INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA

UMI®
800-521-0600

DISSERTATION

EVALUATION OF ROUTING ALGORITHMS AND THEIR IMPLEMENTATIONS

Submitted by

Dianne Kumar

Department of Computer Science

In partial fulfillment of the requirements

for the Degree of Doctorate of Computer Science

Colorado State University

Fort Collins, Colorado

Summer 1999

COLORADO STATE UNIVERSITY

May 27, 1999

We hereby recommend that the **Dissertation** prepared under our supervision by Dianne Kumar entitled **EVALUATION OF ROUTING ALGORITHMS AND THEIR IMPLEMENTATIONS** be accepted as fulfilling in part requirements for the degree of Doctorate of Computer Science.

Committee on Graduate Work

Committee Member: Dr. Anura P. Jayasumana

Committee Member: Dr. Yashwant K. Malaiya

Committee Member: Dr. Pradip K. Srimani

Adviser: Dr. Walid A. Najjar

Department Head: Dr. Stephen B. Seidman

ABSTRACT OF **DISSERTATION**

EVALUATION OF ROUTING ALGORITHMS AND THEIR IMPLEMENTATIONS

Routing algorithms in interconnection networks can greatly impact system performance. Deterministic routing is simple resulting in small cycle times but does not perform well under congestion. Adaptive routing does perform well under congestion but is more complex resulting in high router cycle times. This dissertation offers a comprehensive evaluation of the routing algorithms for k-ary n-cube networks under virtual cut-through switching. The results include router cycle times for each routing algorithm simulated. A pipelined hybrid routing scheme is then proposed combining the advantages of small deterministic router cycle times with adaptive router flexibility.

Multicast communication is a common operation in parallel computing and its efficient implementation is critical to the performance of multiprocessors. In this dissertation, a hardware tree-based routing algorithm (HTA) is presented for multicast communication. HTA is designed to keep the probability of message blocking at each intermediate node along a message's path low. By keeping the blocking probability low, the probability of deadlock is reduced. The algorithm is fully compatible with existing unicast routing schemes and relies on deadlock detection and recovery.

Dianne Kumar
Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523
Summer 1999

ACKNOWLEDGEMENTS

I am extremely grateful to my advisor, Dr. Walid Najjar, for his invaluable guidance, advice and support throughout my research. My thanks also goes to Dr. Anura Jayasumana, Dr. Yashwant Malaiya, Dr. Pradip Srimani and Dr. Dale Grit for all their help and assistance.

I am grateful to my family and friends for their support, encouragement and prayers. An extra special thanks goes to my parents who have taught me the value of education and determination.

Finally, I would like to thank my husband and my best friend, Sushil, for his constant encouragement, support and love. I would never have been able to complete my program without him.

DEDICATION

To my husband and best friend, Sushil

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Interconnection networks are used in many different applications. They can provide communication between processors within close proximity of each other (as in a multiprocessor system) or they can provide communication between computers as far apart as those in a wide area network. Each type of interconnection network has its own unique set of requirements and each requires a different solution to using the network efficiently.

This dissertation focuses on interconnection networks in multiprocessor systems. The inter-processor communication speed is a major factor in the performance of massively parallel processor systems and can often become its limiting bottleneck. Understanding the relationship between different parameters in an interconnection network and implementing these parameters effectively can result in fast and efficient processing in a multiprocessor system.

1.1 Interconnection Network Issues

The performance of interconnection networks depends on several factors such as network topology, routing algorithm, router design and implementation [DYN97].

Topology defines how the nodes in the network are interconnected by the channels. Some of the characteristics which describe network topology are bisection width, channel width, channel direction, network diameter, symmetry and node connectivity. No optimal topology exists for all applications and implementation of topologies is physically limited.

For example, a five-dimensional network may give excellent theoretical performance. However, its implementation is impractical in physical space and results in high cycle times due to its long wire connections and high complexity. On the other hand, two dimensional meshes have high package density and allow for transfer of high speed signals along short wires between the routers.

The routing algorithm determines the path taken by a message while traveling from its source to its destination. In dimension-order routing, a message is routed along decreasing dimensions with a dimension decrease occurring only when zero hops remain in all higher dimensions. Virtual channels (VCs) are included in the router to avoid deadlock [DS87b]. However, deterministic routing algorithms can suffer from congestion since only a small subset of all possible paths between a source and destination are used.

In adaptive routing, messages are not restricted to a single path when traveling from source to destination. Moreover, the choice of path can be made dynamically in response to current network conditions. Such schemes are more flexible, can minimize unnecessary waiting, and can provide fault-tolerance. Several studies have demonstrated that adaptive routing can achieve a lower latency, for the same load, than deterministic routing when measured by a constant clock cycle for both routers [GN92, Lag94].

The delay experienced by a message at each node can be broken down into: *router* delay and *queuing* or *buffering* delay. The former is determined primarily by the complexity of the router. The later is determined by the congestion at each node which in turn is determined by the degrees of freedom the routing algorithm allows a message. The main performance advantage of adaptive routing (besides its fault-tolerance) is that it reduces the queuing delay.

However, the delay for deterministic routers, and consequently their corresponding clock cycles, can be significantly lower than adaptive routers as pointed out in [Chi93, AC95]. This difference in router delays is due to two main reasons:

- *Number of VCs:* Two VCs are sufficient to avoid deadlock in dimension ordered routing [DS87b]; while adaptive routing (as described in [Dua93] and [BGPS92]) requires a minimum of three VCs in k-ary n-cube networks.

- *Output channel selection:* In dimension-ordered routing, the output channel selection policy is very simple: it depends only on information contained in the message header itself whereas in adaptive routing the output channel selection policy depends also on the state of the router (i.e the occupancy of various VCs) causing increased router complexity and thereby higher switching delays.

Because of these differences in complexity, the switch delays for adaptive routers can be much greater than those for deterministic routers. The results in [Chi93, AC95] show that the switch delays for the various adaptive routers are about half to more than twice as long as the dimension-order router for wormhole routing. These results, however, do not account for the advantage of adaptive routing in reducing queuing delays in the nodes between source and destination. Furthermore, the various routing algorithms evaluated, both deterministic and adaptive, require a variable amount of resources such as buffer area or physical channels between nodes. The advantage of adaptive routing in reducing queuing delays in the nodes between source and destination is evaluated and reported in [DL94] for wormhole routing.

1.2 Multicast Communication

Multicast communication is a common operation in parallel computing and efficient implementation of multicast is critical to the performance of multiprocessors. Since most

current multiprocessors only support unicast communication, multicast is implemented as multiple unicast messages resulting in high message latencies.

Hardware-based schemes can greatly improve performance but deadlock freedom is much more difficult in multicast communication, resulting in more involved routing algorithms and higher startup delays. Among the proposed hardware-based schemes for multicast are path-based and tree-based routing algorithms [BCR98, RMC97, KP96, MDT96, PSP94, BCR94, LN91, BSD89, BND87]. Each one of these schemes uses multidestination messages, which are messages that have more than one header flit. The main difference between these two multicast schemes lies in how the header flits in a multidestination message are routed.

In tree-based routing a multidestination message is routed through the network at each intermediate node along a multidestination message's path using *all* header flits in the message. In path-based routing a multidestination message is routed through the network using only the first header flit in the message. Once the first header flit reaches its destination and is absorbed by the node, the next header flit is routed.

In tree-based routing, no ordering of the destinations is required before the message is injected into the network and the shortest path between the source node and all destinations is always taken. However, this type of routing suffers from a high probability of message blocking at intermediate nodes. Path-based routing does not suffer from this high probability of message blocking. However, it does require destinations to be ordered at the source and does not always provide the shortest path between the source node and each destination node.

1.3 Dissertation Outline

The purpose of this dissertation is to report on the performance of deterministic, adaptive, and hybrid routers for k-ary n-cube networks evaluated under various implementation

constraints and to present a multicast routing algorithm which is simple to implement and is fully compatible with existing unicast routing schemes.

The dissertation is organized as follows: Background material is given in Chapter 2. It includes design parameters of interconnection networks, router complexity issues, hybrid techniques, examples of machine implementations, collective communication services, encoding schemes for multidestination messages, and software and hardware multicast algorithms. Performance issues in adaptive routers are discussed in Chapter 3. Topics include input/output selection policies, adaptive router performance, physical wire bandwidth, and effects of router delay on performance. In Chapter 4, the hybrid deterministic/adaptive router is presented. This chapter gives a description of the hybrid routing scheme, its implementations and clock cycle times, and presents the experimental results. A hardware tree-based multicast routing algorithm is explained in Chapter 5. It includes a description of the routing scheme, deadlock recovery mechanism, router implementation, and experimental evaluation.

Chapter 2

BACKGROUND

The background information concerning interconnection network design is given in this chapter. Section 2.1 discusses the main parameters of network design: topology, flow control, and routing algorithm. Recent research concerning router complexity and its affects on performance is explained in Section 2.2.

2.1 Design Parameters of Interconnection Networks

The main parameters of interconnection network design are topology, flow control, and routing algorithm. The optimal choice of each parameter may be different depending on the interconnection network's use and each parameter can either adversely or positively affect the other parameters. Therefore, not only is it important to understand each parameter individually, but also how each inter-relates with the others.

2.1.1 Topology

Topology defines how the nodes in the network are interconnected by the channels. There exist two types of topologies: direct and indirect. Direct networks (also called point-to-point networks) are composed of nodes, each of which contain routers that are incorporated as part of the processing elements (PEs). These routers are directly connected to a subset of routers in other network nodes. Some examples of direct networks are meshes, tori, k-ary n-cubes, rings and trees [Hwa93, DYN97].

In indirect networks (also called switch-based networks) the PEs are not directly connected to each other. Instead switches which are independent of the PEs are organized in groups or stages, and a path connecting one PE to another goes through several of these stages. Some examples of indirect networks are the Banyan, Omega, and Delta [AG89, DYN97].

Some characteristics of the above direct and indirect networks are as follows:

- *Bisection Width:* Bisection width is defined as the number of channels that are cut when the network is divided into two equal parts. Bisection width is often one of the limiting factors in interconnection networks because it is directly related to two scarce commodities in chip design: wire density and chip area.

- *Channel Width:* Channel width is the number of wires per channel (or the number of bits that can be carried simultaneously on one channel). Under constant bandwidth, the number of bits per channel versus the number of channels is often an important design parameter and can affect the performance greatly.

- *Channel Direction:* Channels can be categorized as unidirectional or bidirectional. Bidirectional channels may or may not support transmissions in both directions simultaneously. Although bidirectional channels can reduce the average number of hops a message must travel (versus unidirectional channels), it is more complex and can reduce the available bandwidth for a message.

The topology used in this proposal is the k-ary n-cube [DS86, Hwa93], where k is the radix of the network and n is the network dimension. The number of nodes (N) in this topology is k^n. Some of the characteristics defining this topology are:

- *Node Degree (d):* Number of channels connecting a node to its neighbors ($d = 2n$). A network in which all the nodes have the same degree is called regular (k-ary n-cube networks are regular).

- *Network Diameter (D):* The maximum distance between two nodes in a network (measured by the number of traversed channels) ($D = n \lceil k/2 \rceil$)

- *Number of Links (l):* Total number of channels in the network (nN)

- *Symmetry:* A network which looks alike from every node (k-ary n-cube networks are symmetric)

- *Average Distance (δ):* The average distance that a message travels. (If both source and destination are unique, $\delta = n\frac{(k-1)}{2}\frac{k^n}{(k^n-1)}$)

Note that the k-ary n-cube has wrap-around edges. This can be both an advantage and disadvantage. The edges make the network regular and symmetric, simplifying analytical analysis. The edges also reduce the network diameter in a bidirectional network and allow for strong connectivity in a unidirectional network. However, wrap-around edges also introduce cycles within the topology, which complicates deadlock issues.

2.1.2 Flow Control

Flow control (also called switching) is a synchronization protocol for transmitting and receiving a unit of information and which prevents message overflow between nodes. Some of the important considerations involved in flow control are explained below:

1. **Channel Control Policy**

   This policy describes the techniques used for acquiring and relinquishing control of the channel and its four main types are described below. (Note that L represents message latency, l is message length, and δ represents the number of hops traveled.)

(a) **Circuit Switching:** In circuit switching, transmission only begins after a message acquires all channels along its path. Control is then relinquished after the entire message has reached its destination. Circuit switching works best for large messages because of the long path setup and removal stages. In addition, once a channel is obtained, the message can use the full bandwidth of the channel. Fewer buffers are required, at the expense of greater channel contention. In this case, $L = (l-1) + \delta + O$, where O is the overhead time necessary for path setup and removal and is proportional to 2δ.

(b) **Store-and-forward (Packet) Switching:** In this type of switching, a message is partitioned into fixed packet lengths which are then transmitted. Each packet contains a header (e.g. the first few bytes of a packet) holding routing information. This information is used to route each packet individually from source to destination, which may result in out-of-order delivery. In addition, the entire packet must be buffered at a node before another channel can be obtained. Store-and-forward switching works best for short and frequent messages because of the low overhead in acquiring and releasing channels and because a channel is held only when a packet is using it. The buffer requirements for this type of routing are much greater than that for circuit switching and, in this case, $L = l\delta$.

(c) **Wormhole Switching:** In wormhole switching, a packet is composed of small flow control units (called flits) which, unlike store-and-forward switching, can be sent along the path as soon as the header flit is routed and the output channel is free. Since the buffers at each node can only hold part of a packet, at any instance of time a packet occupies buffers in several routers. The advantage of wormhole switching is the low buffer requirements, while its main disadvantage

is the high channel contention resulting from the small buffers. In this case, $L = l + \delta$.

(d) **Virtual Cut-through Switching:** Virtual cut-through switching [KK79] has message advancement similar to wormhole switching [NM93], except that the body of a message can continue to progress even while the message head is blocked, and the entire message can be buffered at a single node. The advantage of this type of switching is that blocked messages can be held completely at one node, reducing the number of buffers occupied by a blocked message. Its main disadvantage is the large buffer sizes required at each node. The message latency is equivalent to wormhole switching message latency when no blocking exists ($L = l + \delta$).

2. **Message Selection Policy**

The two main types of message selection policy are the input and output selection policies. The input selection policy involves arbitrating between two or more messages at a node that request the same output channel, while the output selection policy involves deciding which output channel a message should be routed to. There exist many input/output selection policies, none of which are optimal for all cases.

3. **Collision Policy**

This policy decides the fate of messages not routed during the current clock cycle and include such methods as buffering, dropping, or misrouting these messages.

4. **Buffer Placement**

There are two main types of buffer placement in a router (input and output buffers), each of which can cause performance improvement or degradation depending on various implementation factors. One of the major factors affecting this performance is

the selection of the routing algorithm. Since this paper focuses on deterministic and adaptive routing, the advantages and disadvantages for each type of buffer placement with respect to these two routing types are discussed.

(a) **Input Buffering:** Input buffering is defined here as routing messages *after* they are put into the buffer at the current node. There are numerous advantages for input buffering. First, input buffering allows easier implementation of many different kinds of selection policies. The reason for this easier implementation is because no off-chip selection information lines are required since the incoming message is buffered on the same chip as the crossbar it uses. This allows all necessary communication for selection policies to be performed on-chip. Another advantage of input buffering is the absence of early commitment of messages. Instead, a message is only assigned an output channel when it is ready to be accepted by a downstream node. This is an advantage for any routing algorithm which allows several routing choices (such as partial or fully adaptive routing). Note that this is not an advantage for deterministic routing where a message has only one choice. The third advantage of input buffering is that its implementation requires less hardware than for output buffering. This is because an incoming message is immediately buffered before its routed. In output buffering a message must be immediately routed and then buffered. This difference, however, is reduced in a pipelined router. The main disadvantage for input buffering is the input queue head of line blocking problem. This problem can cause unnecessary blocking if more than one message is in the input queue. Since only the first message in the queue is seen by the router, only the first message has the possibility of being routed. If the first message is blocked due

to unavailable channels, the second message cannot be routed even if it has available channels [DYN97, Lag94].

(b) **Output Buffering:** In output buffering a message is routed *before* being put into a buffer. The main advantage of output buffering is that no head of line blocking problem occurs. This is easily seen since a message coming in on a channel is immediately routed to the appropriate output buffer. When a message is placed in an output buffer, only messages going to the same output channel are in the buffer. If two messages are in an output buffer, only the first message can be routed since the router only sees the top of the queue. However, since both messages need the same output channel, neither message can be blocked unnecessarily. They can only be blocked by congestion. There are a number of disadvantages with output buffering. First, many different kinds of selection policies cannot be used to improve performance because, unlike input buffering, many selection policies need information from the output buffers on the previous node (chip). Since, off-chip wires are a high commodity, selection policies are limited. For example, looking at buffer length is too complicated for a selection policy because it requires much off-chip information. The early commitment problem is another disadvantage for output buffering when several choices are available for routing (such as in adaptive routing). This problem can occur when a message is assigned to one of two possible output buffers each of which already have other messages in them. If the message is assigned to the queue that takes longer to empty, it could have routed faster if it had been assigned to the other queue. Finally, implementing output buffers requires more hardware than that required for input buffers. This results because a message

must be routed and buffered in the same cycle it enters a node. Note that this difference diminishes in a pipelined router [DYN97, Lag94].

2.1.3 Routing Algorithm

Routing algorithms determine the path used by each message or packet. There are countless routing algorithms all of which fall into three main categories: deterministic, partially adaptive, and adaptive.

1. **Deterministic Routing**

   In deterministic routing, a message always uses the same path between each source and destination node and this path is determined as a function of the destination address. One of the most popular deterministic routing algorithms is called dimension-order routing. In this type of routing, a message is routed along decreasing dimensions with a dimension decrease occurring only when zero hops remain in all higher dimensions. Virtual channels are included in the router to avoid deadlock [Dal92].

   Deterministic routing algorithms are very common in commercial multicomputers, such as the Intel Paragon [Int91], Cray T3D [KS93], and NCUBE-2/3 [NCU90]. This type of routing is also prevalent in many experimental multicomputers, such as the Stanford DASH [LLG+92] and MIT J-Machine [NWD93]. This routing type is so popular because of its simplicity both in concept and in implementation. However, deterministic routing algorithms can suffer from congestion since only a small subset of all possible paths between a source and destination are used.

2. **Partially Adaptive Routing**

   In this type of routing, the adaptivity of a message is increased with little resource addition. An example of this routing type is called planar-adaptive routing and is proposed for n-dimensional meshes and hypercubes [CK92]. In this type of routing,

a packet is only routed in two dimensions at a time. In order to prevent deadlock, an ordering is imposed. For example, a message is first routed in plane A_0, then plane A_1, and so on. Since at each clock cycle, adaptivity is limited to two dimensions, resources are minimized while gaining adaptivity. The Turn Model proposed by Chien and Ni [GN92] is another example of a partially adaptive routing algorithm. In this algorithm deadlock freedom is accomplished by preventing turns and does not require virtual channels. Once again adaptivity is gained with a minimal increase in resources.

3. **Adaptive Routing**

In adaptive routing, messages are not restricted to a single path when traveling from source to destination. Moreover, the choice of path can be made dynamically in response to current network conditions. Such schemes are more flexible, can minimize unnecessary waiting, and can provide fault-tolerance. Several studies have demonstrated that adaptive routing can achieve a lower latency, for the same load, than deterministic and partially adaptive routing when measured by a constant clock cycle for both routers [GN92, Lag94].

The Chaos router [KS91] is an example of an adaptive router. It has buffers associated with each input and output port of a node, as well as centralized buffers which are used for deadlock prevention. This router performs particularly well when hot-spots are present due to its ability to retain adaptivity as well as to efficiently use network status information.

A class of adaptive routing algorithms is know as the hop algorithm class [Gop85]. In this type of algorithm, deadlock is avoided by splitting buffers into several classes. The packets can then only be moved from one buffer to another in such a way that buffer class is never decremented. This algorithm can be used for any topology.

However, the number of buffers required is very high and depends on the network size.

Another interesting example of adaptive routing is the hot potato algorithm [Bar64]. In this algorithm, a packet is routed as fast as possible by allocating it to the shortest output queue. Because the packet is routed without regard to where the channel leads, the choice may not be optimal. However, this algorithm is very fast, allows for flexibility, and is fault tolerant.

Many researchers have tried avoiding deadlock in adaptive routers using virtual channels. Unfortunately, many of the early models required a large number of virtual channels. In [LH91], the concept of virtual channels was extended to that of virtual networks. In this strategy, a single network was logically split into several layers of networks, and messages moved through succeeding layers as they traversed the network. Adaptivity was allowed within a layer, and thus a message could choose among a variety of paths. The drawback of this system, however, is that the number of virtual channels required by the model grows exponentially with the size of the network. Even if these are not separate physical channels, the buffer space and control logic required to manage such a system quickly makes the model infeasible.

The adaptive routing scheme considered in this proposal is known as Duato's algorithm or the *-channels algorithm [Dua91, Dua93, BGPS92]. In this algorithm, adaptive routing is obtained by using adaptive virtual channels along with dimension-order routing. A message is routed on any adaptive channel until it is blocked. Once blocked, a message is routed using dimension-order routing if possible. Note that a message may return to the adaptive channels in the following routing decisions if the adaptive channels are available. This algorithm has been proven to be deadlock-free as long as the following routing restrictions are imposed: when the message size is greater than the buffer size (i.e. size of the the virtual

channel), deadlock is prevented by allowing the head flit of a message to advance to the next node only if the receiving queue at that node is empty. If the message size is less than the buffer size, then deadlock is prevented by allowing a message to advance only as long as the whole message fits in the receiving queue at that node. This algorithm requires a minimum of three virtual channels per dimension per node for each physical unidirectional channel. Therefore, the number of virtual channels grows linearly with the size of the network.

2.2 Router Complexity

Much of the research on adaptive routing has been done in the context of developing deadlock-free routing algorithms. Therefore, much of the emphasis was on the algorithm design and implementation of the router algorithm was secondary. Since increases in router complexity also increases router delay, the effects that this complexity has on clock cycle time needs to be understood. Some of the earliest work in this area involved deterministic routers. In the torus routing chip [DS86], deadlock-free wormhole routing techniques were used along with an implementation based on a full crossbar architecture. Later implementations greatly increased router performance by taking advantage of dimension-order routing's uniformity. This was accomplished by subdividing the full crossbar into a set of smaller crossbars, as well as subdividing all other regular logic [DS87a, Fla87, SS93].

Further investigation involved adaptive routers. Comparisons of adaptive and deterministic router implementations for wormhole routing are described in [AC95, Chi93] and [DL94]. In [AC95], delay equations for the routers are derived. These models account for both the logic complexity of the routers as well as the size of the crossbar as determined by the number of virtual channels that are multiplexed on one physical channel. These models were modified here to account for the varying buffer space used in virtual cut-through routing. The parameters of these models are:

Symbol	Variable (delay)
T_{AD}	Address decoding
T_{ARB}	Routing arbitration
T_{CB}	Crossbar
T_{FC}	Flow control
T_{SEL}	Header selection
T_{VC}	Virtual channel controller
P	Max. no. of IP or OP ports in crossbar
F	Degrees of freedom (OP choices of a message)
C	No. of virtual channels
B	Buffer size (in number of flits)

The address decoding term (T_{AD}) includes the time for examining the packet header and creating new packet headers for all possible routes. The time required for selecting among all possible routes is included in the routing arbitration delay (T_{ARB}). The crossbar delay (T_{CB}) is the time necessary for data to go through the switch's crossbar and is usually implemented with a tree of gates. The flow control delay (T_{FC}) includes the time for flow control between routers so that buffers do not overflow. T_{SEL} is the time for selecting the appropriate header. Finally, the virtual channel controller delay (T_{VC}) includes the time required for multiplexing virtual channels onto physical channels.

Delay equations for the routers are derived in [Chi93], using the above parameters. The constants in these equations were obtained using router designs along with gate-level timing estimates based on a 0.8 micron CMOS gate array process. The models were modified here to account for the varying buffer space used in virtual cut-through switching. Three main operations are used in all of the routers which contribute to the following three delays:

- T_r: Time required to route a message
- T_s: Time necessary to transfer a flit to the corresponding output channel
- T_c: Time required to transfer a flit across a PC

The equations are:

$$T_r = T_{AD} + T_{ARB} + T_{SEL}$$

$$T_r = 2.7 + 0.6 + 0.6 * \log_2 F + 1.4 + 0.6 * \log_2 F$$

$$T_s = T_{FC} + T_{CB} + T_{Latch}$$

$$T_s = 0.8 + 0.6 * \log_2 B + 0.4 + 0.6 * \log_2 P + 0.8$$

$$T_c = 4.9 + T_{VC}$$

$$T_c = 4.9 + 1.24 + 0.6 * \log_2 C$$

To decrease the overall router delay, it is assumed that all three operations are overlapped through pipelining as described in [DL94], and therefore the clock period is determined by the longest delay:

$$T_{ccperiod} = Max(T_r, T_s, T_c)$$

The results in [AC95] show that 15% to 60% improvement is required for f-flat routers with similar number of virtual channels and under wormhole routing. However, the comparison in [AC95, Chi93] does not account for the reduced queuing delay in adaptive routing. In [DL94] the reduction in queuing delay for wormhole routing is taken into account and the comparison is based on a constant total buffer area.

2.3 Hybrid Techniques

Hybrid techniques attempt to combine the beneficial aspects of basic approaches while reducing the negative aspects. They are commonly used to enhance performance or to optimize performance metrics other than traditional latency and throughput (e.g. fault tolerance and reliability).

One such example is the architectural support for the reduction of communication overhead as described in [DYD97]. This scheme exploits the communication locality in message passing programs to distinguish between cacheable and non-cacheable virtual channels. Cacheable virtual channels are retained for multiple messages thereby allowing an overlap

of communication and computation and eliminating the overhead of multiple message set-up. This mechanism is a hybrid scheme combining circuit and wormhole switching. The implementation of a router supporting this scheme is described in [DLSY96]. Its routing properties are discussed in [DLY97].

Another example includes buffered wormhole switching (BWS). In this technique, as long as the path is free, the flow control is at the flit level and the message continues just as in wormhole switching. However, when messages block, flow control within the switch is organized into chunks where each chunk is composed of eight flits. These chunks are then stored within the switch in a dynamically allocated central storage. This storage is organized as a linked list for each output, where each element (chunk) of the list is part of a message to be transmitted on that output port. BWS differs from wormhole switching because flits are not buffered in place, but are aggregated and stored in local memory within the switch. Similar to packet and cut-through switching, if a message is small enough and space is available in the central queue, the input port is released for use by another message even though the message packet remains blocked.

Pipelined circuit switching (PCS) is another hybrid technique. It combines aspects of both circuit and wormhole switching. In this type of switching, data flits do not immediately follow the header flit into the network as in wormhole switching. Instead, a header flit is used to set up a path similar to circuit switching. When the header flit reaches the destination node, an acknowledgment flit is transmitted back to the source node. Only then are data flits pipelined over the path as in wormhole switching. Note that, unlike circuit switching, the paths use virtual channels instead of physical ones. By using the header flit to set up a path without data flits following it, increased routing flexibility is obtained. For example, instead of blocking on a faulty output channel, the header may "backtrack" to the preceding router and release the previously reserved channel. A new output channel may then be attempted.

