permitted by the adaptive VCT algorithm are exhausted, as is the situation in Fig. 1. In this situation, the blocked packet remains in the network (as in WH) until one of the permissible outgoing channels becomes available or the "permitted waiting time" (PWT) expires. In the latter case, the packet is received at the current node (3) and inserted into the source buffer of the current node just as if the packet originated there (the circuit implementation is discussed in Section 4). The packet at the head of the source buffer waits until one of the outgoing channels permitted to it becomes available, and then exits through that outgoing channel, as shown by (4) in Fig. 1. The ejection latency out of and the reinjection latency into the network for blocked messages in intermediate nodes are alleviated with the aid of a specially designed network interface controller.

Based on the above switching model, simulations were conducted with several different variations of adaptive VCT. In [11], we showed that *deflection VCT*, based on the deflection routing methods proposed for computer networks with packet switching [12], adaptive VCT with PWT > 0, and adaptive VCT with the

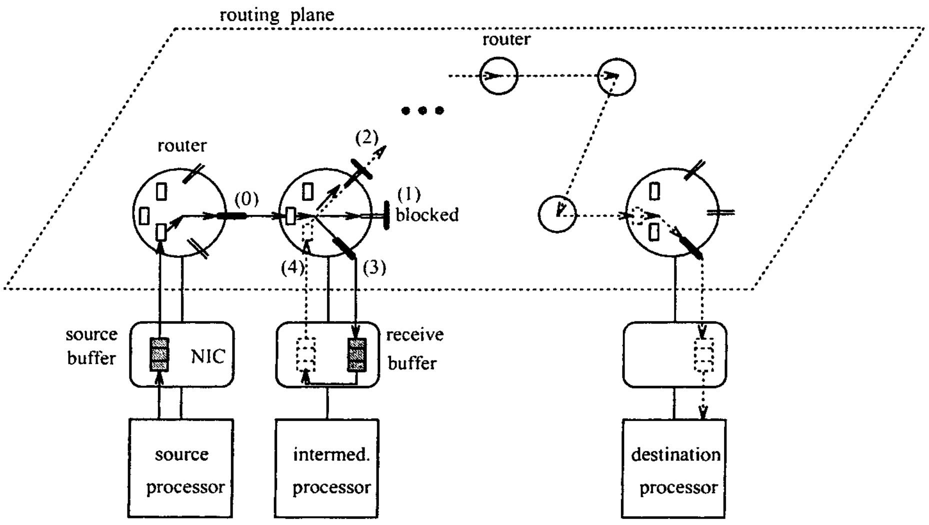

**FIG. 1.** A general switching model for adaptive VCT.

choice of adaptivity limited to a fixed number were inferior to adaptive *minimal-path* VCT with PWT = 0. In adaptive minimal-path VCT, the packet is permitted to choose between all possible minimum-length paths to the destination. Also, within adaptive minimal-path VCT, it is possible to attempt to follow a "zig-zag" path (algorithm $A_{\mathrm{mc}}$) or to simply follow a "priority-$X$-direction" path (algorithm $A_{\mathrm{ms}}$). Again, in [11], we showed that $A_{\mathrm{mc}}$ performs worse than $A_{\mathrm{ms}}$ with certain destination node distributions and is only slightly better than $A_{\mathrm{ms}}$ with other destination node distributions. Thus, this paper uses the $A_{\mathrm{ms}}$ adaptive minimal-path VCT algorithm, in which the switching model of Fig. 1 is used and minimal paths are followed with the $X$-direction attempted first.