In addition, reserving channels does not by itself lead to loss of physical channel bandwidth because virtual (and not physical) channels are used. This reduces excessive blocking of other messages during path setup.

2.4 Examples of Machine Implementations

From a commercial perspective, a computer that maximizes the performance/cost ratio is optimal. However, performance and cost vary greatly depending on many factors such as topology, flow control, routing algorithm and on the implementations of each. In addition, the main traffic patterns and message sizes of applications used on a multi-computer are just a few of the specific parameters that need to be known before designing begins. Despite the complex process, many designers have been very successful in implementing the theoretical design parameters into actual multi-computers.

In particular deterministic routing algorithms are very common in many multicomputers because of their simplicity. Some of the machines which use this type of routing are the Intel Paragon [Int91], Cray T3D [KS93], and N-CUBE/2 [NCU90], MIT J-Machine [Dal89], and Stanford DASH [LLG+92]. Under random traffic and symmetric topologies, deterministic routing algorithms perform quite well. However, as hot spot traffic becomes more prevalent and topologies become more asymmetric, adaptive routing's flexibility gives better performance. The Cray T3E router [ST96, Sco96], the Chaos Router [KS91], and the Reliable Router [DDH+94] all use adaptive routing.

Topology is another important design parameter. Some of the most popular topologies are the mesh and low-dimensional k-ary n-cubes. This is because these types of topologies can be feasibly implemented due to their low wiring densities, small chip pin out requirements, and short wire lengths. Some examples of mesh-based machines are the Intel Paragon [Int91], Ametek 2010 [SAF+88], MIT Alewife [KA93], J-Machine [Dal89], Stanford DASH [LLG+92], and Chaos router [KS91]. Some of the multi-computers that use k-ary n-cubes

are the iPSC series of machines from Intel [MS86], the Cray T3 series D and E machines [KS93, ST96, Sco96], the Tera Computer [ACC+90], N-CUBE/2 [NCU90], and Thinking Machines' CM-1 and CM-2 [Hil85, CM287]. Many other topologies exist. The fat-tree topology is such an example and is found in the CM-5 [L+92] and the *T multiprocessor [PBGB93].

Numerous switching techniques exist. One of the most popular is wormhole switching. Machines using this type of switching include the Intel Paragon [Int91], N-CUBE/2 [NCU90], Ametek 2010 [SAF+88], J-Machine [Dal89] and Stanford DASH [LLG+92]. Virtual cut-through switching is also popular and is implemented in the Chaos router [KS91], Arctic router [Bou94], and Reliable Router [DDH+94].

2.5 Multicast Communication

Multicast communication is a common operation in parallel computing and efficient implementation of multicast is critical to the performance of multiprocessors. Software-based approaches for implementing multicast can result in high message latencies. Hardware-based schemes can greatly improve performance but deadlock freedom is much more difficult in multicast communication, resulting in more involved routing algorithms and higher startup delays. Hardware tree-based algorithms do not require these high startup delays but do suffer from high probabilities of message blocking leading to poor performance.

2.5.1 Multicast Communication Services

There are four main types of multicast communication services. They are multiple one-to-one, one-to-all, all-to-one, and all-to-all. Multiple one-to-one communication simply involves each process sending and receiving one message.

In one-to-all communication, one process is the sender (also called the root) and all remaining processes in the group are receivers. Two specific examples in this category are

broadcast and *scatter* (also called *personalized broadcast*). In a *broadcast*, the same message is sent from the sender to all receivers. In a *scatter*, the sender gives different messages to different receivers.

In all-to-one communication, all processes in a group are the senders and only one process in the group is the receiver. Two examples of this are *reduce* (also known as *personalized combining* or *global* combining) and *gather*. In a *reduce* operation, the messages from different senders are combined into a single message for the receiver. Some combining operators are addition, multiplication, maximum, minimum, and logical operators (e.g. AND, OR, exclusive OR). In a *gather* operation, different senders send different messages that are then concatenated together for the receiver.

In all-to-all communication, each process in the group sends one message to every other process in the group. Examples include *all-broadcast* (also known as *gossiping* or *total-exchange*) and *all-scatter* (also known as *personalized all-to-all broadcast*, *index*, or *complete-exchange*). In *all-broadcast*, each process performs its own broadcast. Therefore, all processes have the same set of received messages. In *all-scatter*, each process performs its own scatter. Therefore, all processes have different received messages.

In addition to the above multicast communication services, there exists some frequently used services that are a combination of the above services. These are called *convenient* or *composite* multicast communication services. They include *all combining* (also known as *reduce and spread*), *barrier synchronization*, and *scan*. In an *all combining* operation, the result of a *reduce* operation is available to all processes and the result may be broadcast to all processes. A *barrier synchronization* operation involves a *reduce* operation followed by a *broadcast*. In this type of synchronization, all processes in a group must arrive before any of the processes in the group are allowed to continue. A *scan* operation involves many *reduce*

operations and performs a *parallel prefix* with respect to a commutative and associative combining operator on messages in a group.

2.5.2 Encoding Schemes for Multidestination Messages

Unicast communication only requires one destination address in a message. However, in multicast communication, many destination addresses are necessary. Therefore, numerous methods can be used to account for these addresses in a message, each having their own advantages and disadvantages.

In a multidestination message, the addresses for the destination nodes are carried in the header flits. The five most common encoding schemes for these address flits are *all-destination*, *bit string*, *multiple region broadcast*, *multiple region stride*, and *multiple region bit-string*. The *all-destination* encoding scheme is the simplest of all the schemes. In this scheme, each destination address is carried by a header flit. To distinguish between header and data flits, an *end-of-header* (EOH) flit can be placed at the end of all the header flits or a bit in each flit can be used to distinguish between header and data flits. The main advantages of the *all-destination* encoding scheme are that the same routing hardware can be used for unicast messages as well as multidestination messages and that header flits can be routed as soon as they arrive. The main disadvantage of this method is its inefficiency when a large number of destination addresses are used.

In *bit string* encoding, this disadvantage of inefficient encoding is greatly reduced because each destination address is encoded as a bit string, where each bit corresponds to a destination ranged between two nodes. Because this string length is predefined, the EOH flit is not required. Although this scheme is very space efficient, its main disadvantage is that the total bit string has to be buffered by the router before the routing decision can be made and a new output bit string needs to be generated.

The last three encoding schemes use ranges of addresses or regions in order to reduce the header length. The *multiple region broadcast* encoding scheme specifies a number of regions in the header flit where each region is composed of the beginning and ending addresses in the region. The message is broadcast to all addresses in the specified range.

In *multiple region stride*, the concept of a stride is added to the region. This can be used in cases where a message may be sent to a set of destination addresses that have a constant distance between two adjacent addresses.

In *multiple region bit-string*, each region is specified by a bit string in addition to the beginning and ending addresses. This is useful when there exists irregularly distributed destination addresses which can still be grouped into regions.

The main disadvantage of the three encoding schemes based on regions is that the address decoding logic in the router becomes more complex. In addition, since several flits are usually necessary to encode each region, these flits all need to arrive at the router before the routing operation can start.

2.5.3 Software Multicast

Most current systems only support one-to-one communication. Therefore, in software implementations of multicast, one or more unicast messages must be sent. The simplest implementation of multicast using unicast messages is to have one unicast message sent for every destination address in the multicast message. Here this method is referred to as Software Multicast. However, this method can be inefficient, especially when many destinations share common paths and message lengths are long.

To increase the efficiency of unicast-based multicast, the multicast communication can be implemented using multicast tree algorithms. In these algorithms, the source node sends each message to a subset of the destinations, each of which forwards the message to one or more destinations. The two important parameters in these types of algorithms

are to reduce the tree height and to minimize the contention among multiple submulticast messages [Ni95].

The Chain algorithm is an example of a unicast-based multicast tree algorithm in which no contention exists between the multidestination messages belonging to the same multicast [MXEN94] when used with dimension-order routing. The Chain algorithm uses dimension-ordered chains. In dimension-ordered chains, a set of destination addresses is represented by the number of hops required in each dimension from the source node to the destination node along a minimal path. These destination addresses are then ordered so that the coordinates for each node are sorted according to increasing or decreasing order. For example for a k-ary n-cube network, the set $(0,0)(1,3)(2,0)(0,3)(0,2)$ is sorted into the following increasing order: $(0,0)(0,2)(0,3)(1,3)(2,0)$. Note that the $(0,0)$ coordinate is always included in the dimension-ordered chain and always represents the source node. The source node can always be represented as $(0,0)$ since a k-ary n-cube network is symmetric.

These destination addresses in the dimension-ordered chain are then placed into several different multidestination messages using a message preparation scheme. In this message preparation phase, the source node first sends a multidestination message to the destination node halfway across the chain from the source node. This message also includes the destination node addresses for all those nodes more than halfway across the chain. However, these destination addresses are not yet used to route the message. Only the destination node halfway across the chain is used for routing the message. The source then sends out a multidestination message to the destination node one quarter of the distance across the chain. This message also includes all those destinations greater than one quarter and less then one half of the distance across the chain. This message preparation scheme continues in this manner until all destination nodes for this multicast message at this source node have been included in one of the submulticast messages.

When a destination node receives one of the submulticast messages sent by the source node and more destination nodes exist in this message, the above message preparation scheme is executed at this current node just as it was executed at the source node. However, this time, the current node is used as the source node for the remaining destinations in the message. The newly composed submulticast message is then reinjected into the network. This process continues until all destinations have been reached. Note that if a message is received at a destination node and no more destinations exist in the message, the message is simply absorbed by this last destination node.

In the above dimension-ordered chain example, the message preparation scheme executed at the source node results in the source node sequentially sending out the following multidestination messages: message $0 = (0,3)(1,3)(2,0)$ and message $1 = (0,2)$. When message 0 reaches the $(0,3)$ destination node, the message preparation scheme is once again performed on the current destination list. This results in the node sending out a new message composed of $(1,3)(2,0)$. Note that the second message sent from the source node (message 1) only has one destination address and therefore when this message reaches its destination, it is simply absorbed by the node.

Other software multicast tree algorithms have been developed for meshes and hypercubes [MXEN94], bidirectional MINs and fat trees [XGN94], and unidirectional MINs [CN95]. However, only the Software Multicast and Chain algorithms are described here because they are the most common software implementations used for comparison purposes.

2.5.4 Hardware Multicast

Unfortunately, multicast latencies are quite high for software-based approaches because of the high latencies for sending and receiving multiple unicast messages [Ni95]. Therefore, hardware multicast support is currently being considered to improve performance. Several machines already have some hardware support for multicast. The nCUBE-2 is a wormhole-

switched hypercube which supports broadcast within each subcube [NCU90]. However, deadlock is possible if multiple multicasts exist [Ni95]. The NEC Cenju-3 supports broadcast within each continuous region, but deadlock is once again possible if multiple multicasts exist. Finally, the Thinking Machines Corporation (TMC) CM-5 supports one multicast at a time via the control network.

The two main types of hardware multicast support are path-based and tree-based. Each use multidestination messages which are messages that have more than one header flit. The main difference between these two types of multicast support is how the routing of the header flits in a multidestination message is performed. Each type is described below.

2.5.4.1 Path-based Routing

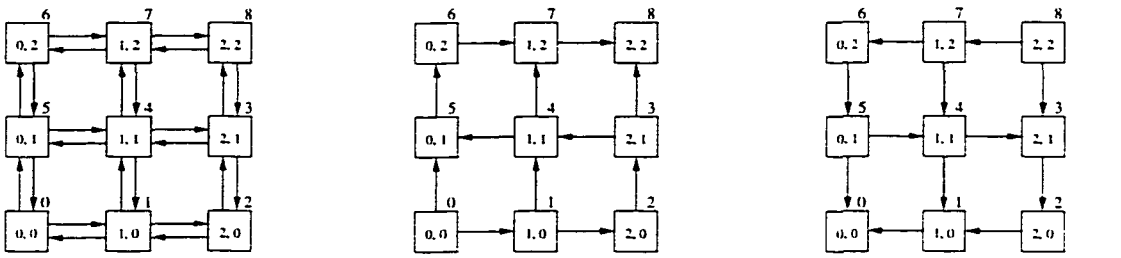
Path-based multicast routing is when a multidestination message is routed through the network using only the first header flit in the message. Once this destination is reached, the first header flit is removed by the router and the next header flit is used to continue routing the message to the next destination. The data flits are forwarded to both the destination node as well as to the next input queue required for the next header flit. This continues until all destinations are reached and the message is completely consumed by the node. To reduce the path length of a multicast message, the destination node set is divided into disjoint subsets. Each disjoint destination subset is then composed into separate submulticast messages and sent along separate multicast paths. By appropriately ordering the destinations within each set or subset at the source node, the messages can be routed more efficiently.

The main advantage of path-based routing is that the probability of a message blocking is low. Assume that the probability that a message is blocked in each channel is p. Then the probability that a message is blocked under path-based routing is also p because each

multicast message only requests one channel at a time. A path-based router is also relatively simple because message replication at each intermediate node is not necessary.

One of the main disadvantages of path-based routing is that an ordered list of destination addresses for each copy of a message is required before the message is injected into the network. In some cases, this destination ordering can be computed at compile time. However, when messages are dynamically created by hardware, this cannot be done. Therefore, the time to order the lists should be small enough so that the lower blocking probability of path-based routing (and therefore lower message latency) offsets the time required to order the messages. Another disadvantage of path-based routing is that the subpath between the source node and one of the destination nodes in a multicast path is frequently not the shortest path.

Hamiltonian Path Algorithms: Two main types of path-based algorithms are those based on Hamiltonian paths and those based on implementations using the Base Routing Conformed Path (BRCP) model. A Hamiltonian path is a path in which every node is visited exactly once. Algorithms based on Hamiltonian paths include the Dual-Path, Multipath, and extensions of these two algorithms. There exists many Hamiltonian paths in each type of network. The path which best accommodates the routing algorithm should be chosen. After an Hamiltonian path for a network with N nodes is chosen, each node u is assigned a label ($l(u)$) based on the node's position along the Hamiltonian path. The first node in the path is labeled 0, while the last node is labeled $N-1$. Figure 2.1a shows an example of two Hamiltonian paths (the high and low-channel paths) in a 2-D mesh. Figures 2.1b and 2.1c explicitly show the high and low channels. The high-channel network is used to send a message to a higher labeled node and the low-channel network is used to send a message to a lower labeled node. The inclusion of these two channel paths allow for deadlock-free multicast routing in the 2D mesh.



(a) Physical Network (b) High-Channel Network (c) Low-Channel Network

Figure 2.1: Hamiltonian path labeling for a 3x3 mesh

The first routing algorithm using the above Hamiltonian paths is called the Dual-Path routing algorithm [LN91]. In this algorithm, the destination node set is divided into two destination node subsets, D_H and D_L. The subset D_H contains all the destination nodes that have a higher label ($l(u)$) than the source node and the subset D_L contains all those nodes that have a lower label than the source node. Within each subset, the nodes are sorted in increasing order for D_H and decreasing order for D_L. Each subset is then placed into its own multidestination message which is then sent along the appropriate Hamiltonian path in either increasing or decreasing order. Once the first destination node in the submulticast message is reached, it is absorbed by the message as well as forwarded to the next destination node. If no other destination nodes exist in the message, the message is completely absorbed by the node. Using this algorithm, a submulticast message is always routed along a shortest path for a given source and destination node. However, when multiple destinations are routed using one multidestination message, the paths between the source node and one or more of the destination nodes is usually not the shortest path.

Because the Dual-Path routing algorithm only allows two multicast paths in the network (the high and low channels), the average path and number of channels used by a message can be large. The Multi-Path routing algorithm increases the number of multi-paths allowed. For example, in a 2D mesh with an outgoing degree of four, four paths can be used to deliver a message. The only difference between the two algorithms is that the

D_H and D_L subsets in the Dual-Path algorithm are each divided into two more subsets in the Multi-Path algorithm. The D_H subset is divided into those nodes with x coordinates greater than or equal to the source node's and those nodes that are the remaining nodes in D_H. The D_L subset is divided in a similar manner. This algorithm does reduce the path length of a multicast message and at low and medium traffic it does perform better than the Dual-Path algorithm. However, at high utilization and a large number of destinations, a source node will most likely be sending messages out on all its outgoing channels. These messages will block any other multicast or unicast message that needs to be routed through this node and this node becomes a hot spot. Under these conditions, the Dual-Path routing algorithm performs better.

Two more extensions of the above routing algorithms using Hamiltonian paths allow for adaptive multicast path-based routing. The Label-Based Dual-Path (LD) algorithm [LME93] is the same as the Dual-Path algorithm except that in addition to the allowable minimum paths between successive destinations, non-minimum paths are also acceptable as long as nodes are crossed in strictly increasing or decreasing label order.

The other extension allows messages to use alternative minimal paths between successive destinations [Dua93, Dua95]. The regular minimal deterministic channels are used in the same way as in the Dual-Path algorithm. However, at least one additional adaptive channel is added per dimension. These adaptive channels can be used to route a message to a destination node with a higher or lower label value than the source node when the intermediate nodes used have label values less than or greater than the next destination node, respectfully. In [LD94], the algorithm is expanded to include the 3D mesh.

Base Routing Conformed Path (BRCP) Algorithms: Hamiltonian path-based algorithms have been shown to perform better than unicast-based multicast routing. However, these algorithms do not incorporate the currently wide-spread unicast routing algo-

rithms such as e-cube and adaptive routing. The second main type of path-based routing algorithms, the Base Routing Conformed Path (BRCP) model, is compatible with existing unicast routing algorithms. This algorithm allows a multidestination message to be sent through any valid path in the network conforming to the base routing scheme. For example, when XY routing is used for the base routing scheme on a 2-D mesh, a valid path is any row, column, or row-column. Note that a column-row path is not allowed just as it is not allowed in XY routing. The BRCP model is formally defined as:

Definition: *A multidestination message from a source s with an ordered destination list $d_1, d_2, ..., d_{n-1}, d_n$ in a network supporting the routing function R conforms to the base routing if and only if the destination set $d_1, d_2, ..., d_n - 1$ can be covered as intermediate nodes on one of the possible paths from s to d_n under the routing constraint R_1, where R_1 is a connected routing subfunction of R that has no cycles in its extended channel dependency graph [DYN97].*

The Column-Path Algorithm is an example of an algorithm using the above BRCP model to implement multicast. In this algorithm, the destinations in a multicast message are placed into submulticast messages according to the column the destination is in. For example, in a 2D mesh, at most $2k$ submulticast messages can be sent per multicast message. one submulticast message sent per column.

The Hierarchical Leader-Based Scheme (HL) is similar to the Column-Path algorithm but more general. This algorithm also uses the BRCP model to implement multicast. However, in this routing scheme, a multicast destination set is grouped hierarchically so that the minimum number of multidestination messages (submulticast messages) are required to reach all the destinations. It relies on a two-phase process. In the first phase, unicast messages are sent from the source node to a set of L_m destination nodes. In the second phase, multidestination messages are sent from these L_m nodes (called leader nodes) to the next set of destination nodes, L_{m-1}. Some of the nodes in L_{m-1} (also called leader nodes) send multidestination messages to the next lower level L_{m-2}. This continues until all destination nodes have been reached. Note that the groups of destination nodes are

obtained using the base routing algorithm. For example, if XY routing is used as the base routing scheme in a 2D mesh, then the grouping is done such that the destination nodes lie on a row, column, or row-column. The advantage of this model is that the grouping becomes more efficient as the degree of multicasting increases.

The HL scheme can, however, create much congestion when numerous multicast messages are sent. This problem is reduced in the Quadrant-Based Hierarchical Leader (SQHL) model by making the leader nodes for different multicast messages as unique as possible. This is done by using the same scheme as the HL model described above with the exception that the nodes are grouped based on the position of the source node in the system.

The Source-Centered Hierarchical Leader (SCHL) model also improves upon the HL scheme. In this model, groups for the HL scheme are each divided into two more groups based on the coordinates of the source node.

2.5.4.2 Tree-based Routing

In tree-based multicast routing a multidestination message is routed through the network using *all* the header flits in a message at each node. The message is routed along a common path among all header flits in the message as far as possible. The message is then replicated and each copy is forwarded on a different channel headed for a unique set of destination nodes. This branching continues as necessary until all destination nodes have been reached.

The main advantage of tree-based routing is that no ordering of the destinations is required before the message is injected into the network. In addition, the shortest path between the source node and all destination nodes is taken.

The main disadvantage of tree-based routing is that the probability of blocking is much greater than that for path-based routing because all branches must be available for the whole multidestination message to continue. Assuming that p is the probability that a

message is blocked in each channel and that k number of channels need to be simultaneously available, the probability that a multidestination message is blocked in tree-based routing is $1 - (1 - p)^k$.

Tree-based routing also requires more complex routers. A number of modifications to the unicast routers used in the previous chapters are required for a tree-based multicast router. Several status registers are now necessary to hold output channel reservations for a message arriving on an input channel. In addition, flow control is more complex because acknowledgment signals from all the output channels reserved by a message must be AND-ed together since flits can only be forwarded when all neighboring channels receiving them are available. The crossbar also needs to be slightly modified. Since data flits must be forwarded simultaneously, the same input may need to be selected for several outputs. In the routers simulated here, a crossbar implementation using a set of multiplexers (one for each output) is assumed. Therefore, this modification is not difficult. Delivery channels at each router must also be able to automatically remove flits containing destination addresses.

Double-Channel XY Algorithm: The Double-Channel XY multicast routing algorithm is a deadlock-free tree-based routing algorithm for a mesh network [LMN91]. In this algorithm, deadlock freedom is obtained by doubling each channel in the mesh. For a 2D mesh, this results in four subnetworks, $N_{+X,+Y}$, $N_{+X,-Y}$, $N_{-X,+Y}$, and $N_{-X,-Y}$. For each multicast message, the destination node set, D, is also divided into at most four subsets according to the relative positions of the destination nodes to the source node ($D_{+X,+Y}$, $D_{+X,-Y}$, $D_{-X,+Y}$, and $D_{-X,-Y}$). Each subset corresponds to submulticasts that are sent from the source node onto their appropriate subnetworks. The major disadvantage of this multicast routing algorithm is that double channels are required for deadlock freedom and the number of channels between every pair of nodes grows exponentially with the number of mesh dimensions. This algorithm has also been shown to perform worse than the path-

based algorithms for wormhole switching. The reason is that when one branch of a tree blocks, the remaining branches cannot advance.

Tree-based Multicast With Branch Pruning Algorithm: In [MDT96], a tree-based routing algorithm called Tree-Based Multicast with Branch Pruning is implemented for wormhole switching and only requires that the unicast routing function be deadlock-free. In this algorithm, in addition to the normal fixed-size input buffers used for flit pipelining in wormhole switching, small fixed-size data buffers capable of holding all the data flits in a message are placed at each input channel. These data buffers only hold the data flits. The address flits use the regular input buffers and still move throughout the network as in wormhole switching.

Because the data flit buffers are small, the routers are compact and fast. However, these small buffers also mean that the data size must be small since all data flits must be able to fit into the data buffers. In [MDT96], only one data flit is used per buffer. The message format is also slightly different from the regular message format in which all data flits follow all header flits. In the new format, the first flit in the message is a header flit, followed by all data flits and then all remaining header flits. After the first header flit is routed, the following data flits are stored in the data buffer at the current node as well as forwarded to the queue that the first header flit was routed to. Any remaining header flits are either routed to the same queue as the first header flit (and are placed behind the data flits) or are routed to a different queue. If the header flit is routed to a different queue, the data flits stored in the data buffer must also be forwarded behind this header flit.

If an address flit blocks for a predetermined amount of time, the header flits already routed are allowed to continue along the network since they already contain all the data flits. This is called pruning and reduces message blocking probability because it frees up queues for other messages. Although pruning can be done on all of the branches of the tree already

routed, it is only done if the pruned branches can forward flits once they are cut. After a tree has been completely pruned, the multicast message becomes a set of multiple unicast messages. Since the unicast routing function is deadlock-free, the multicast algorithm is also deadlock-free. This algorithm is best used for invalidation or update commands in distributed shared memories (DSMs) with coherent caches since small data buffers are required.

Resumable Multicast and Restricted Branch Multicast: In [BSD89, BND87], deadlock detection and recovery for multicast communication is used with virtual cut-through switching. The deadlock detection and recovery is implemented with timeouts and reinjection delays. The virtual cut-through switching is implemented using a common buffer pool at each node. Not only is this buffer shared among all channels, but it is also located in the local processing node (not in the router itself). Because all buffers at each channel are small, wormhole switching is used until a message becomes a potential deadlock (i.e. a message is blocked for a predetermined amount of time). Only then is the message stored in the common buffer area. This temporary buffering at intermediate nodes avoids deadlock instead of using virtual channels as in [DS87b]. Note that if a message deadlocks, all header flits (including all those that have already been routed at the current node and at any of the downstream nodes) are aborted. Aborting a message is allowed because whenever a multicast message is split at a node (whether deadlock occurs or not) the entire message is always copied to the local processing node using a channel called the split channel. When an abort does happen, the message is already stored at the local node and is ready to be reinjected into the network after a predetermined amount of time.

Two tree-based multicast protocols, resumable multicast (RM) and restricted-branch multicast (RBM), are described in [BSD89, BND87] using the above basic algorithm. In both cases each input and output channel are able to store one target address.

In the Resumable Multicast (RM) protocol, a message is routed along a common path as far as possible. When a packet must be split during the routing of a multicast message, a copy of the message is sent to the current node and is stored in the common buffer pool. If the current multicast is then aborted due to a potential deadlock, the message is retransmitted from this node using this copy. Note that a potential deadlock occurs when one or more branches of the tree have been blocked for more than a given threshold. When a deadlock does occur, all branches at the current node are aborted even those branches that have already been successfully routed. If space is not available or if access to the current node is blocked, the packet is not split. Instead, it is "passed through" the node toward the first destination node in the message. Note that the total storage among all nodes for blocked messages must be large enough to hold all necessary messages. Otherwise deadlock could result. This algorithm is shown to perform poorly.

The Restricted Branch Multicast (RBM) algorithm is the same as the RM except that the number of fanouts at each node is limited to at most two branches, one of which includes either the split or sink channel. The split channel is connected to the local node just as the sink channel is. However, the split channel is only used for multicast messages that need to be split. If the split channel is not available, the message may not be split. It may be buffered at the local node using the sink channel or it may pass through the node without servicing any local targets. If the message is buffered at the local node, any local destination address will be removed and the remaining message will be re-transmitted later. Note that a multicast message that is the leaf of the multicast is identical to a unicast packet and therefore uses the sink channel to connect to the local node. If this sink channel is busy, the channel will eventually become free because no splitting message can use it and because all packets are of finite size. Therefore deadlock is avoided. Note that two local ports are sufficient to avoid deadlock because only two-way splits are allowed. If higher fanouts were

used, additional ports would be required. By reducing the fanout of a message to at most two, the probability of blocking is greatly reduced.