The $A_{\mathrm{ms}}$ algorithm deals with deadlock in the following manner. Since VCT buffers blocked packets, the deadlock conditions for VCT are identical to packet switching—thus, the methods used to avoid deadlock in packet switching can be used [13, 14]. In our hardware implementation of adaptive VCT, the "structured buffer pool technique" for deadlock avoidance [13] was used because of its simplicity. In this method, the source buffer is partitioned into $K$ buffer classes, where $K$ is the diameter of the network. The $i$th buffer class, also called the $i$-hop class, may only hold those packets that have $i$ of fewer hops to go to their respective destinations. It is well known that this technique ensures deadlock-free store-and-forward packet switching [13]. Our implementation in Section 4.3 shows that the source buffer can be implemented in a small SRAM (or even main memory)—in fact, several researchers are investigating the injection of packets into the network directly from memory, thus avoiding intermediate buffer copies [15, 16]. The number of buffers in the NIC required to guarantee deadlock freedom is high and depends on the network size. There are other techniques to reduce those buffer requirements to small constant values regardless of network size [17, 18]. However, at the time we initiated the NIC design, we were not aware of these techniques.

## 4. HARDWARE IMPLEMENTATION

### *4.1. Communication Architecture*

The proposed communication architecture for supporting adaptive VCT is shown in Fig. 2. As can be seen from this figure, there is a network interface controller (NIC), source and receive buffers, and a processor responsible for preparing and sending the packets out onto the network. The NIC is logically partitioned into the external and internal NI. The external NI responds to the network and the internal NI responds to the processor and memory [19]. Packets are initially stored in the source buffer before being sent out to the router chip. The router chip is connected to other router chips in a connection pattern determined by the interconnection network topology used. A packet which is injected into a router chip from a processing node is routed to the router chip connected to the destination node. Packets to be received by the current node are initially buffered in the receive buffer before being transferred to local memory.

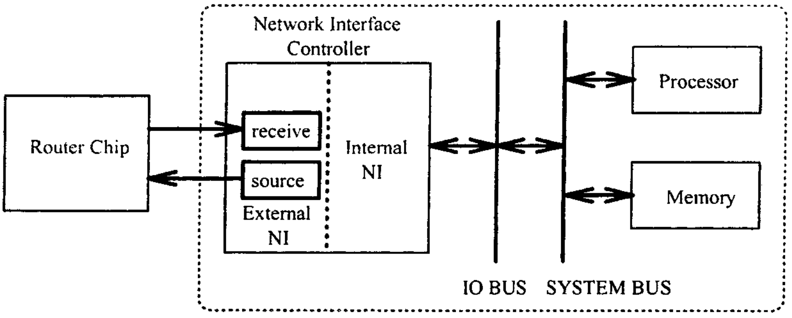

**FIG. 2.** Communication architecture.

To demonstrate the feasibility of adaptive VCT, the $A_{\mathrm{ms}}$ algorithm was implemented in a semi-custom-designed VLSI router chip and NIC chip. The complete hardware implementation of the $A_{\mathrm{ms}}$ algorithm in a multicomputer would involve using our router and NIC chips in the network interface. Upon receiving a new packet in the receive buffer, the NIC must examine the header to determine if the packet is destined for the current node or is "in transit." In the latter case, the packet must be rerouted to the source buffer to be transmitted when one of its permitted outgoing channels becomes available. (Note that bandwidth matching requirements imply that the data transfer rates in the source and the receive buffers must match the data transfer rates within the router network.) In order to perform this rerouting efficiently without having to contend for the local bus, a dedicated path must be established to the source buffer within the external NI—arbitration with packet insertion from the local processing node (on a FCFS basis) is of course necessary.

## 4.2. *An Adaptive VCT Router Chip*

A block diagram of our router chip, designed to implement the $A_{\mathrm{ms}}$ adaptive VCT algorithm in a two-dimensional mesh or torus network, is shown in Fig. 3. As can be seen, there are single-flit buffers at the receivers and transmitters and a crossbar switch connecting the receivers to the transmitters. The crosspoint logic includes arbitration logic to handle contention for common transmitters (outgoing channels). Finite state machine control logic in the receiver implements algorithm $A_{\mathrm{ms}}$ by first attempting to connect to the first-choice transmitter, then the second-choice transmitter (if one exists), and then the PN (processing node) transmitter (for reception) if none of the first two choices are available. Note that in our prototype chip, 6-bit-wide data lines were used to connect to other routers and the local processing node due to a shortage of I/O pins. Note also that the "permitted waiting time" used is 0.