The two tree-based routing algorithms (RM and RBM) in [BSD89, BND87] are compared against the multiple unicast (MU) routing algorithm. The RM algorithm is a fully tree-based routing algorithm while the RBM algorithm is very similar to path-based routing because the branch fanouts at each node are limited to two (one of which includes the channel to the local node). The RBM algorithm is shown to perform the best among all the schemes, while the RM algorithm performs the worst. The poor performance of the RM algorithm due to the high probability of a message blocking supports the idea that tree-based algorithms perform poorly.

Chapter 3

PERFORMANCE ISSUES IN ADAPTIVE ROUTER ARCHITECTURES

The performance issues in adaptive router architectures addressed here include input/output selection policies, effects of virtual channel quantity and varying buffer sizes, resource utilization with respect to physical wire bandwidth, and performance effects of router complexity and delay. All these issues involve both advantages and disadvantages and none are optimal under all conditions. Traffic type, message size, network topology, and routing algorithm are just a few of the parameters that affect the performance of each of the above issues. In this chapter, the affects of such parameters on the above issues are shown and conclusions under the given parameters are determined.

3.1 Input/Output Selection Policies

Determining an efficient and practical input/output selection policy can enhance or degrade router performance. The effects of such policies under specific parameters are examined here. Numerous selection policies exist in the literature today, many of which have been previously tested under various types of routing (e.g. packet switching, wormhole routing). While, the *input* selection policies used here are well-known policies, a new *output* selection policy is implemented. A total of four input/output selection policies are simulated. They are as follows:

- *Optimized:* an optimized input/output selection policy

- *Random:* a random input/random output selection policy
- *Partially Random:* a randomly selected input selection/greatest number of hops output selection policy
- *Incremental:* an increasing order input selection policy/greatest number of hops output selection policy
- *Round Robin:* a round robin input selection/greatest number of hops output selection policy
- *Prioritized Round Robin*: a prioritized round robin input selection/greatest number of hops output selection policy

To determine the most efficient and practical policy among all six input/output policies, simulations were performed using varying message lengths, buffer sizes, and number of virtual channels. Results show that a well-known *input* selection policy/new *output* selection policy performs the best overall.

The first input/output selection policy is called the Optimized Policy and is a modified version of the Ford Ferguson algorithm in which an optimal matching between inputs and outputs at a node is performed [Tuc84]. If more than one optimal match exists, one is chosen at random. Messages are assigned to the first free physical channel. When no free channel exists, the message is assigned to any useful channel. This reduces contention and message latency because unnecessary channel multiplexing is avoided.

This optimized algorithm was only simulated as a reference because it is too complex and time-consuming to implement. The other five policies simulated here are more practical. The first of these five policies is the Random Policy in which both the input and output channels are randomly selected.

The remaining four input/output selection policies, use different input policies while keeping the output policy the same. This constant output selection policy is as follows: the dimension of the messages having the greatest number of hops left is selected first. If the necessary channels in this dimension are unavailable (e.g. in use by other messages), the dimension with the next highest number of hops will be chosen. This continues until the message is either routed or blocked. Upon a tie, the smallest numbered dimension is chosen. Using this output selection policy, the greatest amount of adaptivity for a message is retained, reducing contention as traffic increases.

The four input selection policies used with this output policy are as follows: A random input selection policy (Partial Random Policy) in which the input buffers are randomly selected. If the randomly selected input buffer contains a message, the message from this input buffer is routed if possible. If not, the message in the next randomly selected input buffer is routed if possible. This continues until a message has been routed or until every input buffer at the node has been selected once.

The second input selection policy (Incremental Policy) selects the input buffers in increasing order. Each cycle the first message in the buffer labeled with dimension zero and channel zero is routed first if possible. Then the first message in the buffer labeled with dimension zero and channel one is routed if possible. This increasing order continues until a message has been routed or until all buffers in each dimension and channel have been attempted.

For the third input selection policy (Round Robin Policy), a round robin selection is used among all of the input buffers. Each input buffer is routinely checked once every n times, where n is the number of input buffers.

The fourth input selection policy is an enhancement of the previous round robin policy (Prioritized Round Robin Policy). In this algorithm, there exists two round robin rings.

The ring used first contains a round robin ordering of all those messages previously blocked. The second list contains a round robin ordering of all the remaining messages that were not previously blocked. This is the most fair of all the practical approaches mentioned here. However, it is also more complex to implement.

3.1.1 Simulation Results

Simulations were performed using the slightly modified version of [Dua93] along with cut-through switching and the six input/output selection policies explained above.

Simulation results were obtained for a 10-ary 3-cube network. A stabilization threshold of a 0.005 difference between channel utilization 1000 clock cycles apart was used to determine steady state before collection of data began. The number of clock cycles simulated was 10,000 and message sizes of 8, 16 and 32 bytes were used. Simulations consisted of varying buffer size while holding number of virtual channels constant and varying the number of virtual channels while keeping buffer size constant. The type of switching done was virtual cut-through switching, and channel utilization between (and including) 0.10 and 1.20 were simulated.

Note that no pipelining was performed for this section's results. Therefore, when no blocking exists, only one clock cycle is required to route and send a message from the current node to the next node. This one clock cycle must be long enough to include the following actions: address decoding and connection requests, arbitration, updating selected headers, transferring data through the crossbar switch, and any controller delay required for the virtual channels. The actual time required for a clock cycle will not be taken into account in this section.

Some representative simulation results for all input/output selection policies used are shown in Figure 3.1. This Figure shows that when the buffer size increases for a given number of virtual channels equal to 3, the message latencies at low channel utilization for

all six policies are approximately the same. However, the random input/output selection policy and optimized algorithms do perform slightly worse than the other four policies. At high utilization, although all policies once again perform almost identically, the optimized policy performs the best, followed by the prioritized round robin input/greatest number of hops output policy.

An interesting characteristic develops, however, when the number of virtual channels is set to six, instead of three, and the buffer size is once again varied. When this happens, the optimized routing algorithm has the smallest message latencies at low channel utilization, followed by the random input/output policy. All other policies have higher and similar message latencies. This is because when the number of channels increases from three to six, the optimized algorithm is the most efficient among all of the input/output selection policies at using the extra available channels for improved throughput and lower message latencies.

Though the optimized algorithm does better at low utilization, the remaining policies have the lowest message latencies and highest saturation points at high utilization. This is because the optimized routing policy is a greedy algorithm and gives priority to short-term profits instead of long-term. On the other hand, because of the fairness of the round robin input policies and because of the retention of choices by the greatest number of hops output policy, the two round robin methods can decrease the probability of blocking and increase saturation. Note that this same behavior is observed for message sizes of 8 and 16 bytes as well.

3.1.2 Conclusion

Although all policies give similar results under most conditions (excluding the optimized algorithm), the simplest and best performing policies overall are the round robin input/greatest number of hops output policy (Round Robin policy), prioritized round robin

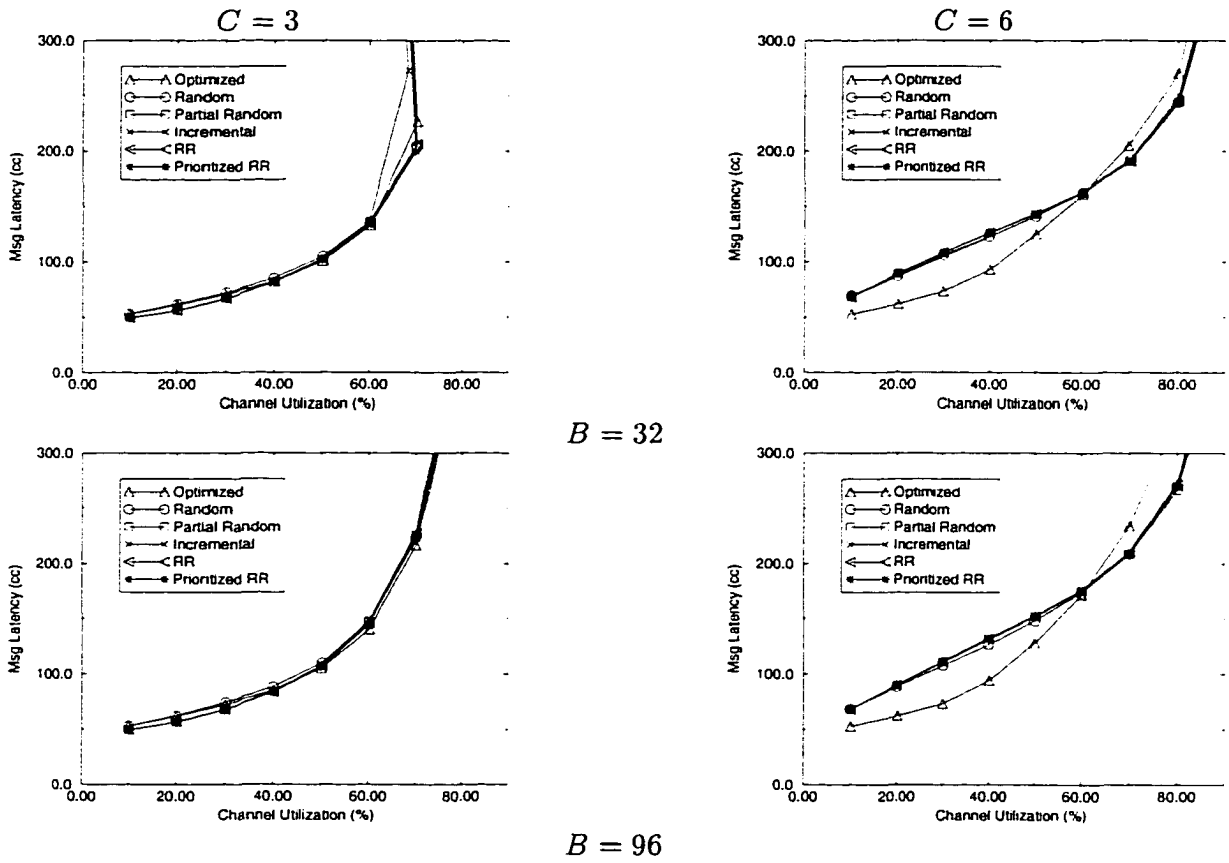


Figure 3.1: Comparison of input/output selection policies for a 10-ary 3-cube network with $L = 32$.

input/greatest number of hops output policy (Prioritized Round Robin policy), and random input/greatest number of hops output policy (Partially Random policy). Because the Round Robin policy performed relatively the same as the Prioritized Round Robin policy and because it is also easier to implement, it was chosen over the Prioritized Round Robin policy. It was also chosen over the Partially Random policy because it is more fair and just as easy to implement. The remaining simulations for this proposal will only use this input/output policy.

3.2 Adaptive Router Performance

The relationship between increasing buffer size versus increasing the number of VCs was addressed in [Dal92] using wormhole switching and dimension-ordered routing with

FIFO queues. It was shown that when the total amount of buffer storage per node was held constant, adding lanes to a network is significantly more effective than than adding depth to a channel. Further investigation [Dua93] showed that adding VCs to an adaptive algorithm implementation under wormhole switching reduces message latency and increases throughput. However, it was also shown that adding VCs has two opposite effects: while it reduces channel contention at high utilization, at low utilization it increases message latency because channel bandwidth is now shared among the requesting messages.

Since neither [Dal92] nor [Dua93] include any results for cut-through switching, this section extends their research to include this switching type. In addition, the input/output selection policy used here is the one explained in the previous section called the Round Robin Policy. In this policy, a round robin input selection/greatest number of hops output selection policy is used.

Simulations were once again performed using an adaptive cut-through simulator and a slightly modified version of [Dua93] for a 10-ary 3-cube network. A stabilization threshold of a 0.005 difference between channel utilization 1000 clock cycles apart was used to determine steady state before collection of data began. The number of clock cycles simulated was 10,000 and message sizes of 8, 16 and 32 bytes were used. Simulations consisted of varying buffer size while holding number of VCs constant, varying the number of VCs while keeping buffer size constant, and, finally, keeping total buffer storage per dimension constant. Channel utilization between (and including) 0.10 and 1.20 were simulated. Once again, no pipelining in the router was performed for this section. and the actual time required for one clock cycle is not taken into account.

3.2.1 Constant number of Virtual Channels

For the first simulations, the effects of buffer size were investigated by keeping the number of VCs constant while varying buffer size. Figure 3.2 shows the results. As buffer

size increases for a constant small number of VCs (i.e. C=3), little difference exists between the message latencies for different sized buffers below the saturation point. This results because at low utilization not enough messages exist to completely fill even the small buffers. However, at high utilization, larger buffer sizes produce smaller message latencies and higher saturation points. The saturation points increase because at high utilization the network now holds more messages.

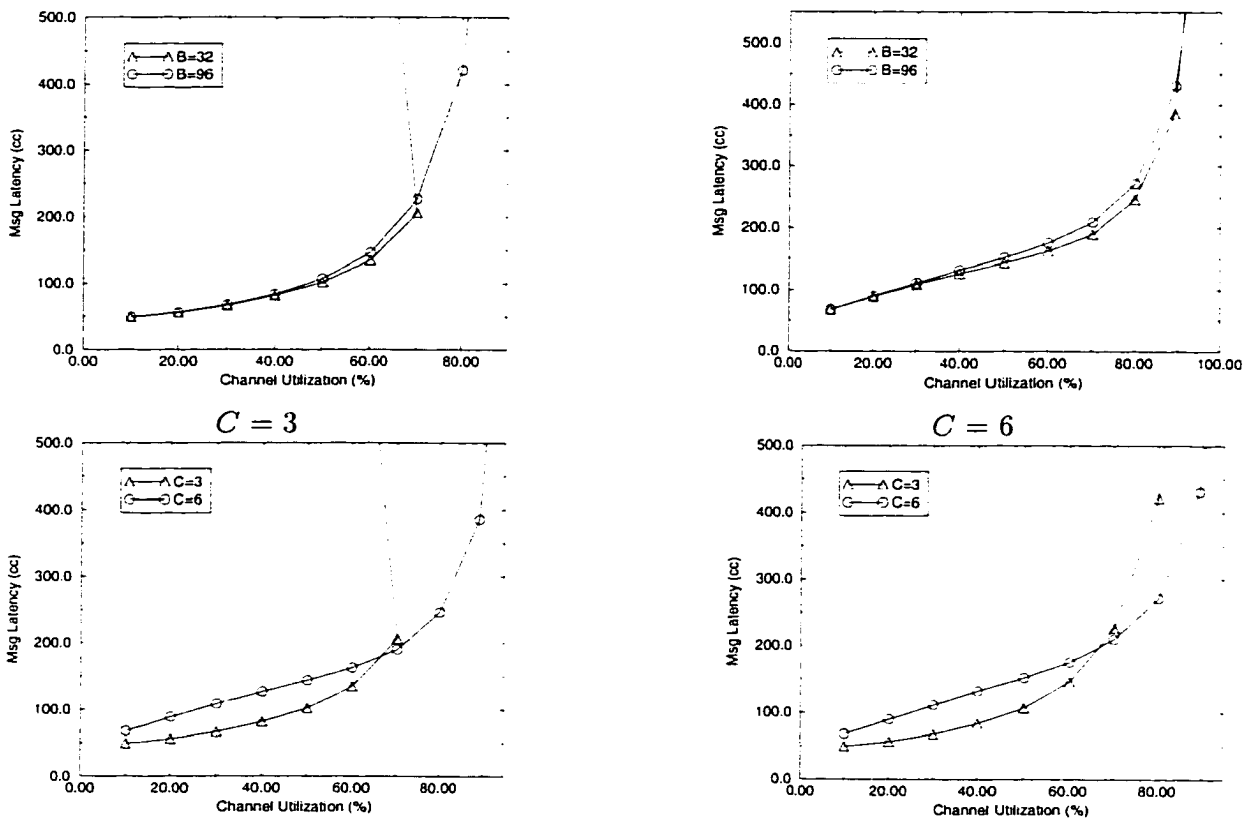
On the other hand, when the buffer size was increased for a constant large number of VCs (i.e. C=6), the difference in message latencies at high and low channel utilization for varying buffer sizes diminished.

These results show that for cut-through switching the buffer size dominates performance for small numbers of VCs but not for greater numbers.

3.2.2 Constant Buffer Size

The impact that the number of VCs has on performance was investigated by varying the number of VCs from 3 to 6 while holding the buffer size constant. The simulation results shown in Figure 3.2 indicate that at low utilization as the number of VCs increase, message latency increases as well. Therefore, fewer number of VCs are more beneficial if low utilization is expected because at low utilization the probability of blocking is low and the extra VCs are not required to pre-empt any blocked messages. However, as message size increases, at low utilization the difference between message latencies for various number of VCs is large. Therefore, when long messages are used at low utilization, larger buffers are more beneficial than more VCs.

At high utilization, however, just the opposite is true. The probability of blocking is very high and the extra VCs are needed for the pre-emption of blocked messages. As the number of VCs increase, the saturation point drastically increases as well.



$C = 3$ $C = 6$

$B = 32$ $B = 96$

Figure 3.2: Comparison of adaptive routing for a 10-ary 3-cube with $L = 32$

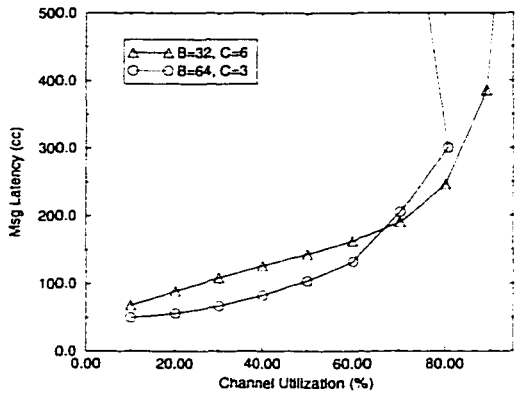


Figure 3.3: 10-ary 3-cube under constant bandwidth for $L = 32$

3.2.3 Constant Bandwidth Results

Keeping total buffer storage per dimension constant was investigated next. The results obtained are very similar to those in the previous two sections and are shown in Figure 3.3. At low utilization a smaller number of VCs with large buffers produce lower latencies than a greater number of VCs with smaller buffers. However, at high channel utilization, just the opposite is true.

The reason for this behavior is that, once again, at low utilization the additional channels are not necessary since message blocking is rare. In fact, if more channels are available at this low utilization, latencies will be higher due to message interleaving. However, at high utilization, message blocking becomes greater since more messages are now present and it is important for unblocked messages to be able to pass blocked messages. This "pre-emption" of blocked messages is accomplished through the increase in the number of adaptive channels.

3.2.4 Conclusion

When varying buffer size for a given number of VCs, the buffer size dominates performance for small number of VCs. For a greater number of VCs, buffer size no longer dominates. On the other hand, when the number of VCs was increased for a given buffer

size, the message latency at low utilization also increased. At high utilization, however, just the opposite was true: more VCs meant lower message latencies and higher saturation points. These results indicate that better performance is obtained by using fewer number of VCs if low utilization is expected and more VCs for high utilization applications.

The constant bandwidth results also support the above conclusions. At low channel utilization a smaller number of VCs with large buffers produce lower latencies than a greater number of VCs with smaller buffers. However, at high utilization, just the opposite is true.

3.3 Physical Wire Bandwidth

This section addresses resource utilization in adaptive routing. Not only is knowledge of resource utilization important when making decisions concerning the trade-off between wire bisection width and improved performance, but is also important in the area of trading performance for fault tolerance. First, the tradeoffs involved in wire bisection width (1 and 2 physical channels (PCs) per dimension) versus performance is evaluated for adaptive routing and compared with deterministic routing performance. Second, the impact of constant bisection width is determined. Note that constant bisection width is defined here as the number of physical channels per dimension times the number of wires per PC. Finally, sink bandwidth effects are discussed with respect to differing number of physical channels.

3.3.1 Simulation Setup

Once again the slightly modified version of Duato's adaptive routing algorithm [Dua93] is simulated here and the results are compared with dimension-order routing results. Both the deterministic and adaptive routing algorithms are implemented using one and two *physical channels* (PCs) per dimension per node. When one PC is implemented per dimension per node, all deterministic and any adaptive virtual channels (VCs) are multiplexed onto one physical channel using the round robin policy. When two PCs are implemented, the

deterministic and any adaptive VCs are multiplexed onto both PCs in, once again, round robin fashion.

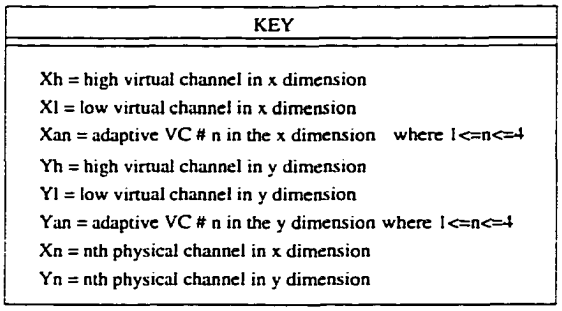
When sink bandwidth effects are considered, the number of physical sink channels is varied from one to infinite. Note that once a sink channel is assigned to a message, it is not released until the whole message has finished its transmission. Finally, all channels are unidirectional channels. Figure 3.4 shows schematics for a deterministic and adaptive router in the 2D case. Note that because the results in this section are not pipelined, the input latches in the deterministic router figure and the output latches in the adaptive router figure are not simulated in this section.

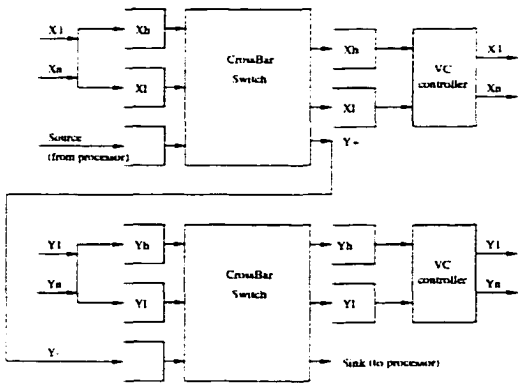
3.3.2 Experimental Results

Both deterministic and adaptive routing simulations were performed under uniform traffic conditions using a discrete-time simulator. Simulation results were obtained for various k-ary 3-cube and k-ary 2-cube network configurations. Message generation rates varied from 0.0694% to 0.694% in 0.0694% increments, where the message generation rate is the probability that a node will generate a new message at each clock cycle. The simulation uses a stabilization threshold of 5 percent difference between message generation rates 1000 clock cycles apart to determine steady state. Message sizes varied from 8 to 32 flits and the buffer sizes simulated for each type of adaptive routing are one, two and three times the message size. The number of virtual channels used by the adaptive router varies from three to six VCs per dimension per node.

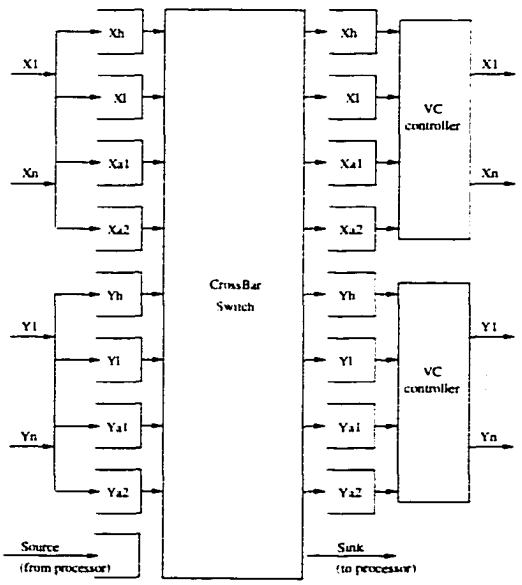
The simulator implements a back-pressure mechanism which results in a negative slope of the latency versus message generation rate plots at higher loads. The following notation is used throughout this section: L = message length, B = buffer size, VC = number VCs per PC, PC = number of physical channels.

1. **Impact of Bisection Width on Performance**





(a) Deterministic Router



(b) Adaptive Router

Figure 3.4: Schematics of 2D deterministic (a) and adaptive (b) routers with n PCs

In this section, the tradeoffs involved in wire bisection width (1 and 2 PCs per dimension) versus performance are evaluated for adaptive routing. Adaptive routing is also compared with deterministic routing.

Figure 3.5 shows the performance for one versus two physical channels per dimension for adaptive routing. In both cases the tradeoff between the size and number of virtual channels can be seen: at low channel utilization, fewer number of virtual channels result in a lower message latency. At high utilization, fewer number of virtual channels result in a lower saturation point. This phenomenon can be explained by the fact that at low channel utilization the interleaving of the messages among many virtual channels results in a higher latency. However, at high utilization, message blocking becomes more frequent and the additional adaptive virtual channels can alleviate the congestion and reduce the blocking time of messages.

This phenomenon is observed less when there are two PCs per dimension than when there is only one PC and for both cases not much benefit is gained from increasing buffer size. Increasing the number of VCs per dimension, however, does improve performance greatly. Finally, it is seen that increasing the number of PCs from one to two immensely improves both message latency and saturation point.

The tradeoff of wire bisection width versus performance for deterministic routing is shown in Figure 3.5. By comparing this figure with the one for adaptive routing, it can be seen that the same behavior that exists for both cases of adaptive routing also exists for both cases of deterministic routing when buffer size is increased. Note that increasing the number of PCs from one to two does not increase performance nearly as much in deterministic routing as in adaptive routing.