The router chip was designed using Mentor Graphics design tools, manufactured using Samsung's 1.0-$\mu$m CMOS process, and successfully tested. The arbitration capabilities of the crossbar switch were successfully tested and the $A_{\mathrm{ms}}$ adaptive routing algorithm worked as designed. The chip required 33,000 transistors, consumed 800 mW of power, and was tested successfully up to a clock frequency of

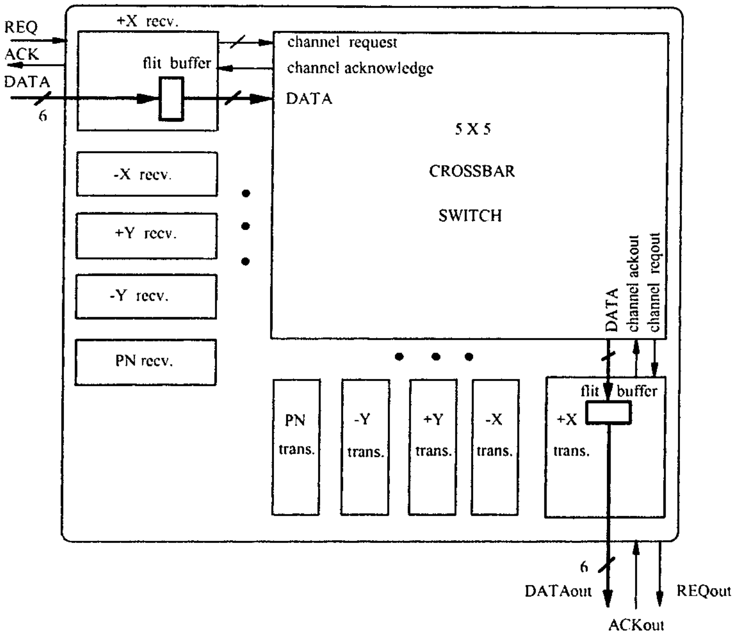

**FIG. 3.** Block diagram of the $A_{ms}$ adaptive VCT router chip.

12.5 MHz. The layout for this chip is shown in Fig. 4. The placement of the various blocks for the router design are approximately the same as in Fig. 3.

The hardware overhead required for our adaptive VCT router chip was estimated by measuring the layout area required by our chip as compared to the layout area required by a comparable deterministic WH router chip. This comparison is shown in Table 1. When our chip is compared to a deterministic WH router chip without multiple virtual channels, the overhead is 2.3%. When our chip is compared to a deterministic WH router chip with two virtual channels per physical channel (to

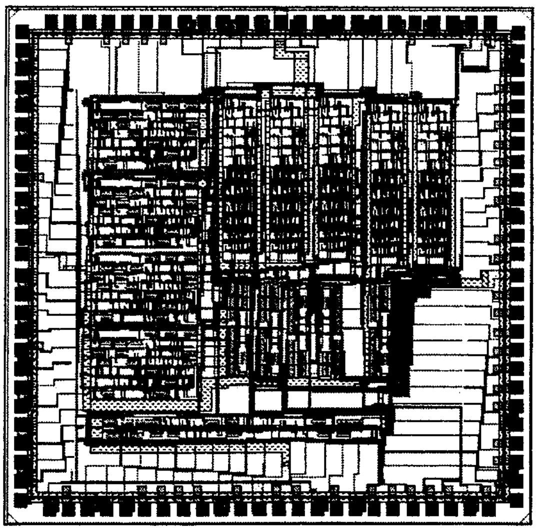

**FIG. 4.** Layout of the completed router chip.

**TABLE 1**

**Router Chip Layout Area Comparison**

| Routing algorithm | Total Area ( $mm^2$ ) |
|---|---|
| (1) Deterministic WH | 18.699 (4.278 × 4.371) |
| (2) Adaptive VCT | 19.136 (4.367 × 4.382) |
| (3) Deterministic WH with 2 virtual chan. | 35.405 (6.098 × 5.806) |
| (4) Adaptive VCT with 2 virtual chan. | 38.763 (6.147 × 6.306) |

avoid deadlock), our chip is smaller by 85.0%, even though the size of the crossbar is kept constant [3]. Since this latter chip has higher performance, however, it should perhaps be compared to an adaptive VCT chip with two virtual channels per physical channel—in this case, the overhead for our chip is 9.5%. The significance of a smaller chip area is that the router can easily be implemented in one corner of a multiple-capability chip (such as a processor + memory + NIC + router chip). It should be noted, however, that this comparison is only a rough estimate, as the layout is not optimized.

It is noted that in a practical VCT router implementation it is more preferable to use multiple-flit buffers, such as in [20, 21], rather than single-flit buffers since single-flit buffers can result in bubbles and reduced throughput.

## 4.3. Network Interface Controller Design

Figure 5 shows a block diagram of our NIC chip, which is designed to interface to our router chip and support the $A_{ms}$ adaptive VCT algorithm. The send queue is a queue of "send packet requests." If an entry is detected at the head of the send

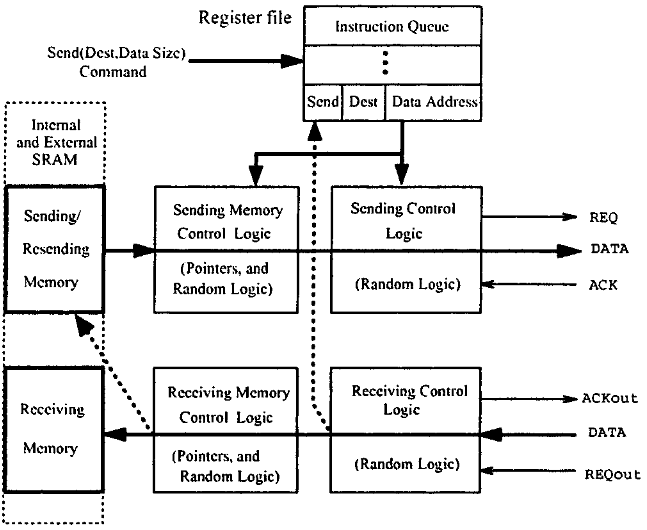
Register file
Instruction Queue
Send(Dest,Data Size) Command
Send
Dest
Data Address
Internal and External SRAM
Sending/ Resending Memory
Sending Memory Control Logic
(Pointers, and Random Logic)
Sending Control Logic
(Random Logic)
REQ
DATA
ACK
Receiving Memory
Receiving Memory Control Logic
(Pointers, and Random Logic)
Receiving Control Logic
(Random Logic)
ACKout
DATA
REQout

FIG. 5. Block diagram of the network interface controller chip.

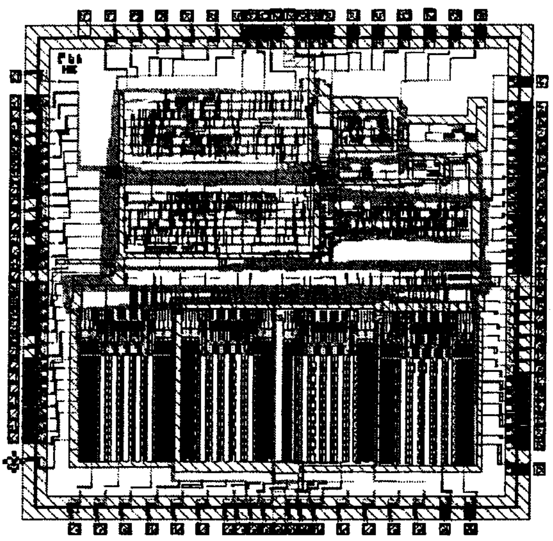

**FIG. 6.** Layout of the network interface controller chip.

queue, the sending control logic forms a packet header from the destination information and sends it out followed by the packet body. The packet body is sent out from the source buffer starting at the address pointed to by "data address." The receiving control logic, upon receiving the header of a packet, determines if the packet is destined for the current node or is simply being stored temporarily. In the former case, the packet is stored in the receive buffer. In the latter case, the received packet is stored in the source buffer and an entry is appended to the tail of the send queue. The only addition required to the basic NIC design to implement the $A_{\mathrm{ms}}$ adaptive VCT algorithm is a flit buffer and some extra control logic to send temporarily buffered messages to the source buffer.