To better compare wire bisection width versus performance under both types of routing, message latency speedup and saturation point ratios were calculated as shown in

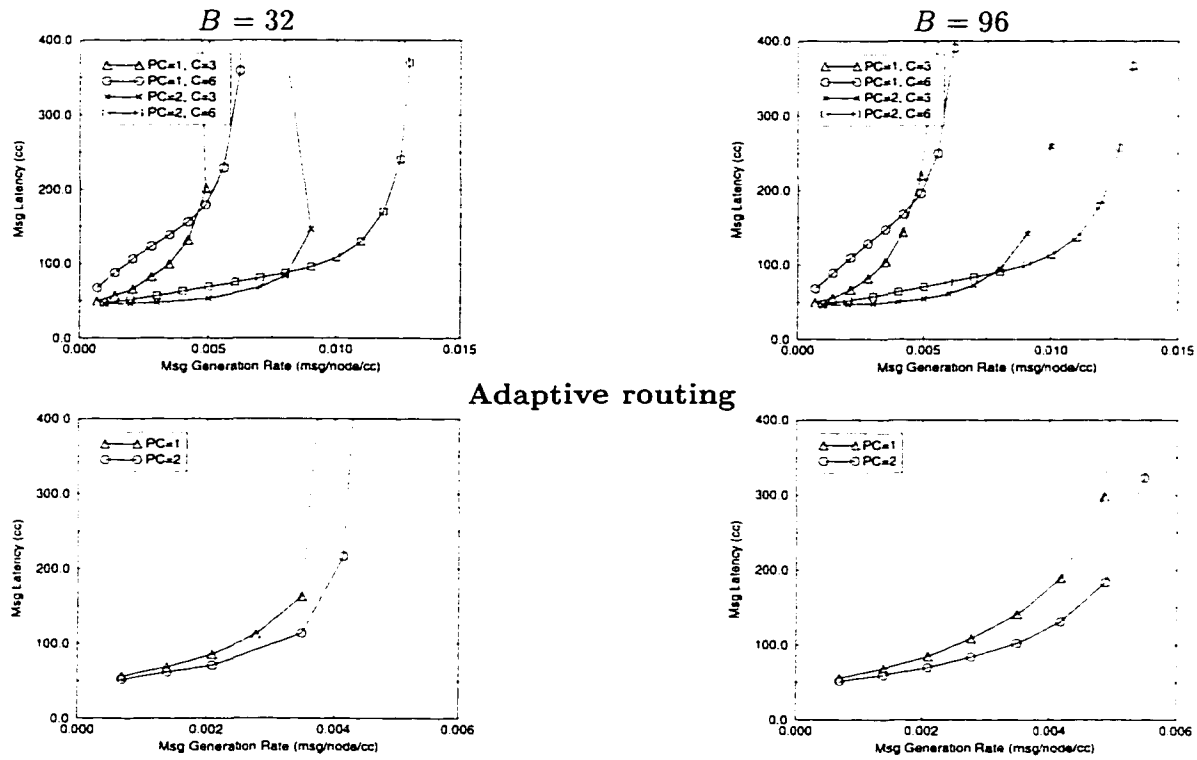


Figure 3.5: Tradeoff of wire bisection width versus performance for a 10-ary 3-cube network with L=32 flits and infinite number of sink channels

B	C	Msg Generation Rate						
		0.001	0.002	0.003	0.004	0.005	0.006	0.007
Deterministic Routing								
32	2	1.07	1.20	1.66	2.27	1.58	1.12	1.09
96		1.11	1.22	1.31	1.44	1.63	1.60	1.34
Adaptive Routing								
32	3	1.15	1.41	1.85	2.60	36.77	34.60	34.49
	6	1.64	2.07	2.31	2.52	3.05	4.78	8.38
96	3	1.16	1.42	2.03	2.87	3.89	15.84	29.51
	6	1.65	2.12	2.45	2.62	3.11	5.08	8.22

Table 3.1: Message latency speedup: 10-ary 3-cube, L=32 flits

B	C	Saturation Point Ratio
Deterministic Routing		
32	2	1.16
96	2	1.00
Adaptive Routing		
32	3	1.93
	6	1.87
96	3	1.81
	6	1.89

Table 3.2: Saturation points: 10-ary 3-cube, L=32 flits

Tables 3.1 and 3.2. In this analysis the saturation point is defined as the first message generation rate at which the *accepted* load in the network is less than $\pm 1.0\%$ of the *offered* load.

These tables clearly show that increasing wire bisection width by adding an extra PC in each dimension can increase the message latency speedup from 0.07 times to 2.27 times for deterministic routing and from 0.15 to 36.77 times for adaptive routing on a 10-ary 3-cube network for 32 flit message lengths and various buffer sizes and number of virtual channels.

In addition, the saturation point ratios result in zero increase to 16% increase for deterministic routing and 81% to 89% greater for adaptive routing. These large

differences in message latency speedup and saturation point ratios between deterministic and adaptive routing demonstrate that the physical channels are more efficiently utilized in adaptive routing than in deterministic routing. This results because the network dimensions are ordered in dimension-order routing. By assigning this order to the network dimensions, the highest priority channels become a bottle neck resulting in under-utilization of lower priority channels with the extra physical channels in these dimensions remaining idle.

However, in adaptive routing, the adaptive VCs can be allocated in any dimension. Therefore, under uniform traffic, no dimension becomes a bottleneck and the extra physical channels are efficiently utilized.

2. **Constant Bisection Width for Adaptive Routing**

Constant bisection width results for adaptive routing are shown in Figure 3.6. Note that bisection width is defined here as the number of PCs per dimension times the number of wires per PC. In addition, to simulate constant bandwidth, the network with two PCs per dimension is given twice the message length as the network with one PC per dimension.

Under constant bandwidth, the use of one PC has slightly better performance (slightly lower message latency and higher saturation point) for small number of VCs (C=3) than two PCs. However, for a large number of VCs (C=6), both cases have very similar behavior. If an extra PC is used, for example, to gain fault tolerance in an adaptive router design, a small amount of performance is lost when few VCs are implemented. However, if the number of VCs is increased, this performance loss becomes negligible. Once again, increasing buffer size in both cases only improves the performance slightly.

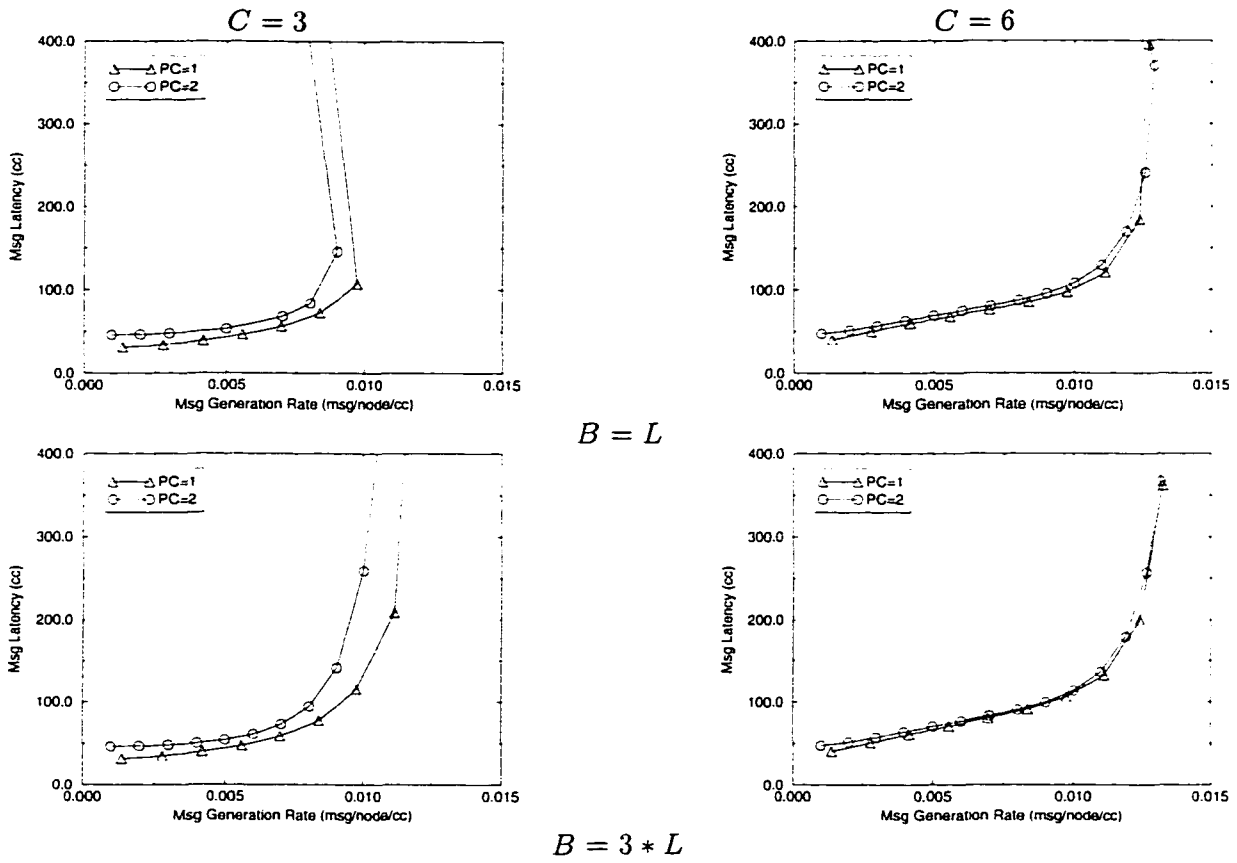


Figure 3.6: Impact of constant bisection width for adaptive routing on a 10-ary 3-cube with an infinite number of sink channels

3. **Sink Bandwidth**

It has been shown that efficient removal of messages from the network is extremely important for fast router performance [ES93]. Therefore, this section uncovers the effects that increasing number of PCs per dimension has on sink bandwidth. The results are shown in Figure 3.7.

These results show that when one PC per dimension is implemented, only one physical sink channel is required to efficiently remove messages from the network. However, when two PCs per dimension are used, two physical sink channels are required for performance comparable to an infinite number of physical sink channels. Once again, there is not much performance gain by increasing buffer size.

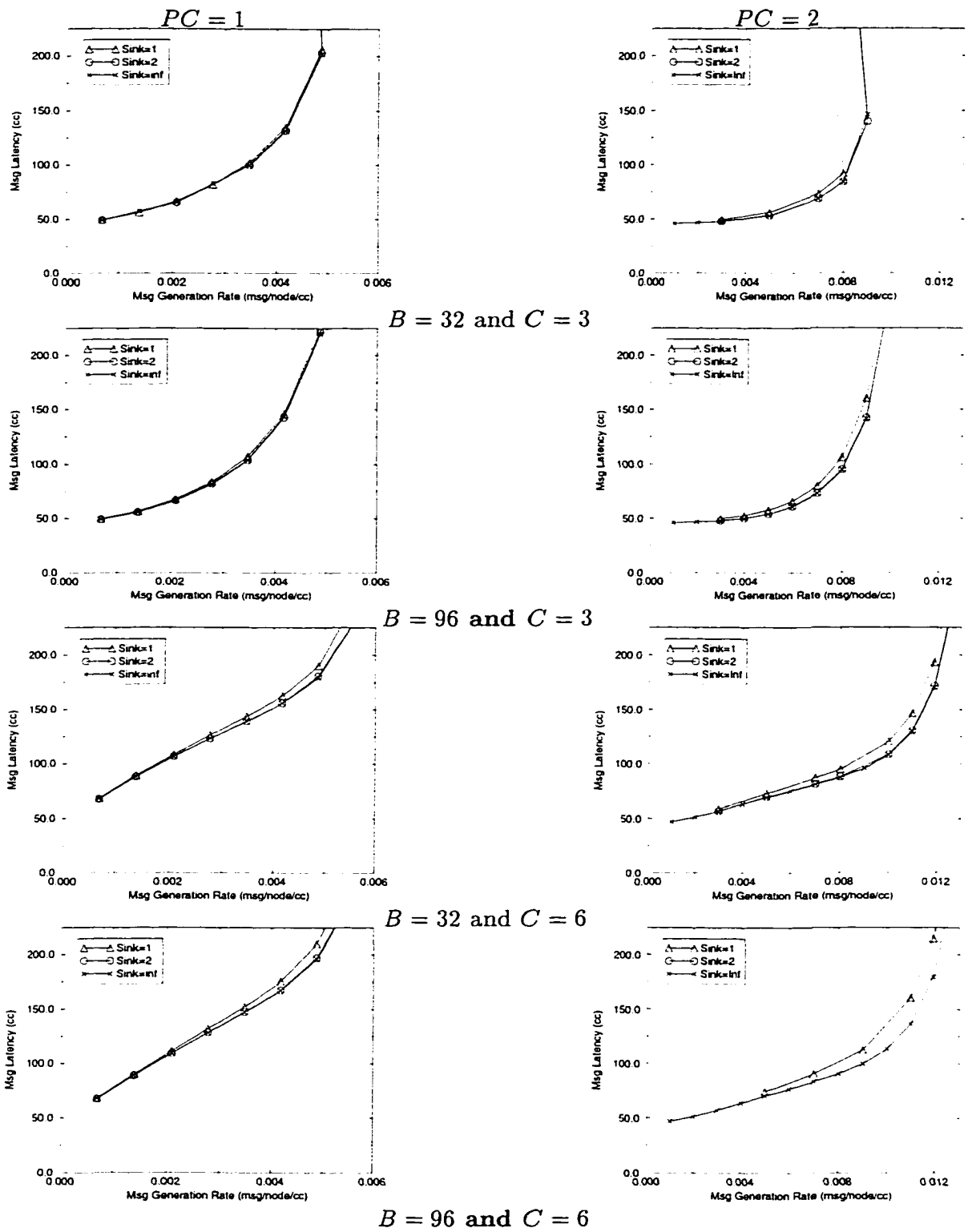


Figure 3.7: Impact of sink bandwidth on adaptive routing latency for a 10-ary 3-cube network with $L = 32$

4. **Discussion of Results**

Note that router clock cycle times were not included in the above experimental simulations. However, when comparing the performance of one and two PCs per dimension, the cycle times do not vary greatly. This conclusion is obtained using the router model described previously in Section 2.2 [Chi93, AC95]. In this model, only the T_{ARB} and T_C terms are affected by the addition of a PC because these are the only terms which interface with the PCs. For the T_{ARB} term, the only modification includes allowing two channel resources to be allocated per dimension instead of one. However, this logic modification would only slightly increase the speed of the cycle time. For the T_C term, simple logic would be required for the VC controller to extend the round robin policy to include two PCs instead of one and an extra buffer and a few more select and data lines would be required. However, these would also only cause a slight increase in cycle time.

3.3.3 Conclusion

We have investigated the effect of resource utilization for adaptive routing by evaluating wire bisection width versus performance for adaptive routers and by comparing these adaptive routing results with deterministic routing. In addition, the impact of constant bisection width in adaptive routers was discussed. Finally, the effects of increasing the number of PCs per dimension with respect to sink bandwidth was presented.

3.4 Effects of Router Complexity and Delay on Performance

This section addresses the issue of how router complexity affects the overall performance in deterministic and adaptive routing under virtual cut-through switching in k-ary n-cube networks. First constant area performance between adaptive routers is compared. Next constant area performance between both adaptive and deterministic routers is com-

pared. Finally, under certain circumstances, it is shown that deterministic routers can reach saturation points comparable to adaptive routers.

3.4.1 Simulation Setup

Once again dimension-order routing [DAC89, DS87b] and a modified version of Duato's algorithm [Dua91, Dua93, BGPS92] are used on k-ary n-cube with virtual cut-through switching. In addition, three different traffic patterns are considered:

- *Random Uniform:* Source and destination nodes are uniformly distributed.
- *Complement:* Node $a_{n-1}a_{n-2}...a_1a_0$ communicates with node $\overline{a_0a_1...a_{n-2}a_{n-1}}$
- *Perfect Shuffle:* Node $a_{n-1}a_{n-2}...a_1a_0$ communicates with node $a_{n-2}a_{n-3}...a_0a_{n-1}$

Both the deterministic and adaptive routing algorithms were implemented using one *physical channel* (PC) per dimension per node. In addition, both the deterministic and adaptive routers are pipelined requiring buffers on both the input and output channels. The input buffer of the deterministic router ensures that no message is lost if a conflict arises. Note that the deterministic router uses storage buffers associated with output channels, while the adaptive router uses storage buffers associated with input channels. Schematics of the two routers are shown in Figure 3.4.

In the deterministic routing case, both high and low VCs of each dimension are multiplexed onto one PC. In the adaptive routing case, the deterministic and adaptive VCs are multiplexed onto one PC. For both cases there is only one PC for the sink channel. Once this channel is assigned to a message, it is not released until the whole message has finished its transmission. Finally, all channels are unidirectional channels.

3.4.2 Modeling Router Delay

The model for router complexity as described previously in Section 2.2 [Chi93, AC95, DL94] is used in this chapter's results for both deterministic and adaptive routing. These

B	T_r	T_s	T_c	CC Period
8	4.70	4.75	6.74	6.74
16	4.70	5.35	6.74	6.74
24	4.70	5.70	6.74	6.74
32	4.70	5.95	6.74	6.74
48	4.70	6.30	6.74	6.74
64	4.70	6.55	6.74	6.74
96	4.70	6.90	6.74	6.90

Table 3.3: Delays for deterministic routing algorithm for k-ary 2-cube and 3-cube networks ($C = 2$, $P = 3$, and $F = 1$ for all); All values are given in nanoseconds

models account for both the logic complexity of the routers as well as the size of the crossbar as determined by the number of VCs that are multiplexed on one PC. These models were modified here to account for the varying buffer space used in virtual cut-through switching.

For all dimension-order routers simulated here, the number of degrees of freedom (F) equals one since there exists a single routing option for each message. The number of switch crossbar ports (P) is three because a deterministic router routes a message in either the same dimension on which the message came (on either the low or high channel) or routes it to the next dimension. For all of the adaptive routers $F = P - 2(n - 1)$, where n is the number of network dimensions. This relationship holds because adaptive routing can use any adaptive channel along with the two deterministic channels in the highest dimension that a message still has hops in. Note that this relationship includes the delivery port.

The delay values were calculated for each of the router algorithms simulated and are shown in Tables 3.3, 3.4 and 3.5.

The following observations can be made about the data in Tables 3.3, 3.4 and 3.5:

- In deterministic routers, increasing buffer size increases the overall router delay when moderate to large buffer sizes are used. For small buffer sizes the clock cycle is dominated by the transfer time T_c while for larger ones it is dominated by the switching time T_s.

B	P	F	T_r	T_s	T_c	CC Period
$C = 3$						
8	7	5	7.49	5.48	7.09	7.49
16			7.49	6.08	7.09	7.49
24			7.49	6.44	7.09	7.49
32			7.49	6.68	7.09	7.49
48			7.49	7.04	7.09	7.49
64			7.49	7.28	7.09	7.49
96			7.49	7.64	7.09	7.64
$C = 4$						
8	9	7	8.07	5.70	7.34	8.07
16			8.07	6.30	7.34	8.07
24			8.07	6.65	7.34	8.07
32			8.07	6.90	7.34	8.07
48			8.07	7.25	7.34	8.07
64			8.07	7.50	7.34	8.07
96			8.07	7.85	7.34	8.07
$C = 5$						
8	11	9	8.50	5.88	7.53	8.50
16			8.50	6.48	7.53	8.50
24			8.50	6.83	7.53	8.50
32			8.50	7.08	7.53	8.50
48			8.50	7.43	7.53	8.50
64			8.50	7.68	7.53	8.50
96			8.50	8.03	7.53	8.50
$C = 6$						
8	13	11	8.85	6.02	7.69	8.85
16			8.85	6.62	7.69	8.85
24			8.85	6.97	7.69	8.85
32			8.85	7.22	7.69	8.85
48			8.85	7.57	7.69	8.85
64			8.85	7.82	7.69	8.85
96			8.85	8.17	7.69	8.85

Table 3.4: Delays for adaptive routing algorithm for k-ary 2-cube networks; All values are given in nanoseconds

B	P	F	T_r	T_s	T_c	CC Period
$C = 3$						
8	10	6	7.80	5.79	7.09	7.80
16			7.80	6.39	7.09	7.80
24			7.80	6.74	7.09	7.80
32			7.80	6.99	7.09	7.80
48			7.80	7.34	7.09	7.80
64			7.80	7.59	7.09	7.80
96			7.80	7.94	7.09	7.94
$C = 4$						
8	13	9	8.50	6.02	7.34	8.50
16			8.50	6.62	7.34	8.50
24			8.50	6.97	7.34	8.50
32			8.50	7.22	7.34	8.50
48			8.50	7.57	7.34	8.50
64			8.50	7.82	7.34	8.50
96			8.50	8.17	7.34	8.50
$C = 5$						
8	16	12	9.00	6.20	7.53	9.00
16			9.00	6.80	7.53	9.00
24			9.00	7.15	7.53	9.00
32			9.00	7.40	7.53	9.00
48			9.00	7.75	7.53	9.00
64			9.00	8.00	7.53	9.00
96			9.00	8.35	7.53	9.47
$C = 6$						
8	19	15	9.39	6.35	7.69	9.39
16			9.39	6.95	7.69	9.39
24			9.39	7.30	7.69	9.39
32			9.39	7.55	7.69	9.39
48			9.39	7.90	7.69	9.39
64			9.39	8.15	7.69	9.39
96			9.39	8.50	7.69	9.62

Table 3.5: Delays for adaptive routing algorithm for k-ary 3-cube networks; All values are given in nanoseconds

- In adaptive routers, the clock cycle time is dominated by T_r. Increasing buffer size only increases the overall router delay when small number of VCs and large buffer sizes are used.

- Varying buffer size affects deterministic routers' clock cycles more than adaptive routers'.

All of these added delays result in adaptive routers that are 13 to 30 % slower than deterministic routers. To offset this difference, adaptive routers must perform at least 12 to 40 % better than deterministic routers. These results are similar to the results in [AC95] where 15 % to 60 % improvement is required for f-flat routers with similar number of VCs and under wormhole routing. The objective of this analysis is how much of this delay can be offset by lower queuing delay in virtual cut-through.

3.4.3 Experimental Results

Both deterministic and adaptive routing simulations were performed under three different traffic conditions using a discrete-time simulator. Simulation results were obtained for various k-ary 3-cube and k-ary 2-cube networks. The simulation uses a stabilization threshold of a 0.005 difference between throughput 1000 clock cycles apart to determine steady state. Message sizes varied from 8 to 32 flits and throughput from 0.1 until saturation was reached in 0.1 increments. The buffer sizes simulated for adaptive and deterministic routing are one, two and three times the message size. The number of VCs used by the adaptive router varies from three to six VCs per dimension per node, while the number used by the deterministic router is always two. The simulator implements a back-pressure mechanism which results in a negative slope of the latency versus throughput plots at higher loads. The following notation is used throughout this section:

Area	L	B	C	Ran	Bit	Per
48	8	8	6	0.320	0.290	0.275
		16	3	0.330	0.289	0.258
96	16	16	6	0.319	0.302	0.277
		32	3	0.345	0.300	0.258
192	32	32	6	0.320	0.301	0.280
		64	3	0.313	0.275	0.241

(a) Adaptive routing

Type	Area	L	B	C	Ran	Bit	Per
Deter	128	32	64	2	0.236	0.219	0.129
Adapt			32	4	0.316	0.278	0.264
Deter	192	32	96	2	0.245	0.233	0.130
Adapt			32	6	0.320	0.301	0.280
			64	3	0.313	0.275	0.241

(b) Constant area for deterministic and adaptive routing

Type	Area	L	B	C	Ran	Bit	Per
Deter	48	8	24	2	0.272	0.257	0.143
Adapt	24		8	3	0.276	0.258	0.219
Deter	96	16	48	2	0.264	0.245	0.144
Adapt	48		16	3	0.289	0.258	0.229
Deter	192	32	96	2	0.245	0.233	0.130
Adapt	96		32	3	0.288	0.259	0.231

(c) Deterministic routing having twice the area as adaptive

Table 3.6: Throughput saturation points (flits/ns/node) for 10-ary 3-cube networks

L message size
B buffer size
C number of VCs per physical channel
D deterministic routing
A adaptive routing

1. **Constant Area Adaptive Routers**

Constant area is defined here as keeping the sum of all VC buffer sizes constant. The constant area results for various message lengths and traffic types are shown in Figures 3.8 and 3.9 with corresponding saturation points in Table 3.4.3a. In this

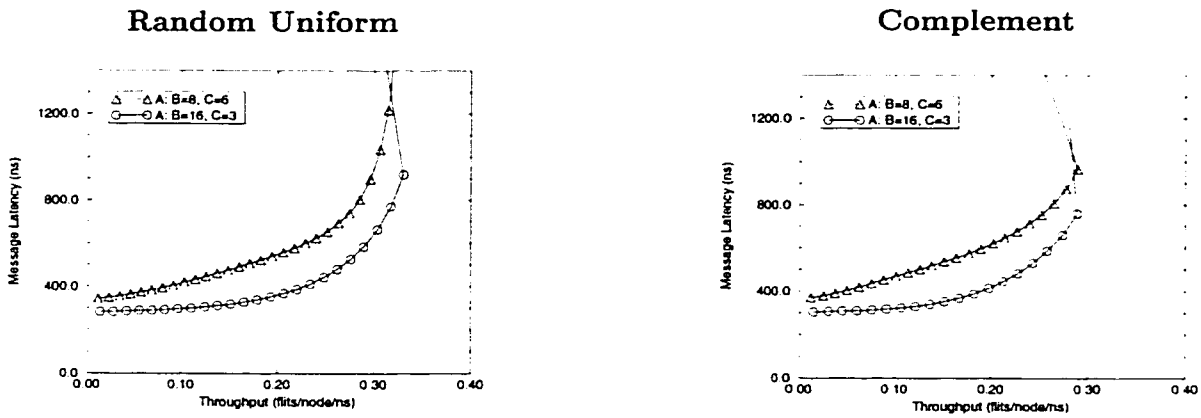
analysis the saturation point is defined as the last throughput at which the *accepted* load in the network was $\pm 1.5\%$ of the *offered* load.

The effects of traffic pattern on the saturation points shows that for all message sizes, the highest saturation points obtained are always for uniform random traffic patterns while the lowest saturation points are always obtained under perfect shuffle traffic. While the simulations performed under complement traffic have 5 % to 13 % lower saturation points than the uniform random traffic simulations, perfect shuffle traffic exhibits 13 % to 26 % lower saturation points.

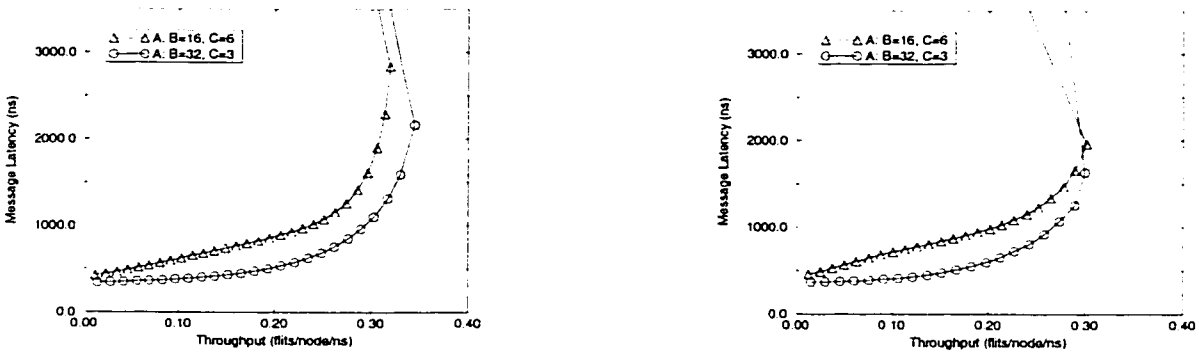
In addition, the results show that under random uniform traffic having fewer VCs always produces better performance. However, for less uniform traffic, a tradeoff exists between the size and number of VCs: at low throughput, fewer VCs result in a lower message latency. At high throughput, fewer VCs result in a lower saturation point. This phenomenon can be explained by the fact that at low throughput the interleaving of the messages among many VCs results in a higher latency.

However, at high throughput, message blocking becomes more frequent and the additional adaptive VCs can alleviate the congestion and reduce the blocking time of messages. Therefore, the larger clock cycle times required for numerous VCs pays off in the form of higher saturation points. Under random uniform traffic much interleaving of messages is not required for high saturation points. For less uniform traffic, the greater number of VCs (more interleaving) is crucial for higher saturation points.

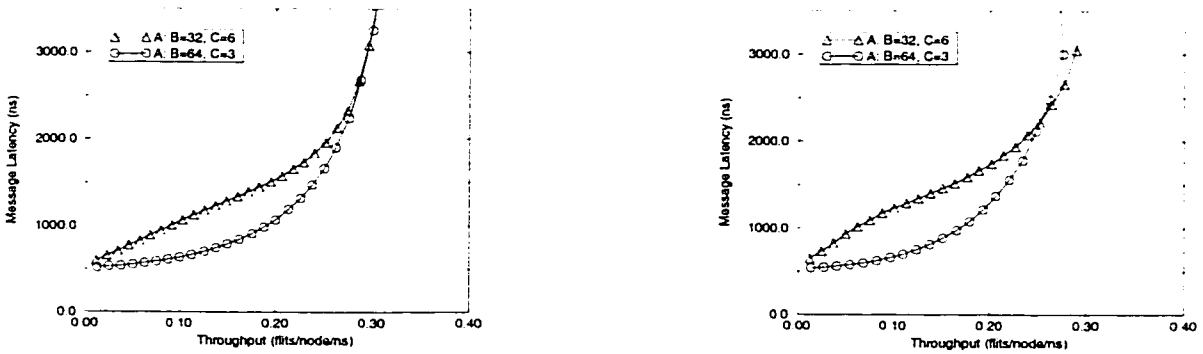
In [DS87b] Dally noted a similar phenomenon for dimension-order wormhole routing in k-ary n-cube networks. However, by accounting for the actual clock cycle times, the difference between a small and a large number of VCs at low throughput is more pronounced. This difference is also accentuated by the message size.



(a) Buffer area = 48 flits and L = 8 flits

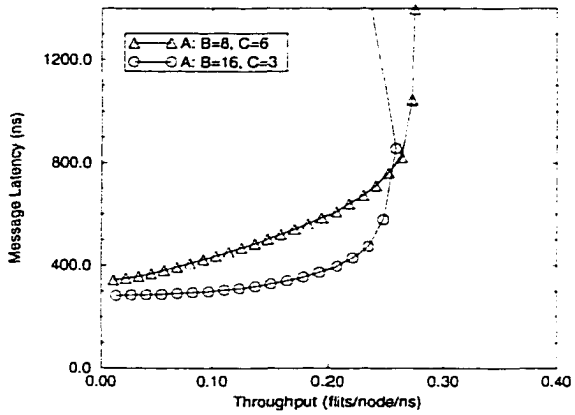


(b) Buffer area = 96 flits and L = 16 flits

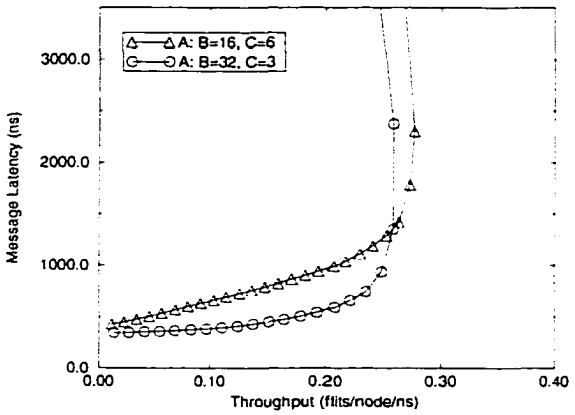


(c) Buffer area = 192 flits and L = 32 flits

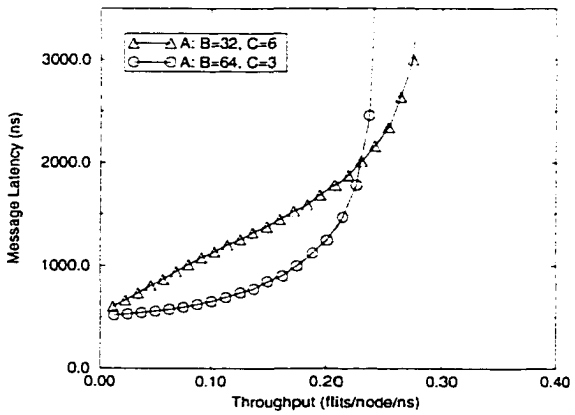
Figure 3.8: Adaptive routing latency for a 10-ary 3-cube network



(a) Buffer area = 48 flits and L = 8 flits



(b) Buffer area = 96 flits and L = 16 flits



(c) Buffer area = 192 flits and L = 32 flits

Figure 3.9: Adaptive routing latency for a 10-ary 3-cube network

2. **Constant Area Deterministic versus Adaptive Routers**

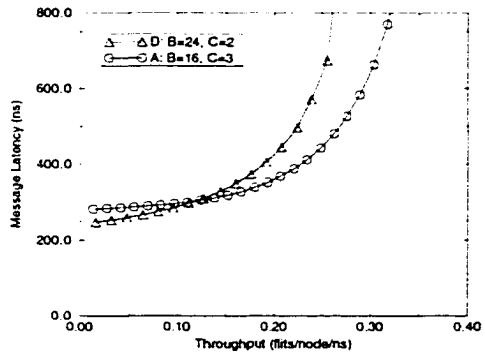
The constant area results for various message lengths and different traffic patterns are shown in Figure 3.10 with the corresponding saturation points in Table 3.4.3b. These results show the tradeoff between deterministic and adaptive routers under all three types of traffic: at low throughput, deterministic routing almost always results in lower message latency. At high throughput, adaptive routing always achieves higher saturation points. Note that deterministic routing has a particularly poor performance under perfect shuffle traffic due to its very localized traffic pattern.

Deterministic routing almost always results in lower message latency at low channel throughput because the router cycle time dominates the performance. At high throughput, adaptive routing results in a higher saturation point because the performance is now dominated by routing freedom which results in lower queuing delays. The only exception to this behavior is when buffer area equals 192. For each traffic type, adaptive routing performs either better than or comparable to deterministic routing when B=64 and C=3 even at low throughput. Under these circumstances. the additional delay due to increased buffer sizes for the deterministic router offsets its previous good performance at low throughput.

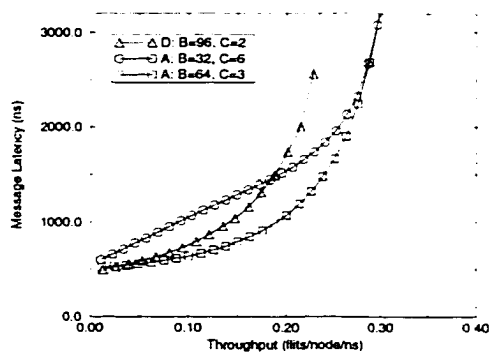
3. **Comparable Performance Deterministic and Adaptive Routers**

In this section, the performance of deterministic and adaptive routers with different total buffer areas are compared to determine if dimension-order routing can ever give similar performance to adaptive routing. The results are shown in Figures 3.11 and 3.12 for various message lengths and different traffic patterns. The corresponding saturation points are shown in Table 3.4.3c. The results for all three traffic types show that a deterministic router with $B_d = 3L$ and $C_d = 2$ gives about the same saturation point as an adaptive router with $B_a = L$ and $C_a = 3$ when message length is small

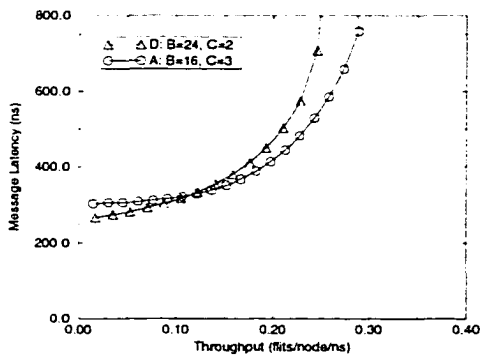
BA = 48 flits; L = 8 flits

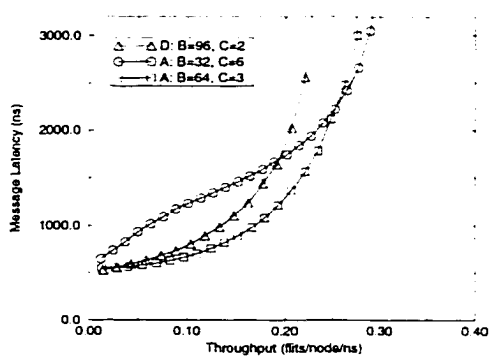


BA = 192 flits; L = 32 flits

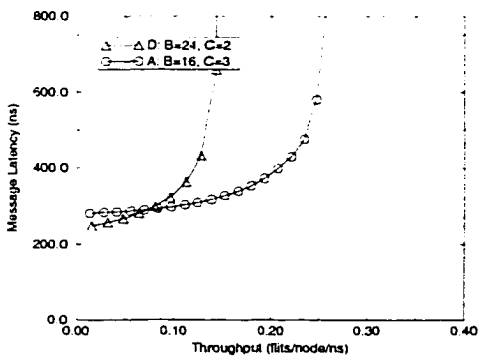


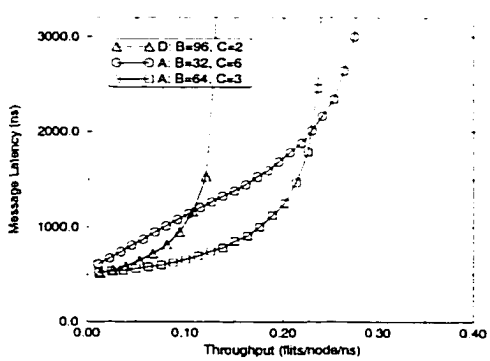
(a) Random Uniform





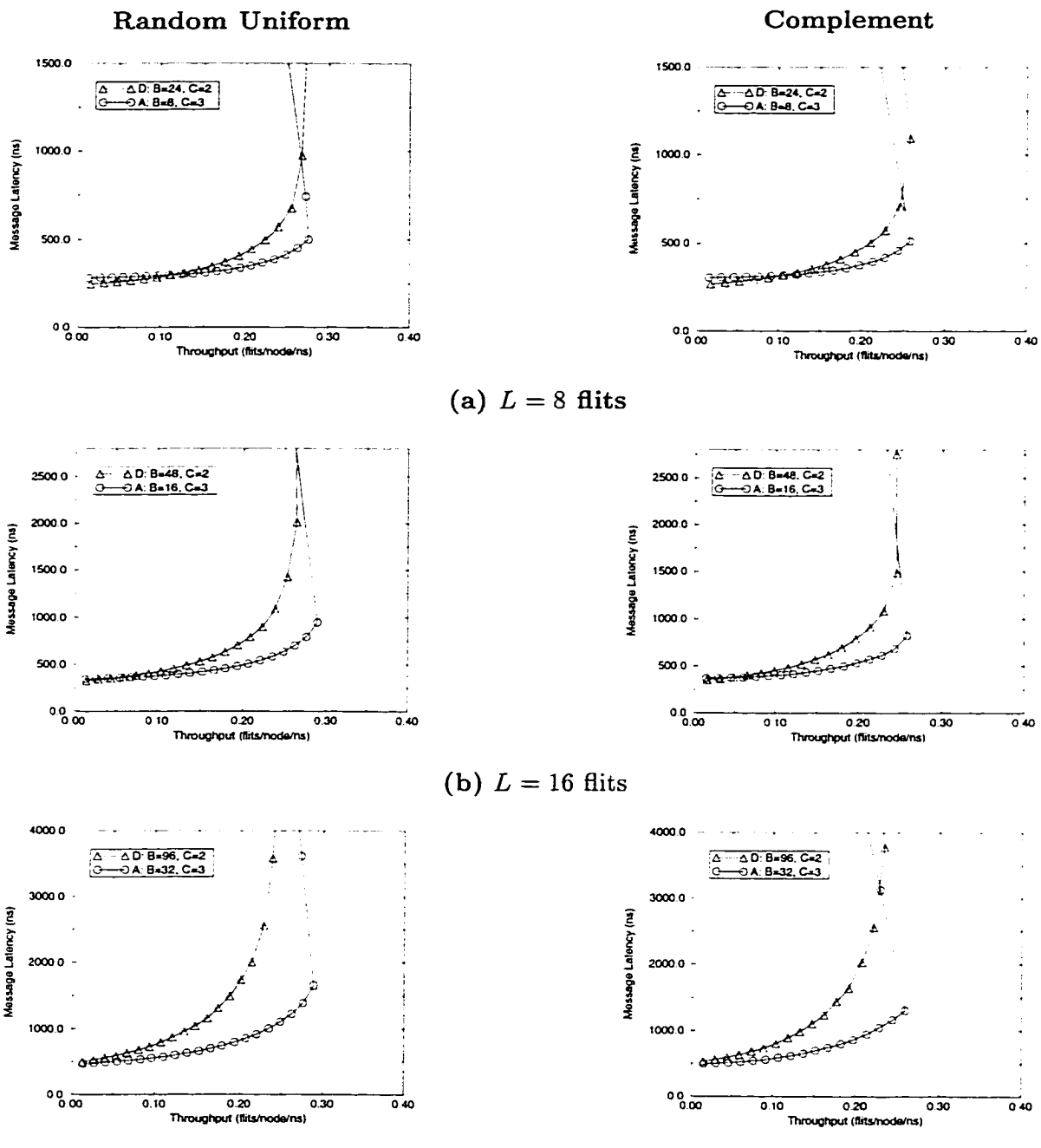
(b) Complement





(c) Perfect Shuffle

Figure 3.10: Message latency for a 10-ary 3-cube under constant buffer area (BA)

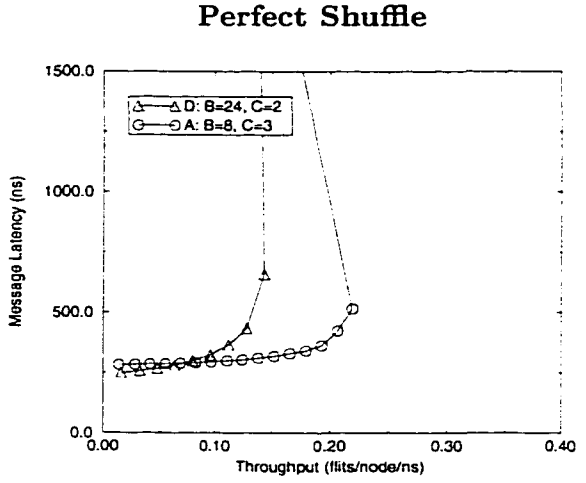


(a) $L = 8$ flits

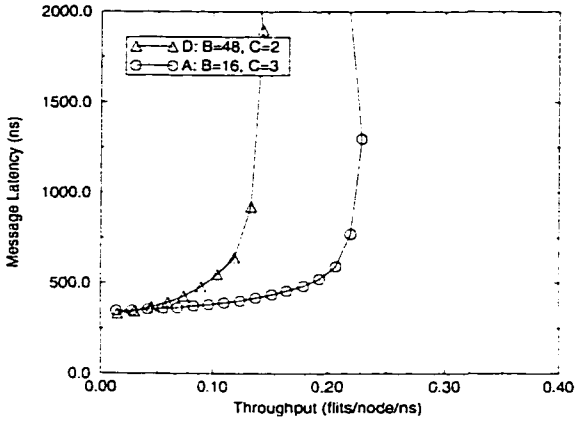
(b) $L = 16$ flits

(c) $L = 32$ flits

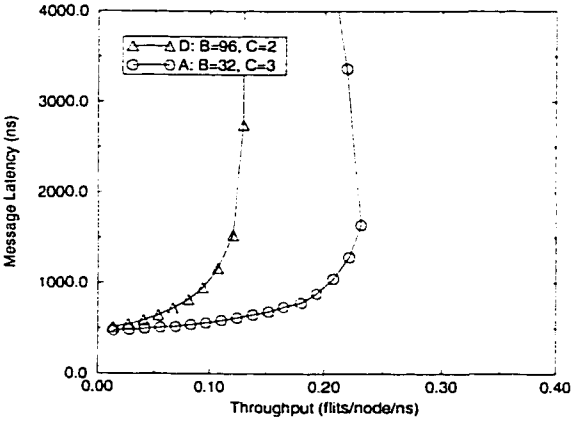
Figure 3.11: Message latency for a 10-ary 3-cube when deterministic router buffer area is twice as great as the adaptive router buffer area



(a) $L = 8$ flits



(b) $L = 16$ flits



(c) $L = 32$ flits

Figure 3.12: Message latency for a 10-ary 3-cube when deterministic router buffer area is twice as great as the adaptive router buffer area

and traffic is relatively random. As message size increases and traffic patterns become less random, adaptive routing becomes better at all throughput values. Once again this is because the increased buffer space offsets the previous low delay associated with deterministic routers.

4. Discussion of Results

The results in this section show that although adaptive routers have longer cycle times than deterministic routers, the added delay is offset at high loads by the adaptive router's ability to provide more degrees of freedom in routing. However, a tradeoff does exist between the two router models. While adaptive routers give higher saturation points deterministic routers give lower message latency at low throughput under uniform random traffic. The cross-over point, where the two latencies are equal, depends, to a large extent, on the number of VCs used in the adaptive routing, buffer size, and traffic pattern.

Among adaptive routers with various total buffer area, a tradeoff in performance exists as well. At low load, fewer VCs result in a lower message latency because the benefit of message interleaving with many VCs is not necessary at low loads and the extra delay required by many VCs is not offset by its performance. At high load, fewer VCs result in a lower saturation point.

The empirical comparison of the performance of deterministic and adaptive routers with different areas, shows that a deterministic router with twice the total buffer area of an adaptive one achieves about the same saturation point for short messages and under relatively random traffic. All of these results have shown that adaptive routers can be worth their extra complexity especially with large message sizes and under non-uniform traffic.

3.4.4 Conclusion

We have investigated the effect of router implementation on network performance by evaluating communication latency using constant area for deterministic and adaptive routers. The results indicate that a tradeoff does exist between constant area adaptive routers under non-uniform traffic patterns. At low throughput, fewer VCs result in a lower message latency. At high throughput, a fewer VCs result in a lower saturation point. However, for more uniform traffic, adaptive routers with fewer VCs always perform better.

A tradeoff between constant area adaptive and deterministic routers was also presented. Under relatively uniform traffic and small message and buffer sizes, at low throughput, deterministic routing results in lower message latency. At high throughput, a lower saturation point is obtained. However, for large message and buffer sizes, adaptive routing becomes performs better at all throughput values for all types of traffic.

Finally, comparable performance deterministic and adaptive routers was found only for small message sizes and for uniform random and complement traffic. Under these circumstances, the results show that a deterministic router has about the same saturation point as an adaptive router with half the total buffer area.

3.5 Summary of Contributions

Three efficient and practical input/output selection policies were selected out of the six policies simulated: the Round Robin, Prioritized Round Robin, and Partially Random policies. Because the Round Robin policy is not only a fair policy but also very simple to implement, it was chosen over the other two policies.

When varying buffer size for a given number of VCs, the buffer size dominates performance for small number of VCs. For a greater number of VCs, buffer size no longer dominates. When the number of VCs was increased for a given buffer size, the message

latency at low utilization also increased. At high utilization, however, just the opposite was true: more VCs meant lower message latencies and higher saturation points. These results indicate that better performance is obtained by using fewer number of VCs if low utilization is expected and more VCs for high utilization applications.

The constant bandwidth results also support the above conclusions. At low channel utilization a smaller number of VCs with large buffers produce lower latencies than a greater number of VCs with smaller buffers. However, at high utilization, just the opposite is true.

The effect of physical wire bandwidth in adaptive and deterministic routers was investigated next. It was shown that two PCs per dimension gave higher saturation points and lower message latency. However, under constant bandwidth, one PC per dimension gave better performance than two. It was also shown that this effect was more pronounced in deterministic routers than in adaptive ones. Finally, it was shown that one PC per dimension only requires one sink channel for ideal performance, while two PCs per dimension require two.

It was found that router implementation has a large impact on network performance. The results indicate that a tradeoff exists between constant area adaptive routers under non-uniform traffic patterns. At low throughput, fewer VCs result in a lower message latency. At high throughput, a fewer VCs result in a lower saturation point. However, for more uniform traffic, adaptive routers with fewer VCs always perform better.

A tradeoff between constant area adaptive and deterministic routers was also presented. Under relatively uniform traffic and small message and buffer sizes, at low throughput, deterministic routing results in lower message latency. At high throughput, a lower saturation point is obtained. However, for large message and buffer sizes, adaptive routing becomes performs better at all throughput values for all types of traffic.

Finally, comparable performance deterministic and adaptive routers was found only for small message sizes and for uniform random and complement traffic. Under these circumstances, the results show that a deterministic router has about the same saturation point as an adaptive router with half the total buffer area.

Chapter 4

HYBRID DETERMINISTIC/ADAPTIVE ROUTER

A novel routing scheme is proposed for virtual cut-through routing that attempts to combine the low routing delay of deterministic routing with the flexibility and low queuing delays of adaptive routing. The hybrid routing scheme is similar in concept to the hot potato algorithm and making the common case fast [Bar64]: a message is routed as soon as possible although the choice may not be optimal, and this routing decision is fast. The results show that the disadvantages of making a non-optimal routing decision are offset by its speed. This hybrid routing mechanism relies on pipelined implementations where different paths and stages are used for different routing modes. The experimental evaluation of this router shows that it can achieve, under most conditions, the low latency of the deterministic approach as well as the high saturation point of the adaptive one.

4.1 Deterministic and Adaptive Routing

The deterministic and adaptive routing algorithms used for comparison with the hybrid routing algorithm are the previously described dimension-order [Dal92] and Duato's (*-channels) [Dua91, Dua93, BGPS92] algorithm, respectively. The schematics of both routers are shown in Figure 3.4 with one unidirectional physical channel (PC) per dimension per node. For all routers there is only one PC for the sink channel. Once this channel is assigned to a message, it is not released until the whole message has finished its transmission. The switching model used is virtual cut-through switching and the interconnection network is

a k-ary n-cube. The model for router complexity as described previously in Section 2.2 [Chi93, AC95, DL94] is used in this chapter's results. The delay values for each of the deterministic and adaptive router algorithms simulated are shown in Tables 3.3, 3.4 and 3.5.

4.2 Hybrid Routing

This section describes the mechanism of the hybrid routing scheme along with two implementations: a pipelined hybrid router (PHR) and a super-pipelined hybrid router (S-PHR).

4.2.1 Hybrid Router Model

The hybrid router consists of three logically independent and pipelined message paths: a Fast Deterministic Path (FDP), a Slow Deterministic Path (SDP), and an Adaptive Path (AP)[1]. The routing algorithm is shown in Figure 4.1 while the pipeline stages of the router are shown in Figure 4.2 and 4.3. Note that the longest stage in *all* paths determines the maximum cycle time of the hybrid router.

The FDP has the highest priority and is used for a message flit entering on a deterministic channel that is also able to leave on a deterministic channel of the same type (low/high) and dimension. Although the choice to route deterministically first may reduce adaptivity, the routing decision and switching logic along this fast path is simpler than the traditional deterministic and adaptive routing and requires the least number of stages: $h + d$ stages for a header flit and d stages for a data flit.

If a message cannot be routed along the FDP (i.e. if a deterministic channel of the same type is not available or a message is being switched to a different type or dimension),

[1]Some physical stages are actually shared among these logically independent paths.

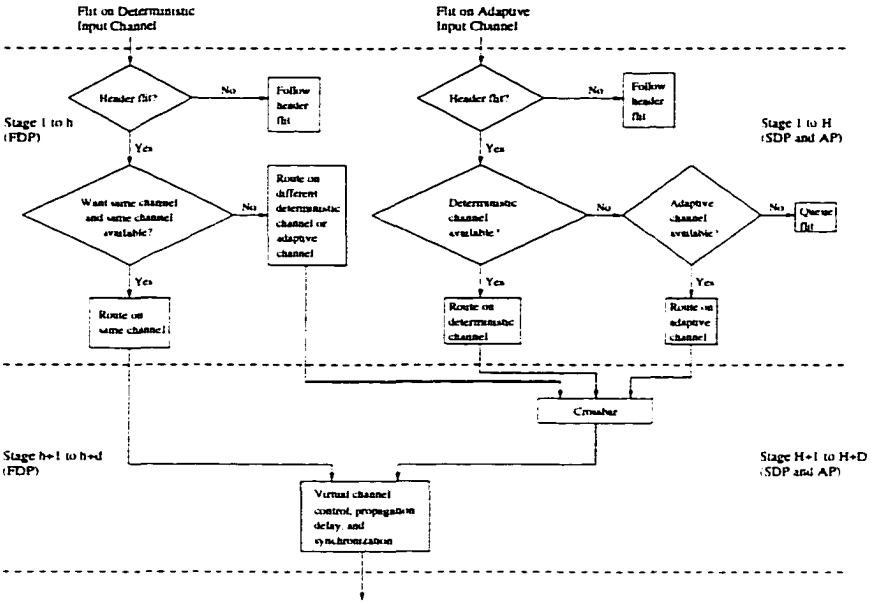


Figure 4.1: Flow chart of hybrid routing algorithm

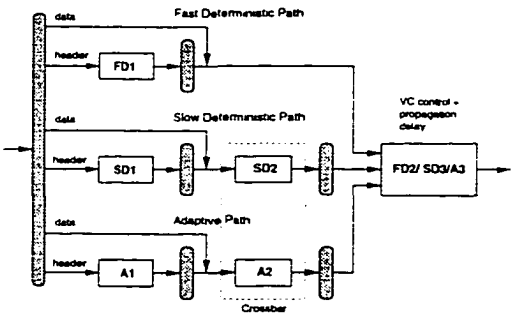


Figure 4.2: Logic schematic of Pipelined Hybrid Router.

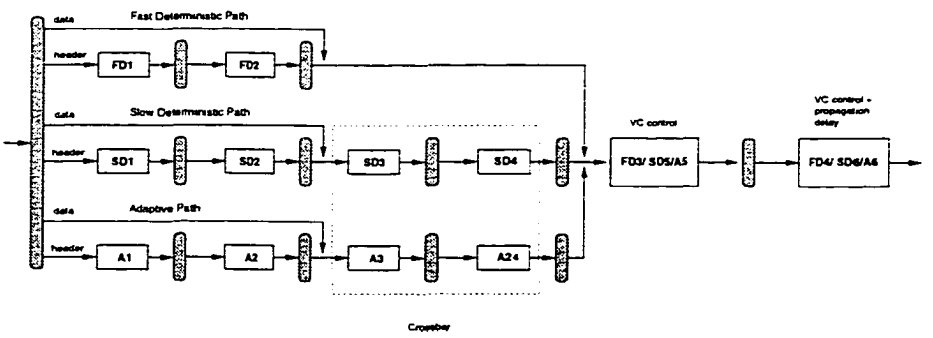


Figure 4.3: Logic schematic of Super-Pipelined Hybrid Router.

then the message is sent along the SDP which requires more logic and more stages than the FDP.