In our NIC implementation, the source buffer and receive buffer are implemented as SRAMs with a possible maximum size of 32 K words (256 words in an internal SRAM and the rest in an external SRAM) and a word size of 6 bits (to interface to our 6-bit router chip). Logically, the source buffer is partitioned into two partitions. In the first partition, which we arbitrarily chose to be 2 K words, packets are stored in any "packet slot" as they are received. The second partition is sub-partitioned into $K$ buffer classes to implement the "structured buffer pool technique" for deadlock avoidance described at the end of Section 3. This second partition is used only when the first partition is completely filled. Our simulation results at the end of Section 5 show, however, that we almost never have to use this second partition.

The NIC chip was designed using a 0.8-$\mu$m CMOS process and is currently awaiting fabrication. Figure 6 shows the layout for the NIC chip.

## 5. SIMULATIONS

Computer simulations were used to compare the performance of the $A_{\mathrm{ms}}$ adaptive VCT algorithm and WH methods. Simulations were conducted for various topologies, network sizes, packet sizes, flit buffer sizes, and multiple virtual channels. A 16 $\times$ 16 two-dimensional mesh topology was chosen to allow meaningful comparison with deterministic and adaptive WH routing algorithms as most WH routing algorithms do not work on a two-dimensional torus with only one virtual channel per communication link. Note that the adaptive VCT algorithms, by contrast, work efficiently (without deadlock) on any network topology, using only one virtual channel per communication link. In the simulations for the $A_{\mathrm{ms}}$ algorithm, the ejection and reinjection overhead for an in-transit buffered packet were modeled based on the delays measured with the circuit simulations for our NIC hardware implementation.

In our simulations, a 20-flit packet size is assumed, and packets are transmitted one flit at a time through the network, with one flit being transferred in one clock cycle time. Each node generates new packets according to a Poisson distribution with rate $\lambda/L$, where $\lambda$ is the applied load and $L$ is the length of the packets. Destination nodes are selected using two methods, one based on a random uniform distribution and one based on a clustered distribution. Although we have experimented with several clustered distributions, for the sake of brevity only the results for the matrix transpose distribution are shown in this paper.

The simulation results are presented with the offered traffic, $\lambda$, along the $X$-axis and the average network latency on the Y-axis. Note that, as in [3], the offered traffic has been normalized by the maximum possible throughput, which on a $16 \times 16$ mesh with uniform random traffic is 64 flits/cycle. The latency is defined as the time from when a packet is first generated and stored in the source buffer to when the tail of the packet leaves the network and enters the receive buffer of the destination node. All times are measured in terms of flit cycles. Each simulation is executed for 30,000 cycles, with statistics collected for only the last 20,000 cycles in order to avoid transient startup effects.

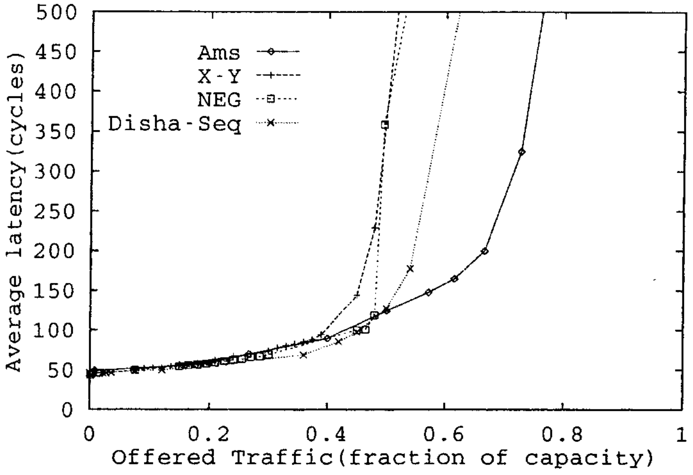

**FIG. 7.** Average latency comparison with deterministic and adaptive WH routing algorithms given a 16x16 mesh and 20-flit packets, a uniform destination node distribution, 2 flit buffers per virtual channel and 4 virtual channels per link.