The AP is used to adaptively route a message and has the lowest priority. It is only used when both the FDP and SDP are unavailable. Both the SDP and AP take $H + D$ cycles for a header flit and D cycles for a data flit, where $(H + D) > (h + d)$. Although a header flit requires more cycles than a data flit, a data flit must always follow a header flit. Therefore, a data flit will block if the header flit has not yet advanced through a given stage.

This routing scheme is deadlock free: for any given message, the selection of paths is always a true subset of those that could be selected by the adaptive algorithm in [Dua93]. Since the adaptive algorithm has been proven deadlock free, the hybrid is also deadlock free.

Note that although a header flit requires more cycles than a data flit, a data flit must always follow a header flit. Therefore, a data flit will block if the header flit has not yet advanced through a given stage.

4.2.2 Pipelined Implementations

- The *Pipelined Hybrid Router* (PHR) implementation is shown in Figure 4.2. It uses the flow chart in Figure 4.1 where $h = d = H = 1$ and $D = 2$ and corresponds to a 2/1 stage pipeline for the FDP and a 3/2 stage pipeline for the SDP and AP. Because the routing decision and switching logic of the FDP is simplest among all the paths, the T_r and T_s delays combine into one stage ($FD1$), while the T_c delay is kept in a separate stage ($FD2$). The SDP and AP are more complex and require separate stages for each of the T_r, T_s, and T_c delays, resulting in a 3/2 stage pipeline. Note that the crossbar is physically shared between both SDP and AP and all paths share the virtual channel control logic.

- The *Super-Pipelined Hybrid Router* (S-PHR) relies on deep pipelines to implement the hybrid router. Using deep pipelines can increase overall throughput at the cost of additional latch delays. Also the clock skew becomes more prominent: if the clock cycle becomes as small as the sum of the clock skew and latch overhead, further pipelining is no longer useful. Other factors to consider is an efficient use of the pipeline stages. Since the stages in all three paths are efficiently used in the PHR, the work in each stage of the PHR is divided into two stages in the S-PHR. Therefore, the 2/1 stage pipeline in the FDP of the PHR becomes a 4/2 stage pipeline in the S-PHR, while the SDP and AP paths are modified from 3/2 stage pipelines to 6/4 stage pipelines. Once again, the crossbar is physically shared between both SDP and AP and all paths share the virtual channel control logic. Figure 4.3 shows the new schematic for this super pipeline. Note that the main difference between the PHR and S-PHR is the number of stages required for each path. The flow chart in Figure 4.1 is used in the S-PHR where $h = d = H = 2$ and $D = 4$.

4.2.3 Clock Cycle Times

The performance of the pipelined and super-pipelined implementations of the hybrid router is compared to the corresponding implementations of both the deterministic and adaptive routers.

Pipelined Router Implementation. Both the deterministic and adaptive routers are implemented as a 3/2 stage pipeline, where 3 stages are required for a header flit and 2 stages are required for a data flit. The cycle times for both are obtained using the equations in Section 2.3. Note that the deterministic router is not implemented using a 2/1 stage pipeline as in the hybrid router. This is because to accommodate both the routing and switching delays for the Fast Deterministic Path into one stage would require the deter-

ministic router's cycle time to be comparable to that of the adaptive router's. This greater cycle time would offset any advantage gained from having fewer number of cycles.

In the more complex hybrid router, the cycle time for its 3/2 stage pipeline path is much larger than that for the 3/2 stage pipeline in the deterministic router. Therefore all the necessary logic in the FDP of the hybrid router can fit into a 2/1 stage pipeline implementation without greatly increasing the cycle time of its 3/2 stage pipeline paths (SDP and AP paths). Since the cycle time is not greatly increased by adding the 2/1 stage pipeline path (FDP), the advantage of fewer number of cycles is retained. The cycle time of the pipelined hybrid router (PHR) is simply one gate delay larger than that of the adaptive router to account for the increased critical path length due to the inclusion of the 2/1 stage pipeline path (FDP).

Super-Pipelined Router Implementation. The super-pipelined implementation for the routers consists of dividing the work in each stage of its corresponding pipelined implementation into two. This results in a 6/4 stage super-pipeline for the adaptive router and a 4/2 stage super-pipeline for the FDP and a 6/4 stage super-pipeline for the SDP and AP of the hybrid router. The deterministic router's super-pipelined implementation is a special case and is discussed later.

The routing and switching delays for the *super-pipelined* implementation of all the routers is represented by T_R and T_S, respectively. These delays were calculated using their corresponding *pipelined* delays (T_r and T_s) in Equation 4.1. Note that the T_R and T_S delays are represented as T_{S-PR} in Equation 4.1, while their corresponding T_r and T_s delays are represented by T_{PR}. The setup time for the latch (L) is 0.8 ns and the delay for one gate (G) is 0.6 ns.

$$T_{S-PR} = \left\lceil \frac{(T_{PR} - L)}{2.0 * G} \right\rceil * G + L \tag{4.1}$$

The T_R and T_S delays involve subtracting the latch setup delay to obtain the combinational logic delay which is then split in two in the super-pipelined router. The number of integer gate delays is then calculated and the latch setup time is added back.

Since the T_c delay (pipelined channel delay) consists of a set of gates as well as a wire, this case is considered separately. The two stages for the super-pipelined implementation of this delay consist of the VC controller delay ($T_{VC} = 1.24 + 0.6log_2C$) in one stage and the propagation delay which is required for transferring a flit across the physical wire ($T_P = 4.9$) in the other stage.

The cycle time for the super-pipelined router (CC_{S-PR}) is then determined by the longest delay among all super-pipelined stages.

$$CC_{S-PR} = Max(T_R, T_S, T_{VC}, T_P) \tag{4.2}$$

As previously mentioned, the deterministic router's super-pipelined implementation is a special case. When the above equations are used for the deterministic router, T_P is found to be the bottleneck. Since the deterministic router's T_r delay of 4.7 ns is already smaller than the T_P delay of 4.9 ns, the super-pipelined routing delay (T_R) is the same as that of the pipelined routing delay (T_r). All other delays are calculated using the above equations which result in a 5/4 stage super-pipeline. The delays for this super-pipeline are found in Table 2a.

A further reduction in the number of stages in the deterministic router can be performed without greatly increasing the overall router cycle time. This is because T_{VC} is quite small as seen in Table 2a and its delay can be combined with the two super-pipelined switching stages (T_{S1} and T_{S2}). This results in a 4/3 stage super-pipeline. These delays are

B	T_R	T_{S1}	T_{S2}	T_{VC}	T_P	CC Period
16	4.70	3.2	3.2	2.64	4.9	4.9
32	4.70	3.8	3.8	2.64	4.9	4.9
96	4.70	4.4	4.4	2.64	4.9	4.9

a - 5/4 stage pipeline

B	T_R	T_{S1}	T_{S2}	T_{VC}	T_P	CC Period
16	4.70	4.4	4.4	-	4.9	4.9
32	4.70	4.4	4.4	-	4.9	4.9
96	4.70	5.0	5.0	-	4.9	5.0

b - 4/3 stage pipeline

Table 4.1: Super-pipelined deterministic router delays ($C = 2$, $P = 3$, $F = 1$) for k-ary 3-cube networks (in *nsec*) for two different super-pipelines

	Deterministic		Adaptive		Hybrid	
B	PR	$S-PR$	PR	$S-PR$	PR	$S-PR$
8	6.74	4.90	7.80	4.90	8.40	5.00
16	6.74	4.90	7.80	4.90	8.40	5.00
32	6.74	4.90	7.80	4.90	8.40	5.00

Table 4.2: Clock cycle times for all three routers (in *nsec*) for k-ary 3-cube networks

shown in Table 2b. This 4/3 stage super-pipeline was used for all the deterministic router super-pipeline simulations.

The cycle times for the pipelined and super-pipelined implementations of the three routers are shown in Table 4.2.

4.3 Experimental Evaluation

Simulations of the deterministic, adaptive and hybrid routing implementations were performed using a discrete-time simulator on an 8-ary 3-cube network. The simulations use a stabilization threshold of a 0.005 difference between traffic 1000 clock cycles apart to determine steady state. Message sizes varied from 8 to 32 flits and traffic from 0.1 until saturation was reached in 0.1 increments. The buffer sizes used in the simulation are all equal to a single message length. The adaptive router and the adaptive path in the hybrid

router use three VCs per dimension. The deterministic router and the deterministic path in the hybrid router use two. The simulator implements a back-pressure mechanism which results in a negative slope of the latency versus accepted traffic plots at higher loads. The following five different traffic patterns were simulated: random uniform complement, perfect shuffle, bit-reversal, and butterfly.

4.3.1 Performance of Hybrid Routing

Figure 4.4 shows the message latencies as well as the accepted load versus offered load plots of the deterministic, adaptive and hybrid pipelined implementations under random uniform traffic and message lengths of 8, 16, and 32 flits. Figure 4.5 shows similar plots for the remaining four traffic patterns and message size of 16 flits.

Message Latency. Under random uniform traffic, for small messages (8 flits) the latency of the Pipelined Hybrid Router is not only lower than the pipelined adaptive one but is also lower than the pipelined deterministic one at low traffic. This is due to the fact that the Pipelined Hybrid Router has a 2/1 stage pipeline for header/data flits, while the deterministic router has a 3/2 stage pipeline. Even though the delay per stage in the deterministic router is shorter than the Pipelined Hybrid Router's, the greater number of stages dominates. For medium messages (16 flits) the latency of the Pipelined Hybrid Router is very close to that of the deterministic one at low traffic and follows the adaptive one at higher traffic. For larger messages (32 flits) the Pipelined Hybrid Router latency is less than the adaptive one at low traffic and greater than the adaptive one at high traffic. In general, the latency of the Pipelined Hybrid Router follows the deterministic one at low traffic and the adaptive one at high traffic.

Effects of Message Length. As message size increases under random uniform traffic, the performance advantage of the PHR decreases compared to the deterministic and adaptive

pipelined routers. This is due to the facts that more messages, and therefore headers, are needed to achieve the same utilization with short message length and the PHR has a performance advantage for header flits, especially at low utilization. While the deterministic router has a 3-stage header flit pipeline with a low cycle time, the PHR has a 2-stage deterministic header flit pipeline with a higher cycle time. Since the number of pipeline stages dominates performance (and not the cycle time), the performance difference between the routers is greater for small message sizes than for large message sizes. This difference also exists at high traffic, although it's much smaller due to the fact that more message blocking occurs covering up differences in header flit time.

Effects of Traffic Patterns. The performance of the PHR under all non-random traffic patterns is similar to that for random uniform traffic. Once again, the PHR performs best at low traffic, while the adaptive router performs slightly better at high traffic. This is due to the higher priority given to the deterministic paths in the PHR: less choices are available as a message is routed through the network on deterministic channels.

Saturation Point. Under random uniform traffic, the saturation point of the PHR is, in all cases, much higher than that of the pipelined deterministic router and is very close to the adaptive one. One reason for the slight decrease in saturation point for the PHR with respect to the adaptive router, is that messages are routed onto the deterministic channels first, reducing the number of options available to a message later on. As traffic increases, this effect causes more blocking and slightly smaller saturation points. Under all non-random traffic the PHR's saturation point is once again much higher than that of the pipelined deterministic router and is very close to the adaptive one.

4.3.2 Effects of Super-Pipelining

The effects of super-pipelining on message latency are shown in Figures 4.6 and 4.7. Under all traffic patterns, the super-pipelined implementations for all routers achieve better overall performance gain than the pipelined implementations. This is due to the higher throughput that is achieved by deeper pipelines. Because of the higher throughput, all super-pipelined routers achieve higher saturation points than the pipelined implementations.

4.3.3 Effects of Path Priorities

The hybrid router's implementation for all the previous results includes first routing on the FDP, then on the SDP, and finally on the AP. This scenario is referred to as the SDP scenario. However, by routing the AP last, adaptivity that could be utilized at high loads may be lost. Therefore, simulations were performed to see if switching the priorities of the AP and SDP would improve performance near saturation. In this scenario, the FDP is still given highest priority. However, the AP is given the next highest priority, followed by the SDP. This scenario is called the AP scenario.

The simulated results are shown in Figure 4.8. The results between the two scenarios are so close that although the AP scenario may allow more routing choices as load increases, the SDP scenario performs equally well because of the high priority and low cycle time of the FDP. Since the FDP has the highest priority, the benefit of retaining messages on deterministic channels allows the FDP path to be utilized more often and offsets any adaptivity loss.

4.4 Summary of Contributions

This chapter reports on the empirical evaluation of a hybrid routing scheme which combines the low router delay of deterministic routing with the flexibility and low queuing delays of adaptive routing. This hybrid routing mechanism is realized using two different

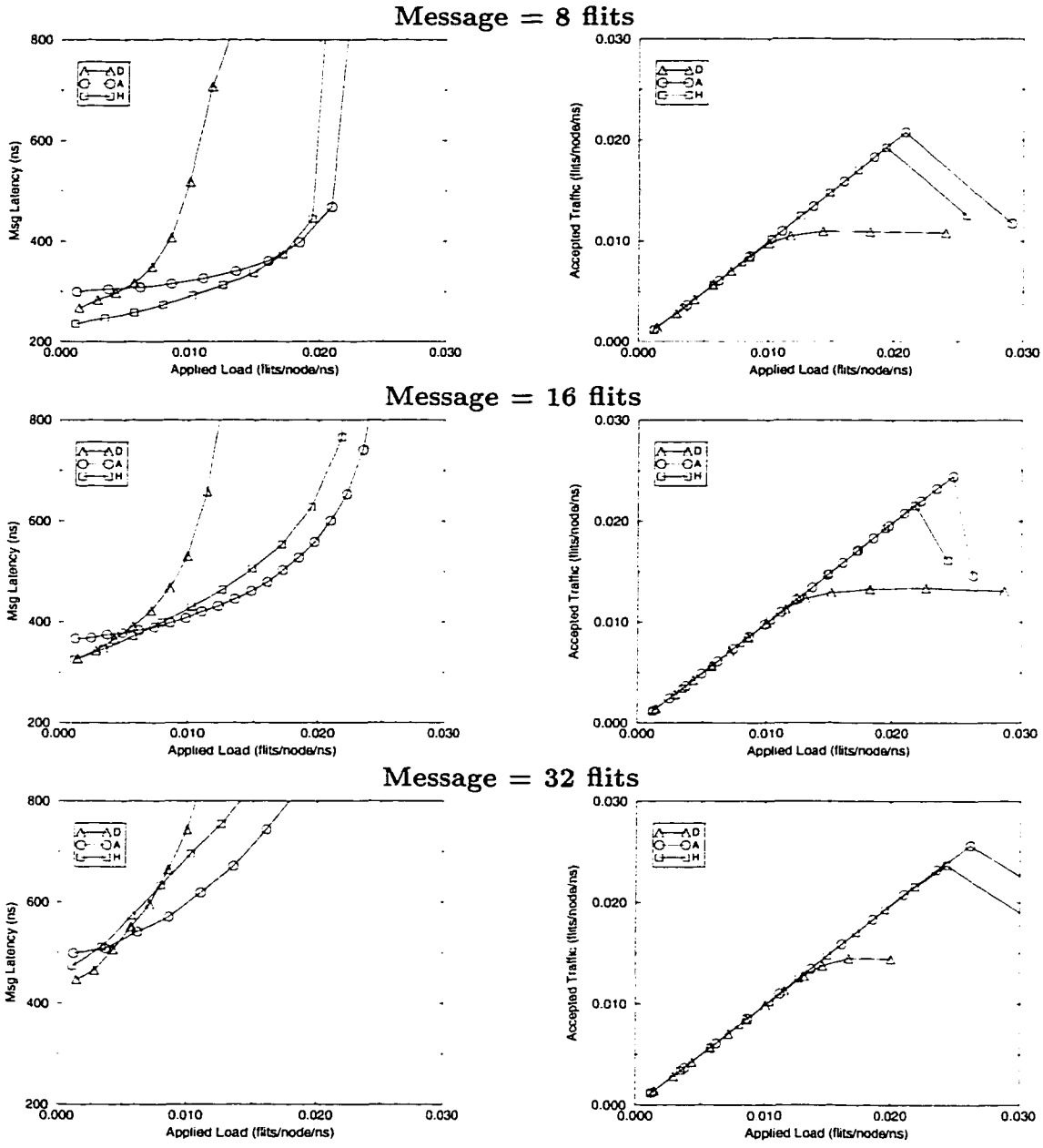


Figure 4.4: Deterministic, adaptive and hybrid pipelined implementation routers in an 8-ary 3-cube under random uniform traffic.

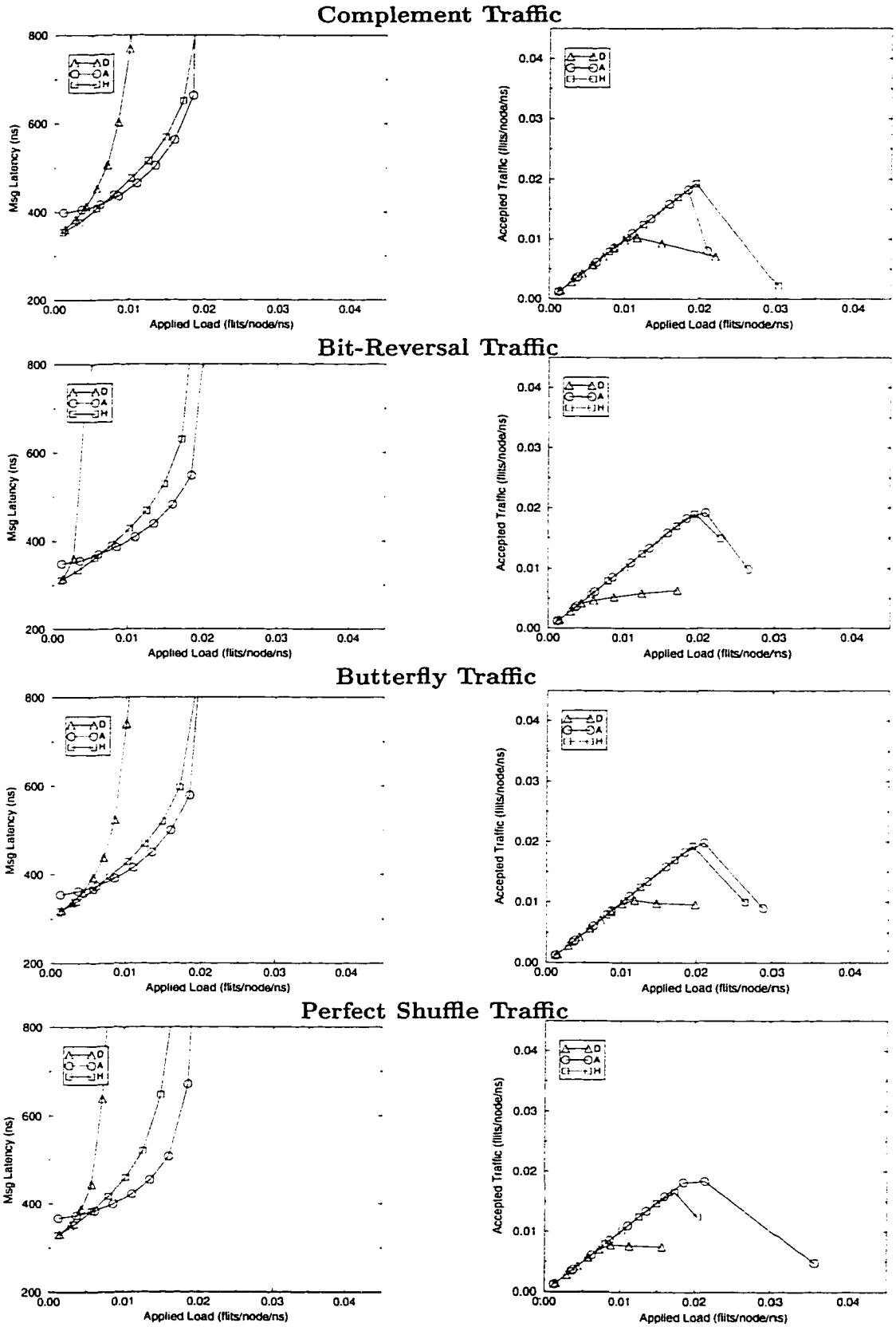


Figure 4.5: Pipelined implementation of deterministic, adaptive and hybrid routers for 8-ary 3-cube with message size of 16 flits

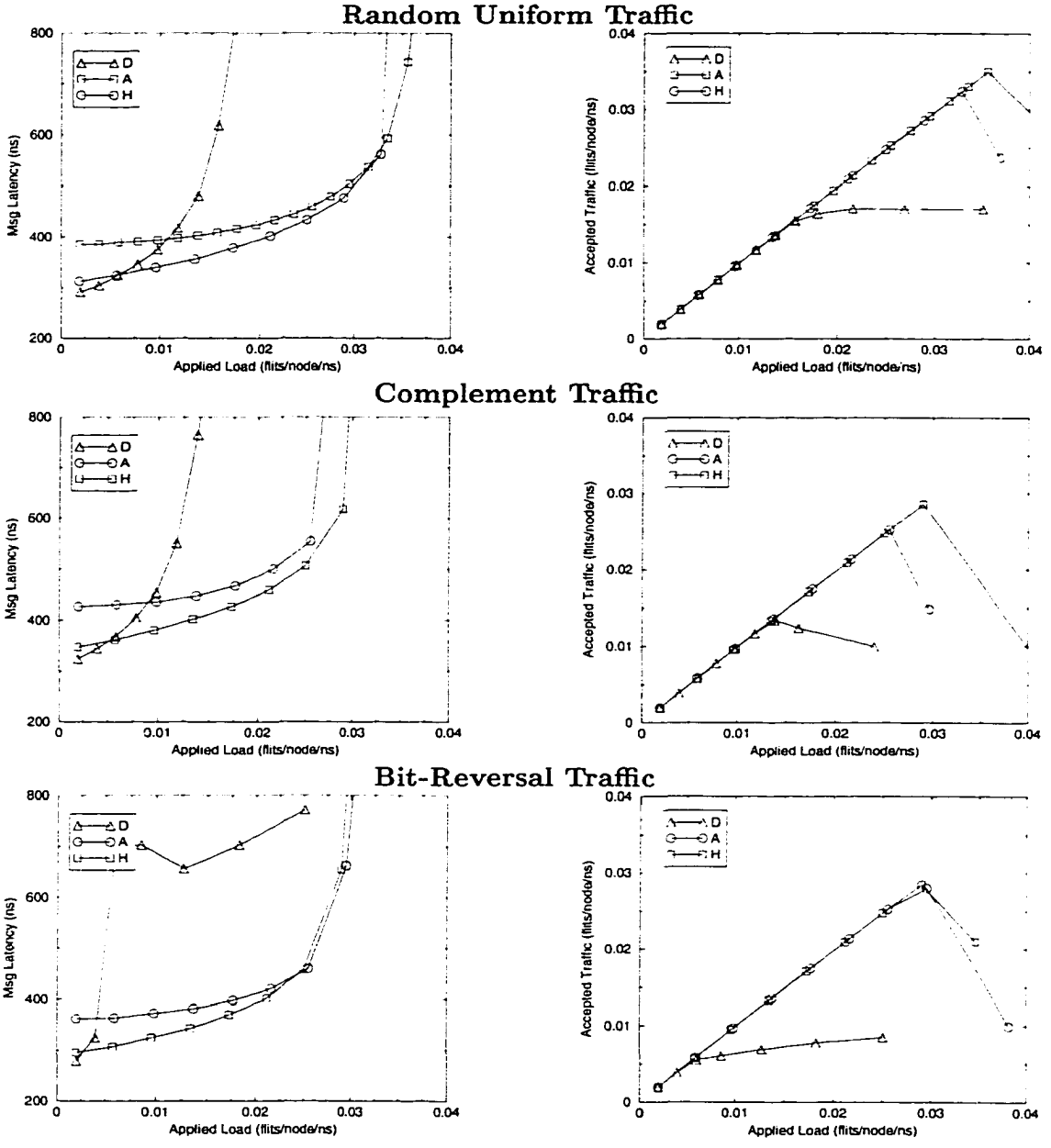


Figure 4.6: Super-pipelined implementation of deterministic, adaptive and hybrid routers for 8-ary 3-cube with message size of 16 flits

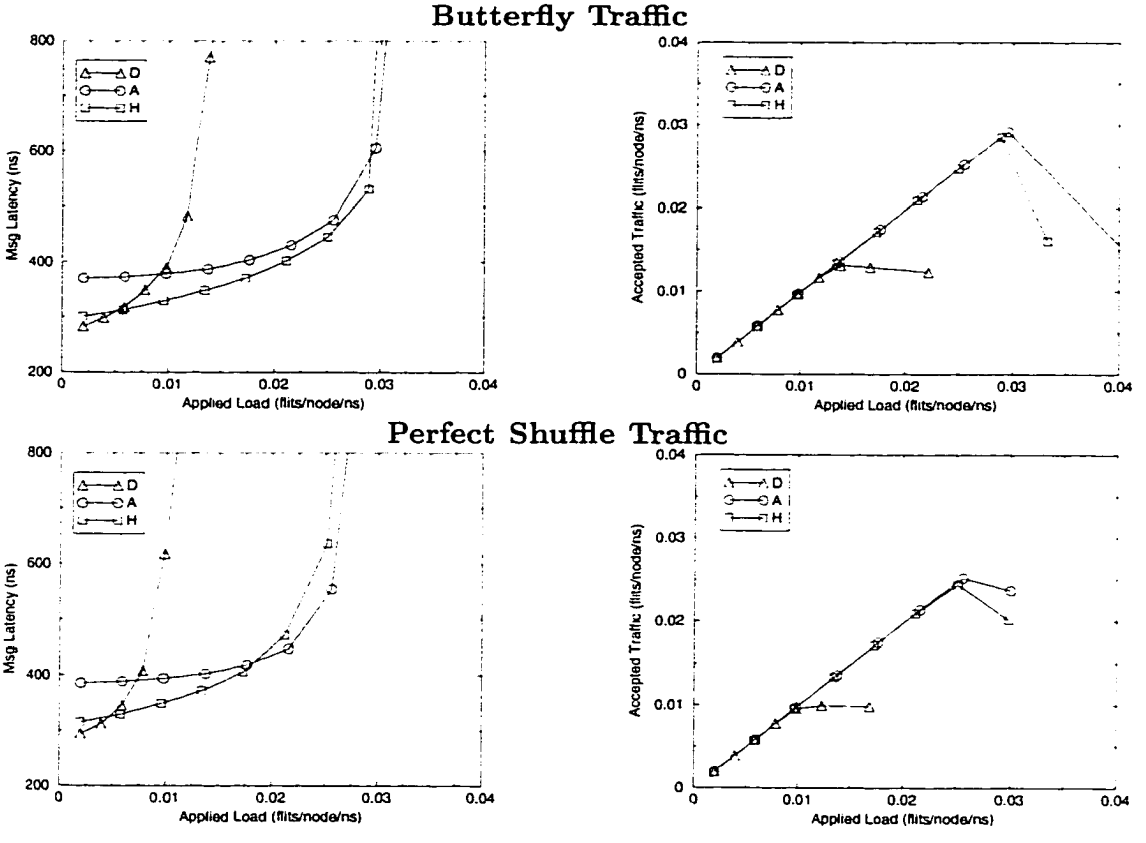


Figure 4.7: Super-pipelined implementation of deterministic, adaptive and hybrid routers for 8-ary 3-cube with message size of 16 flits

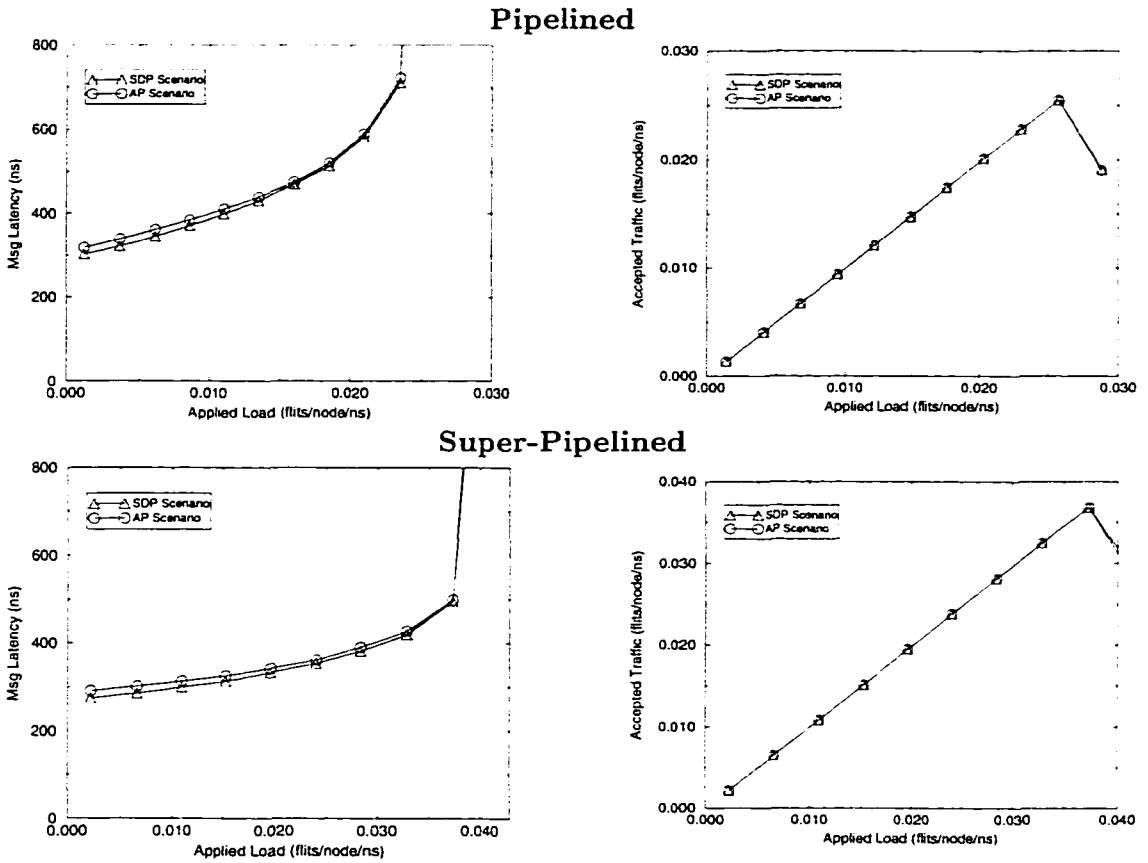


Figure 4.8: Comparison of SDP and AP Scenarios for an 8-ary 3-cube network under random uniform traffic for message size of 16 flits.

implementations (PHR and S-PHR) in which different paths and stages of the router are used for different routing modes. The scheme also relies on making the "common case fast" and is similar in concept to the hot potato algorithm.