Figure 7 shows a comparison of the $A_{ms}$ algorithm with deterministic and adaptive WH routing algorithms based on a uniform destination node distribution. The deterministic WH routing algorithm used is the $X$–$Y$ routing algorithm in which the packet is first sent in the $X$ direction and then in the $Y$ direction. The adaptive WH routing algorithms simulated are Glass and Ni's negative-first algorithm based on the turn model [7] (NEG), which is a partially-adaptive minimal algorithm, and Disha's WH routing algorithm based on deadlock recovery (Disha-Seq), which is a fully-adaptive algorithm [8, 9]. The Disha-Seq algorithm is simulated with a time out of 80, the recommended value. At low traffic loads, all methods perform similarly, with Disha-Seq showing slightly better performance than the other algorithms. However, when comparing the saturation point, it can clearly be seen that the $A_{ms}$ algorithm has the best performance—only the $A_{ms}$ algorithm is capable of supporting a 75% traffic load.

Figure 8 shows the results for a matrix transpose destination node distribution. With a matrix transpose destination node distribution, the Disha-Seq algorithm again has the best performance at low traffic loads and the $A_{ms}$ algorithm has the highest network saturation point.

Figures 9 and 10 show the results of using 16 virtual channels per communication link (with 32-flit buffers allocated to each virtual channel). The interested reader is referred to [3] and [22] for the effects of larger numbers of virtual channels and larger buffers. Figure 9 compares the $A_{ms}$ algorithm with deterministic and adaptive WH routing algorithms assuming 20-flit packets, two virtual channels per link, and a uniform destination node distribution. Figure 10 shows a similar comparison assuming a matrix transpose destination node distribution. It can be seen that while the performance of all algorithms have improved significantly from Figs. 7 and 8, the $A_{ms}$ algorithm again has the highest network saturation point.

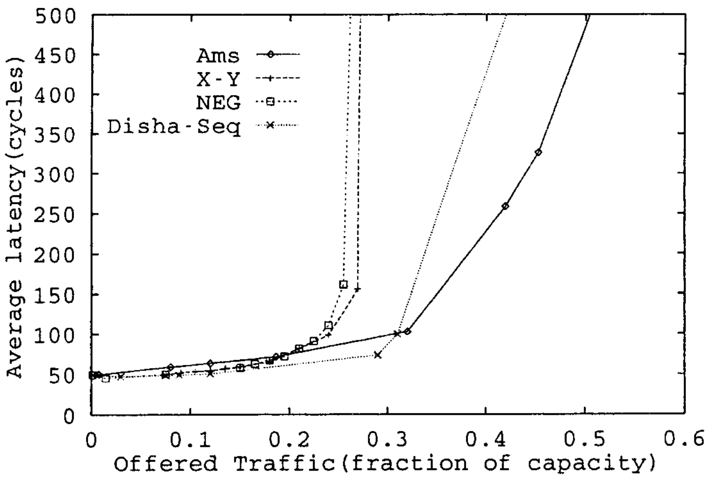

**FIG. 8.** Average latency comparison with deterministic and adaptive WH routing algorithms given a 16 x 16 mesh and 20-flit packets, a matrix transpose destination node distribution, 2 flit buffers per virtual channel and 4 virtual channels per link.