The results from the simulation evaluation of this scheme show that both implementations of the hybrid router do achieve their objectives: a message latency comparable to that of the deterministic router at low traffic and a saturation point close to that of the adaptive router at high traffic. In addition, deeper pipelines achieve better overall performance gain than the pipelined implementations.

Chapter 5

MULTICAST COMMUNICATION

Multicast communication is a common operation in parallel computing and efficient implementation of multicast is critical to the performance of multiprocessors. Software-based approaches for implementing multicast can result in high message latencies. Hardware-based schemes can greatly improve performance. Deadlock freedom in multicast communication is difficult to achieve resulting in more involved routing algorithms and higher startup delays. Hardware tree-based algorithms do not require these high startup delays but do suffer from high probabilities of message blocking leading to poor performance.

In this chapter, a hardware tree-based routing algorithm (HTA) is proposed for multicast communication under virtual cut-through switching in k-ary n-cubes. Simulation results are compared against several commonly used multicast routing algorithms and show that HTA performs extremely well under many conditions.

5.1 Hardware Tree-Based Multicast Routing Algorithm

HTA is a tree-based routing scheme for multicast communication designed to keep the probability of message blocking at each intermediate node along a message's path low. By keeping the blocking probability low, the probability of deadlock is reduced. The algorithm is fully compatible with existing unicast routing schemes and relies on the deadlock detection and recovery algorithm described in Section 5.1.3. The main characteristics of HTA are:

- Virtual cut-through switching is used with distinct virtual paths for unicast and multicast messages, each path using three VCs per dimension (total of six VCs per dimension).
- Unicast messages are routed using the deadlock-free routing algorithm proposed in [Dua93, BGPS92] (see Section 5.1.1).
- Multicast messages are routed using fully adaptive routing along with a deadlock detection and recovery scheme (see Section 5.1.2).
- Delayed header flit routing is implemented for multicast (see Section 5.1.1).

In this algorithm, each multicast message is composed of all header flits followed by all data flits. Each header flit holds one destination address and destinations do not need to be ordered within the multicast message. As messages move throughout the network, all header flits are routed using the tree-based routing scheme described below. Once all header flits are routed, the data flits are moved simultaneously to all channels allocated to the header flits in the message. HTA's routing algorithm is shown in Figure 5.1 and is explained in the following sections.

5.1.1 The Routing Scheme

The deadlock-free routing algorithm proposed in [Dua93, BGPS92] is used for unicast communication in HTA and is an adaptive routing algorithm based on dimension-order routing. In this adaptive routing scheme, a message is routed on any adaptive channel until it is blocked. Once blocked, a message is routed using dimension-order routing if possible. A message may return to the adaptive channels in the following routing decisions if the adaptive channels are available.

When a message is routed using dimension-order routing, it is routed along decreasing dimensions with a dimension decrease occurring only when zero hops remain in all higher

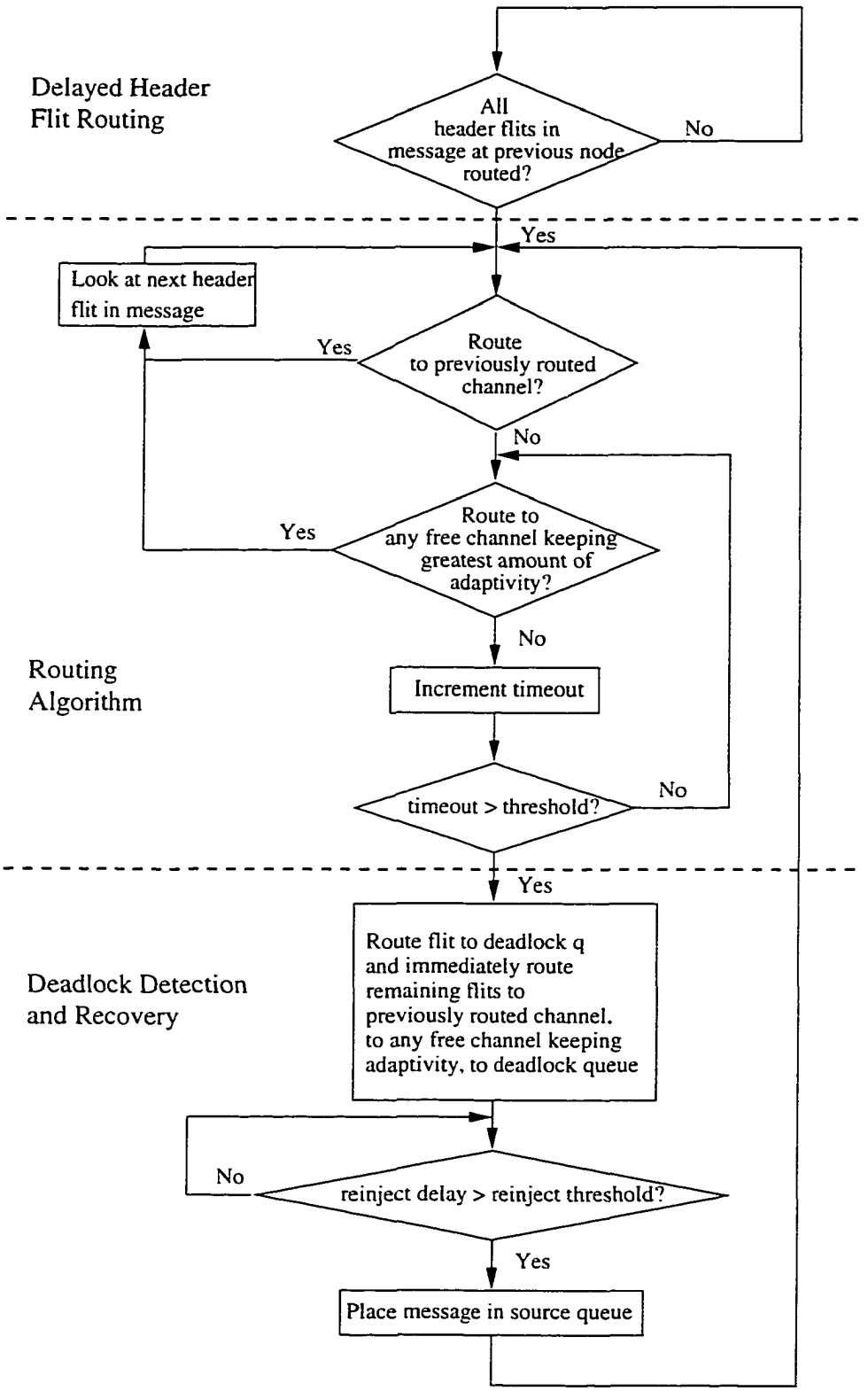


Figure 5.1: HTA Routing Algorithm

dimensions. By assigning an order to the network dimensions, no cycle exists in the channel-dependency graph and the algorithm is deadlock-free.

A minimum of three VCs per dimension is required for deadlock-free routing in k-ary n-cubes. Two VCs per dimension are used for dimension-order routing and all remaining VCs are used for adaptive routing.

Multicast communication uses the proposed tree-based deadlock detection and recovery (see Section 5.1.3) and delayed header flit routing scheme (explained below). Since tree-based routing has a high probability of message blocking, minimizing the hold and wait on VCs is of greater importance than maximizing the adaptivity of a message. Therefore, as each header flit reaches the top of the queue, it is routed first to any channel already allocated to this message by a preceding header flit in this message. Then it is routed to any channel, starting with all channels in the dimension that has the greatest number of hops left in it, followed by the next dimension with the greatest number of hops left, and so on. After the header flit is routed at the current node, it is moved to the neighboring node's queue.

Immediate Header Flit Routing: When a header flit is moved to a neighboring queue, it is usually routed at this neighboring node without waiting for the remaining header flits in the message to be routed. This scheme is most commonly used [LMN91, MDT96, BND87] and we refer to it as *immediate header flit routing.*

Figure 5.2 shows an example of *immediate header flit routing.* In this figure, header flit $A.2$ is blocked while header flit $A.1$ continues to be routed, holding VCs and causing message B to block.

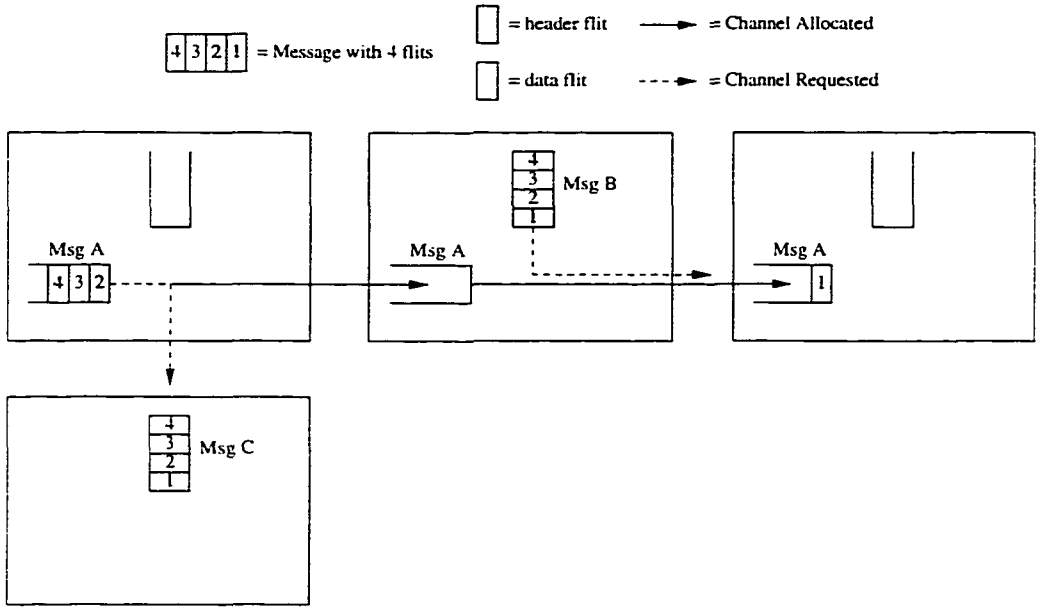


Figure 5.2: Immediate header flit routing: Header flit $A.2$ is blocked while header flit $A.1$ continues to be routed, holding VCs and causing message B to block.

Delayed Header Flit Routing: HTA uses *delayed header flit routing* to lower the probability that a message will block. In *delayed header flit routing*, a header flit at a neighboring node is not allowed to be routed until all header flits at the current node have been routed.

Because in tree-based routing, the remaining header flits may not be immediately routed, it's more advantageous to keep all header flits within close proximity of one another using delayed header flit routing. This close proximity prevents header flits from being assigned to queues at downstream nodes before all flits in the message can use them. This keeps the downstream queues free so that they are available for *other* messages in the network which can use them immediately.

Figure 5.3 shows an example of *delayed header flit routing*. In this figure, header flit $A.2$ is blocked while header flit $A.1$ waits at the neighbor node until the last header flit in this message (flit $A.2$) is routed. Message B is now able to be routed. As the number of destinations grows, delayed header flit routing becomes increasing important in reducing the probability of message blocking.

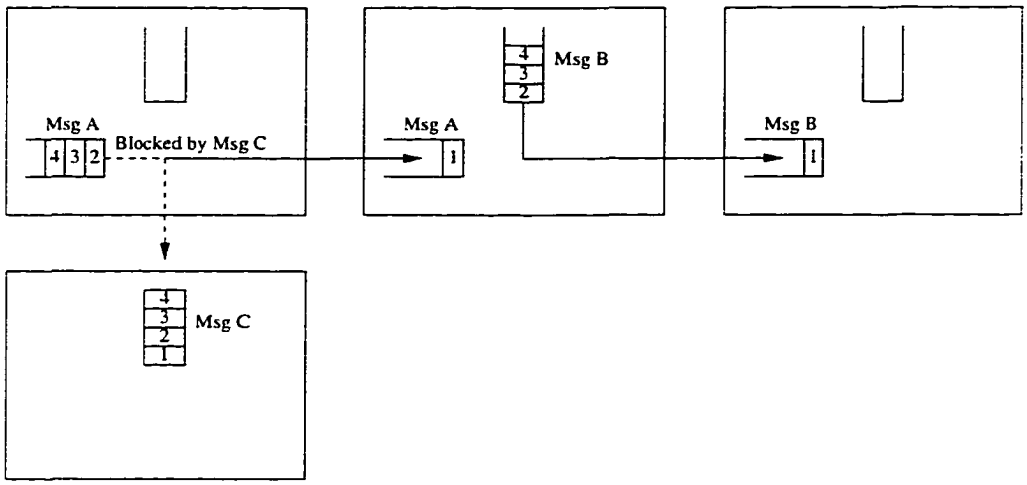


Figure 5.3: Delayed header flit routing: Header flit $A.2$ is blocked while header flit $A.1$ waits to be routed at the neighbor node until the last header flit in this message (flit $A.2$) is routed. Message B is now able to be routed.

5.1.2 Deadlock Recovery Mechanism

In HTA, each node has a dedicated holding queue called the deadlock queue. If a header flit currently under consideration cannot be routed to a channel in a predetermined amount of time (timeout delay), the header flit is considered to be in a potential deadlock situation and is routed to the deadlock queue at the current node.

Once one of the header flits in a message has been assigned to the deadlock queue, all remaining header flits in the message must be routed as soon as they reach the top of the queue. If any remaining header flit cannot be immediately routed, it is also considered to be in a potential deadlock and is routed to the deadlock queue at the current node. Messages in the deadlock queue are reinjected into the network after a predetermined amount of time (reinjection delay).

Figure 5.4 shows an example of a potentially deadlocked situation. Flit $A.1$ has been routed to a neighboring queue in the X dimension. Flit $A.2$ has not been routed in T number of cycles (timeout delay) and has been routed to the deadlock queue at the current node. Since this flit is potentially deadlocked, all remaining header flits in this message (Flit

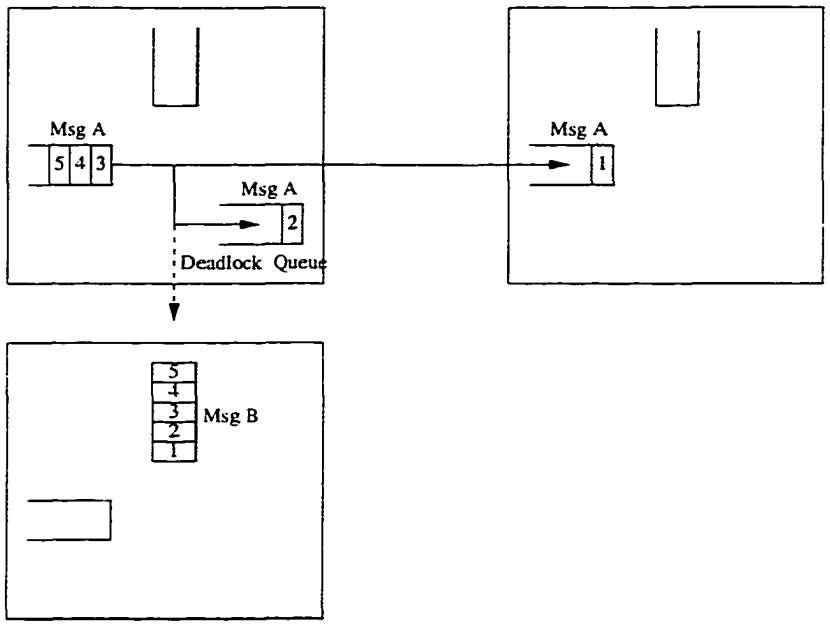


Figure 5.4: Potentially Deadlocked Situation: Flit $A.1$ has been routed to a neighboring queue. Flit $A.2$ has timed-out and been routed to the deadlock queue, requiring flit $A.3$ to be immediately routed. Since flit $A.3$ requests the queue occupied by *Message B*, it must also be routed to the deadlock queue behind flit $A.2$.

$A.3$) must now be immediately routed. Since Flit $A.3$ requests the queue already occupied by *Message B*, this flit cannot be immediately routed to a free channel and must also be routed to the deadlock queue. After a predetermined amount of time (reinjection delay), the message that was routed to the deadlock queue is moved to the source queue at the current node and reinjected into the network.

When the deadlock queue is full and another message is potentially deadlocked, an interrupt is generated and the message is absorbed into the current node. When space is available in the router's deadlock queue, the message is prefetched from the local processing node and moved to the deadlock queue in the router. By allowing the overflow of messages to be stored in the local processing node, this deadlock queue essentially becomes infinite for all practical purposes without causing any additional delay in routing.

5.1.3 HTA Router Implementation

The HTA router implementation is shown in Figure 5.5 and uses one *unidirectional physical channel* (PC) per dimension per node. Six VCs are multiplexed over one PC, with three of the VCs dedicated to unicast and three dedicated to multicast. At least one sink channel is used for each node. Once a sink channel is assigned to a message, it is not released until the whole message has finished its transmission.

Storage buffers are associated with IP channels, requiring the routing decision to be made after buffering the message. A crossbar is used to connect input buffers to output buffers allowing the transfer of data flits to multiple VCs.

5.2 Experimental Evaluation

Simulations were performed and compared for the Software Multicast, Column-Path, and HTA algorithms. A discrete-time simulator was used for 8-ary 2-cube and 16-ary 2-cube networks. Message sizes varied from 16 to 64 flits and number of destinations per multicast message were randomly chosen and varied from 8 to 32. The buffer sizes used in the simulation are all equal to a single message length. All router implementations use six VCs per dimension. The Software Multicast and Column-Path algorithms both use two VCs per dimension for deterministic routing and four for adaptive. The timeout and reinjection delays for all message sizes simulated here for HTA are 16 and 50 clock cycles, respectively.

The communication startup time required for ordering the messages in the Column-Path algorithm is *not* included in the simulations. The time required for creating and placing the messages in the source queue is also *not* included for any of the routing algorithms simulated here. The simulations use a stabilization threshold of a 0.005 difference between traffic 1000 clock cycles apart to determine steady state. Traffic was varied from 0.1 until

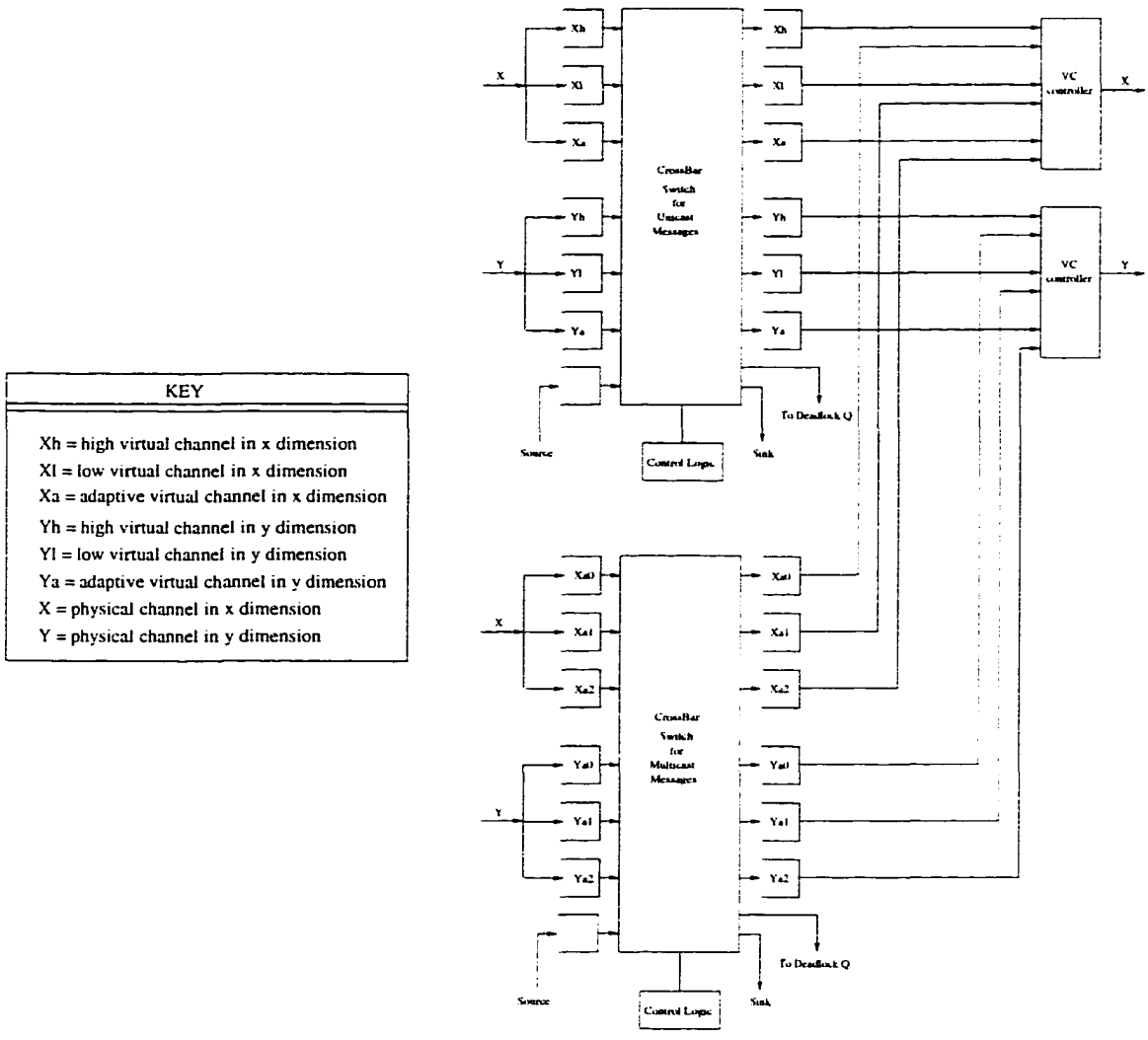


Figure 5.5: Schematic of 2D router for HTA

saturation was reached in 0.1 increments. Simulations were performed for traffic composed of only multicast communication, only unicast communication, and half unicast and half multicast communication.

The simulator implements a back-pressure mechanism for all routing implementations in which only three unicast messages and one multicast message are allowed in the source queue at one time. This back-pressure mechanism sometimes results in a fairly flat curve near saturation in some of the latency versus target traffic graphs.

All implementations use 12 sink channels. Although this is an unusually high number of consumption channels, the Column-Path routing algorithm requires this many channels for deadlock freedom since the adaptive routing algorithm proposed in [Dua93, BGPS92] is used for the base routing conformed path (as opposed to e-cube routing where less number of sink channels are required). For fairness, the Software Multicast and HTA are also simulated with 12 sink channels, although both only require one sink channel.

5.2.1 Deadlock Probability

No exhaustive study has been done on all the factors that affect deadlock probability. However, some of the main factors which affect deadlock include:

Timeout Delay: Timeout delay is a critical factor in HTA's deadlock detection and recovery scheme. Too small of a timeout will result in many false deadlocks, while too large of a timeout will result in delayed detection of true deadlocks leading to poor performance. Choices for timeout values include timeouts equivalent to the size of the message [KLC94], four times the message length [MLDP97], 8-16 cycles [VP95a], and 800-1000 cycles [PDLM97].