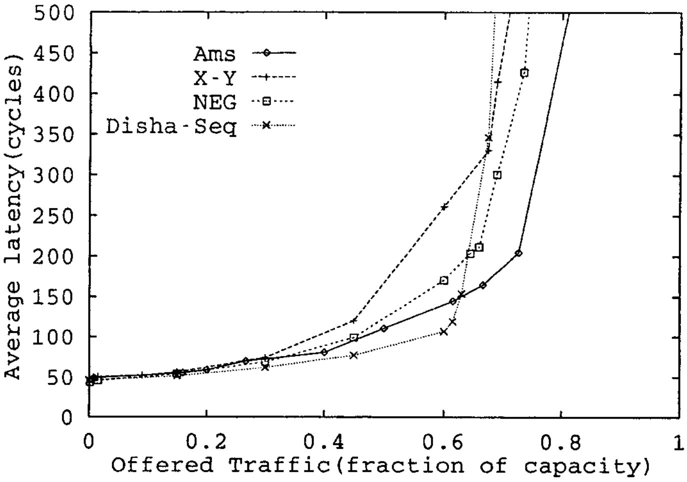

**FIG. 9.** Average latency comparison with deterministic and adaptive WH routing algorithms given a 16 x 16 mesh, 20-flit packets, a uniform destination node distribution, 32 flit buffers per virtual channel and 16 virtual channels per link.

During all simulations with the $A_{ms}$ algorithm, records were kept of the maximum source buffer lengths observed. This is important as our adaptive VCT method is dependent on using the source buffer in the NIC to buffer temporarily blocked packets. The maximum of the source buffer lengths observed under very heavily-loaded network conditions in all of our simulations (taken to be the point when the average latency was over 70 times the minimum average latency—the network will rarely be used or usable at loads beyond this point) was only 11 packets. Thus, in this case, with a packet length of 20 flits, a 220-flit source buffer is sufficient to support adaptive VCT.

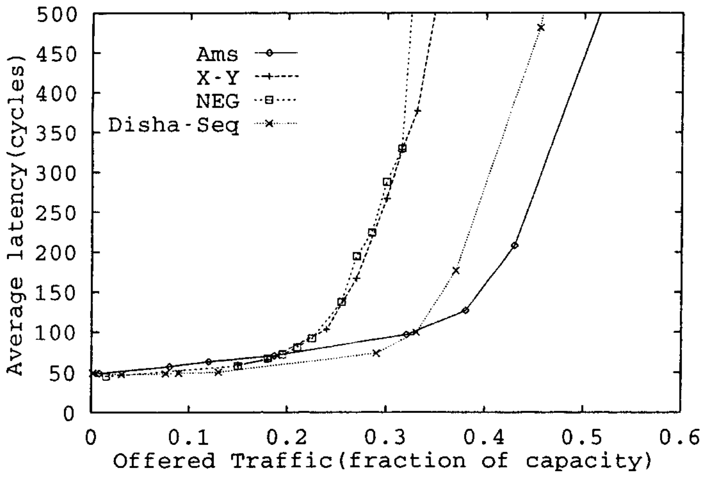

**FIG. 10.** Average latency comparison with deterministic and adaptive WH routing algorithms given a 16 x 16 mesh, 20-flit packets, a matrix transpose destination node distribution, 32 flit buffers per virtual channel and 16 virtual channels per link.

## CONCLUSION

This paper has investigated adaptive VCT as a viable alternative to WH switching. It has been shown that adaptive VCT has several attractive features when compared to WH: negligible overhead for avoiding deadlock (in terms of both performance and hardware), the possibility of using 100% of the free buffers and channels, good performance with heavy network traffic, and good performance with uniform or clustered locations of destination nodes. Simulations have been used to investigate the performance of a minimal-path adaptive VCT algorithm ($A_{\mathrm{ms}}$). When compared to other deterministic and adaptive WH routing algorithm, the results show that $A_{\mathrm{ms}}$ has a significantly higher network saturation point, and only slightly poorer performance at low network traffic loads. Finally, the hardware complexity issue was addressed by showing that VCT can be implemented with little hardware overhead (in some cases, less than WH) by exploiting the source buffer in the network interface to the router.

## REFERENCES

1. W. Dally and C. L. Seitz, Deadlock-free message routing in multiprocessor interconnection networks, *IEEE Trans. Comput.* **C-36** (May 1987), 547–553.
2. S. Kermani and L. Kleinrock, Virtual cut-through: A new computer communication switching technique, *Computer Networks* **3** (1979), 267–286.
3. W. J. Dally, Virtual-channel flow control, *IEEE Trans. Parallel and Distrib. Systems* **3** (Mar. 1992), 194–205.
4. W. J. Dally and H. Aoki, Deadlock-free adaptive routing in multicomputer networks using virtual channels, *IEEE Trans. Parallel and Distributed Systems* **4** (Apr. 1993), 466–475.
5. J. Duato, A new theory of deadlock-free adaptive routing in wormhole networks, *IEEE Trans. Parallel and Distributed Systems* **4** (Dec. 1993), 1320–1331.
6. Y. M. Boura and C. R. Das, A class of partially adaptive routing algorithms for *n*-dimensional meshes, *in* "International Conference on Parallel Processing, Aug. 1993," Vol. III, pp. 175–182.
7. C. J. Glass and L. M. Ni, The turn model for adaptive routing, *in* "Proc., 19th International Symposium on Computer Architecture, 1992," pp. 278–287.
8. K. V. Anjan and T. M. Pinkston, An efficient, fully adaptive deadlock recovery scheme: Disha, *in* "22nd Annual International Symposium on Computer Architecture, June 1995," pp. 201–210.
9. K. V. Anjan and T. M. Pinkston, Disha: A deadlock recovery scheme for fully adaptive routing, *in* "9th International Parallel Processing Symposium, Apr. 1995," pp. 537–543.
10. Y. Choi and T. M. Pinkston, Crossbar analysis for optimal deadlock recovery router architecture, *in* "11th International Parallel Processing Symposium, June 1997".
11. H. S. Lee, H. W. Kim, J. Kim, and S. Lee, Adaptive virtual cut-through as an alternative to wormhole routing, *in* "International Conference on Parallel Processing, Aug 1995," pp. 68–75.
12. N. F. Maxemchuk, Problems arising from deflection routing: live-lock, lockout, congestion and message reassembly, *in* "Proc. of NATO Workshop on Architectures and Performance Issues of High Capacity Local and Metropolitan Area Networks, May 1990."
13. M. Gerla and L. Kleinrock, Flow control: A comparative survey, *IEEE Trans. Comm.* **COM-28** (Apr. 1980), 553–574.
14. D. Gelernter, A dag-based algorithm for prevention of store-and-forward deadlock in packet networks, *IEEE Trans. Comput.* **C-30** (Oct. 1981), 259–270.

15. F. Hady, R. Minnich, and D. Burns, The memory integrated network interface, *in* "Hot Interconnects II Symposium Record, Aug. 1994," pp. 10–22

16. C. Dubnicki, A. Bilas, K. Li, and J. Philbin, Design and implementation of virtual memory-mapped communication on myrinet, *in* "11th International Parallel Processing Symposium, Apr. 1997."

17. R. Cypher and L. Gravano, Requirements for deadlock-free, adaptive packet routing, *in* "11th Symposium on Principles of Distributed Computing, Aug. 1992," pp. 25–34.

18. J. Duato, Necessary and sufficient condition for deadlock-free routing in cut-through and store-and-forward networks, *IEEE Trans. Parallel and Distributed Systems* **7** (Aug. 1996), 84–854.

19. S. S. Mukherjee and M. D. Hill, "A Survey of User-Level of Network Interfaces for System Area Networks," Tech. Rep. 1340, C.S. Dept., University of Wisconsin–Madison, Feb. 1997.

20. S. Konstantinidou and L. Snyder, The chaos router, *IEEE Trans. Comput*. **43** (Dec. 1994), 1386–1397.

21. A. G. Nowatzyk, M. C. Browne, E. J. Kelly, and M. Parkin, S-connect: From networks of workstations to supercomputer performance, *in* "22nd Annual International Symposium on Computer Architecture, June 1995," pp. 71–82.

22. A. A. Chien, A cost and speed model for *k*-ary *n*-cube wormhole routers, *in* "Hot Interconnects '93, Palo Alto, CA, Aug. 1993."