Tree-based multicast communication timeout requirements are slightly different than those for unicast communication. In tree-based multicast, more than one header flit is

usually routed at each node. If the timeout is too great for each header flit, message progress through the network will be very slow since data flits are not forwarded until all header flits are routed. If the timeout is too small, many false deadlocks will result. HTA uses 16 cycles for all message sizes and number of destinations per message.

Reinjection Delay: Reinjection delay is not as critical as the timeout delay. However, it must be large enough that messages are reinjected into the network after congestion has been alleviated but small enough to keep message latency low.

Reinjection delays for unicast communication on wormhole switching under k-ary n-cube networks are around 200 cycles [PDLM97, MLDP97]. Deadlock detection and recovery techniques similar to DISHA [VP95a] do not require reinjection delays because when messages are potentially deadlocked they are immediately routed using "floating buffers" to their destination. HTA uses a reinjection delay of 50 cycles. Fifty cycles is a feasible delay because the deadlock detection and recovery scheme is not a software-based approach. Instead, the deadlock queue that holds potentially deadlocked messages in HTA is located in the router.

Injection Limitation Schemes: Deadlock probability greatly increases near saturation [KLC94, PW97, WP97]. Injection limitation schemes have been implemented to reduce this large probability of deadlock. Limitation schemes include limiting injection into the network based on the number of free virtual channels at the current node [LMDP97, PDLM97, MLDP97]. In HTA, message injection is limited to how many messages are in the source queue. Three unicast messages and one multicast message are allowed simultaneously. This backpressure mechanism includes messages that have potentially deadlocked and have been reinjected into the network using the source queue.

Directionality: It was shown in [WP97] that bidirectional networks have a lower deadlock probability than unidirectional networks. Deadlock recovery schemes using bidirectional channels include [KLC94, PDLM97, WP97, MLDP97]. Unidirectional channels are addressed in [WP97, FR97]. HTA uses unidirectional channels.

Switching Type: Virtual cut-through switching can greatly decrease the probability of deadlock over wormhole switching when a fixed number of virtual channels are used [WP97]. Wormhole switching is used in [KLC94, PDLM97, WP97, MLDP97, VP95b] and virtual cut-through switching is used in [WP97, PDLM97]. HTA uses virtual cut-through switching.

Number of Virtual Channels: Increasing the number of VCs increases routing freedom which in turn exponentially decreases the probability of deadlock [WP97, PW97]. The number of VCs used per dimension include one VC [FR97], one to two VCs [KLC94], two to three VCs [PDLM97, MLDP97], and one to four VCs [WP97, PW97]. HTA uses three VCs for unicast communication and three VCs for multicast communication.

Figure 5.6 shows the probability of deadlock versus normalized applied load for HTA. The probability of deadlock is the total number of potentially deadlocked messages (PDM) divided by the number of messages that have reached their destinations. The probabilities are low except near saturation. Taking into account the differences in the simulations (e.g. timeout and reinjection delays, bidirectionality, switching type, message and network size), HTA's results are comparable to those reported for k-ary n-cubes under unicast traffic in [PW97, WP97, KLC94, FR97, MLDP97].

In schemes such as DISHA, when a potential deadlock occurs one of the messages in the deadlocked set is removed from the network using additional buffers at each node to route the message directly and immediately to its destination and therefore the PDM

does not have another chance of deadlocking [VP95b, WP97, PW97]. In HTA the PDM is removed from the network at the current node and is reinjected into the network after a given amount of time at the same current node in which it deadlocked. The reinjected PDM may potentially deadlock again along the path to its destination node. HTA does not require as much complex logic, is scalable, and does not have a single point of failure.

Increasing the number of destinations per message results in a greater probability that a destination will block, increasing deadlock probability. The greater the message size, the greater the deadlock probability because more resources are occupied. Increasing network size results in a longer path between source and destination and also results in greater deadlock probability.

5.2.2 Multicast Latency

Message Latency. Figure 5.7 shows that for all multicast communication and for small message sizes simulated on an 8-ary 2-cube network, HTA performs best among all three algorithms at low utilization. At high utilization the Column-Path algorithm performs best. However, the time necessary for destination ordering in the Column-Path algorithm is not accounted for. When this time is included, HTA performs best even at high utilization.

For large message sizes, HTA always performs best because the blocking probability has been kept low, the deadlock detection and recovery algorithm is efficient, and because tree-based routing does not unnecessarily copy data flits when routing multicast messages.

As the number of destinations per message increases, the Column-Path algorithm performs better (although not as well as HTA) because more destinations can be grouped together, requiring less number of submulticast messages to be sent per multicast message.

Saturation Point. HTA always has the highest saturation point since channels are only used and data flits are only copied when necessary. Traffic in the network is kept low resulting in increased saturation points.

Effects of Network Size. HTA's performance increases as as the 8-ary 2-cube network is increased in size to a 16-ary 2-cube network. Its performance is always better than the other two algorithms for all message sizes.

The Column-Path algorithm performance slightly suffers due to the lower probability that messages will fall in the same column and therefore uses more submulticast messages. The HTA is much more flexible with respect to topology and its latency remains low.

Effects of Traffic Type (Unicast vs. Multicast). When traffic is composed of all unicast messages (Figure 5.8), the Column-Path and Software Multicast algorithms give similar performance because both these algorithms use the same unicast routing scheme and have the same number of VCs devoted to unicast messages. The slight variances are simply due to the random generation of messages.

At low utilization, HTA performs best because only three VCs per dimension are devoted to unicast communication while the other two algorithms devote all six VCs. Having fewer number of VCs per dimension means less message multiplexing, resulting in lower message latencies.

At high utilization, the Column-Path and Software Multicast algorithms perform better because each of these algorithms has six VCs per dimension for unicast messages. Six VCs provides greater adaptivity for messages to be routed around blocked messages at high utilization, resulting in higher saturation points.

When traffic is composed of half unicast and half multicast traffic (Figure 5.9), unicast and multicast latency versus applied load graphs are shown. HTA performs comparable to

or better than the other algorithms for all messages sizes and at low utilization. At high utilization, HTA performs best for large message sizes and Software Multicast performs best for small message sizes.

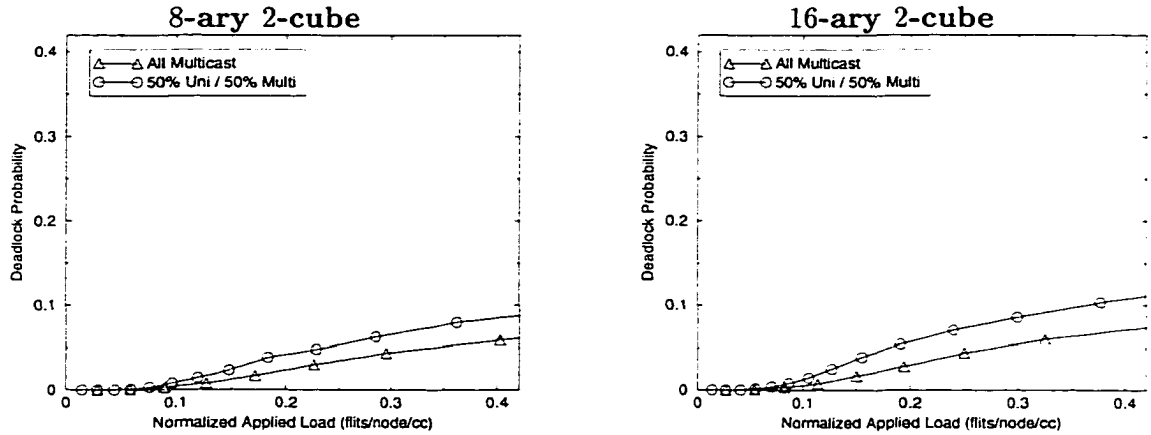
5.3 Related Work

HTA differs in many important respects from the previously proposed tree-based routing algorithms [LMN91, MDT96, BSD89, BND87]. First, in most tree-based routing algorithms, wormhole switching is implemented. Although in [BSD89, BND87] cut-through switching is used, the switching is implemented using a common buffer pool and the buffer is located in the local processing node (not in the router itself). In HTA a buffer is implemented at every VC and every buffer is located in the router. This greatly increases performance.

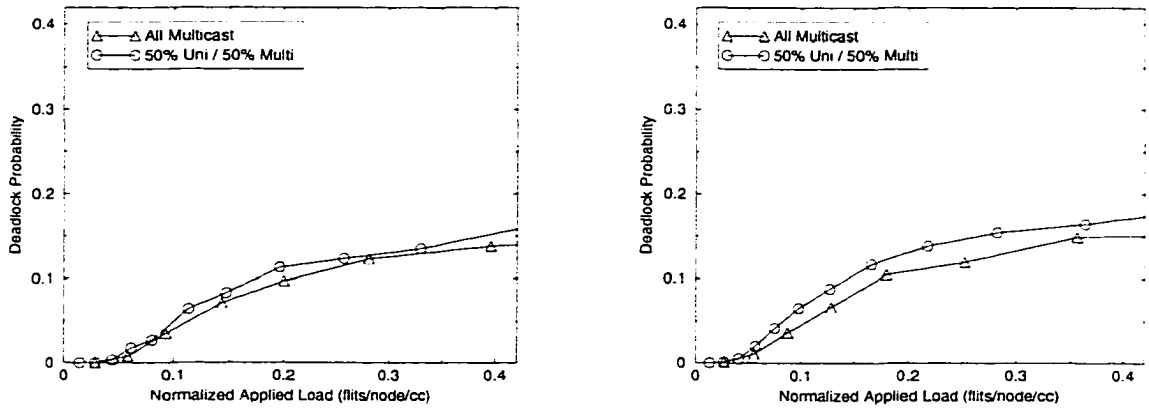
In [BSD89, BND87] only one PC (with no VCs per PC) is used per dimension and only one sink channel is implemented. Using more than one VC and sink channel, once again, greatly decreases the probability of blocking and results in better performance.

Although [BSD89, BND87] uses a deadlock detection and recovery scheme, the HTA scheme is more efficient. When a header flit blocks for a predetermined amount of time in HTA, the header flit is routed to the deadlock queue. All remaining header flits are then immediately routed to, first, any previously routed channels, then to any available and applicable channel and finally to the deadlock queue if no other routing option remains.

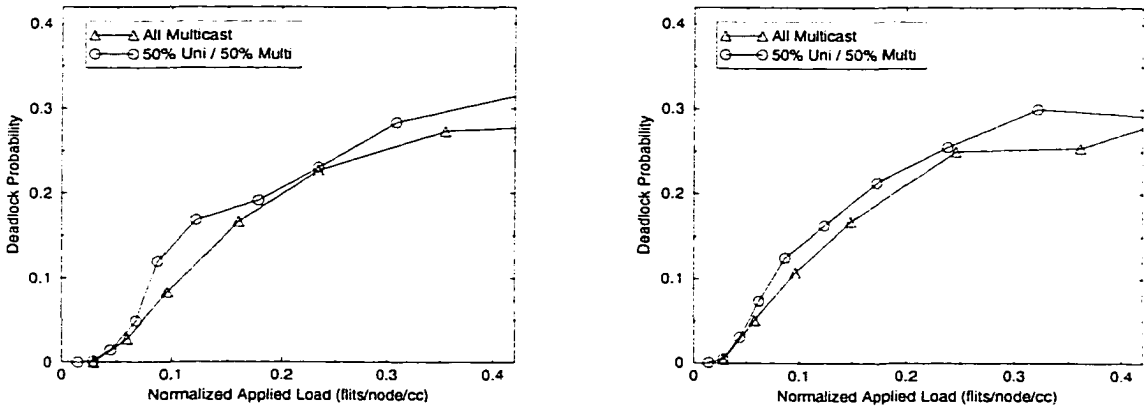
The HTA scheme improves upon the deadlock recovery method in [BSD89, BND87] in which all header flits (including all those that have already been routed at the current node and at any of the downstream nodes) are aborted. In addition, for their deadlock recovery scheme, the entire message is always copied to the local processing node when a multicast is split (whether deadlock occurs or not). When an abort does happen, the message is already stored at the local node and is ready to be reinjected into the network after a given amount



Message = 16, Destinations = 8



Message = 64, Destinations = 8



Message = 64, Destinations = 32

Figure 5.6: Deadlock Probabilities

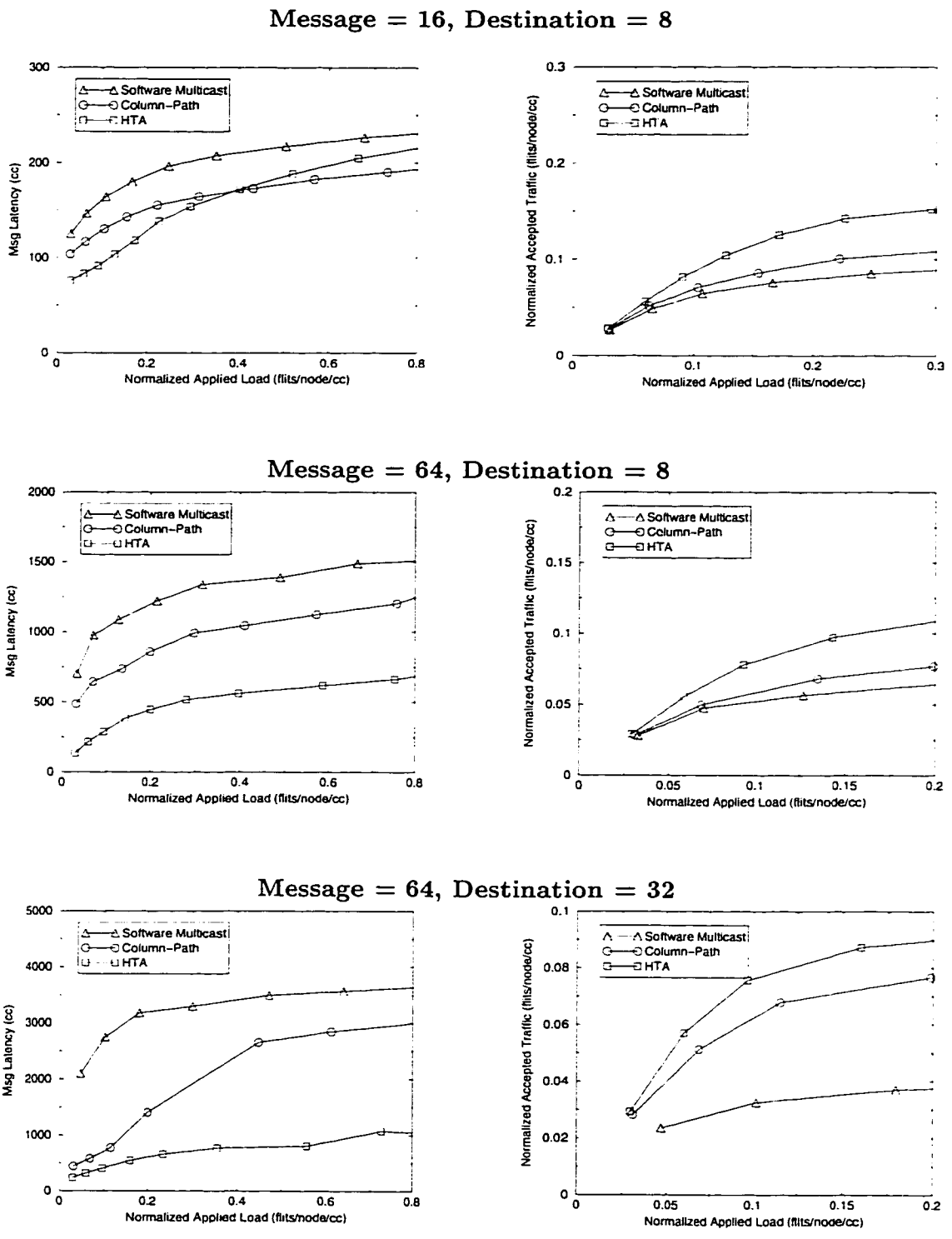


Figure 5.7: All multicast communication for an 8-ary 2-cube network.

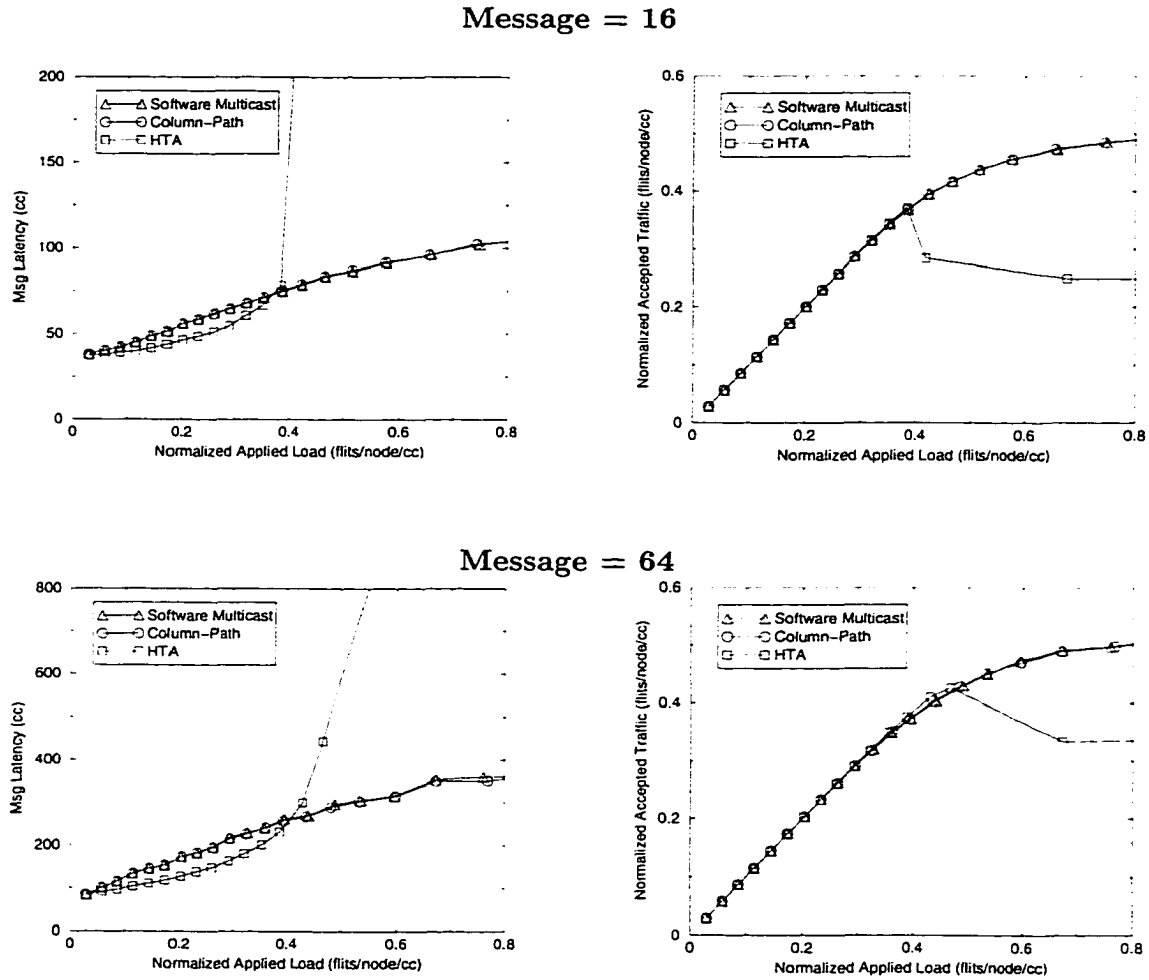


Figure 5.8: All unicast communication for an 8-ary 2-cube network.

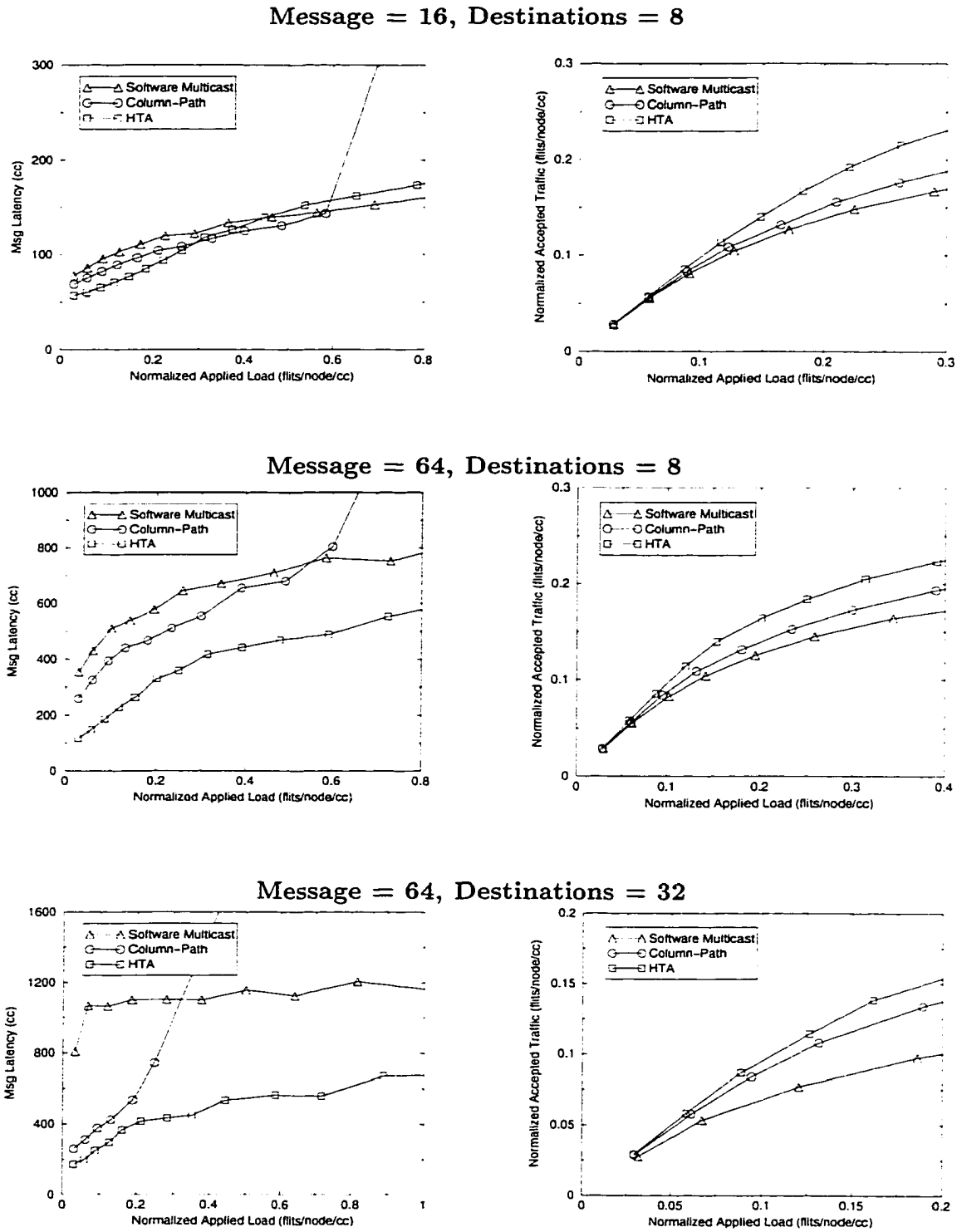


Figure 5.9: 50% unicast and 50% multicast communication for an 8-ary 2-cube network.

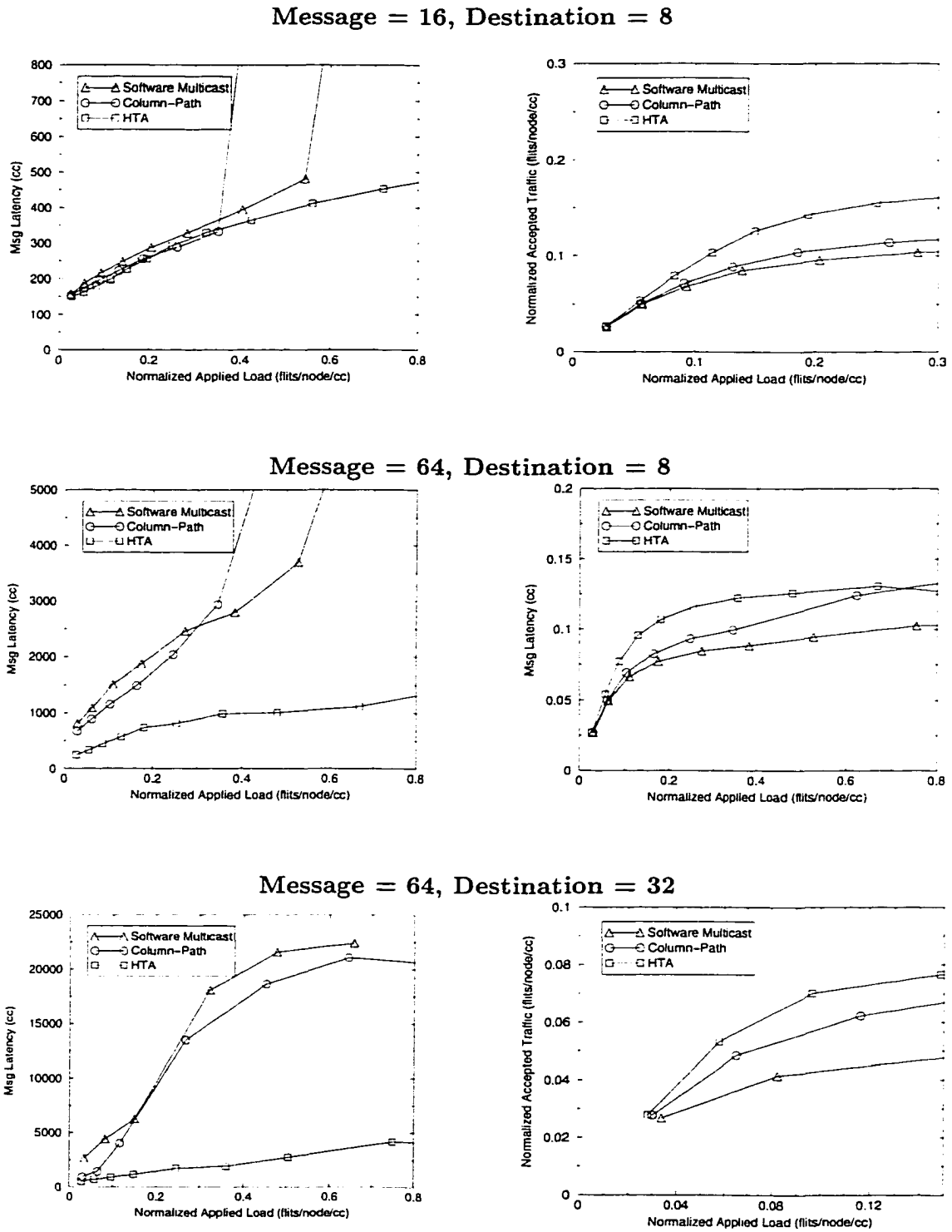


Figure 5.10: All multicast communication for a 16-ary 2-cube network.

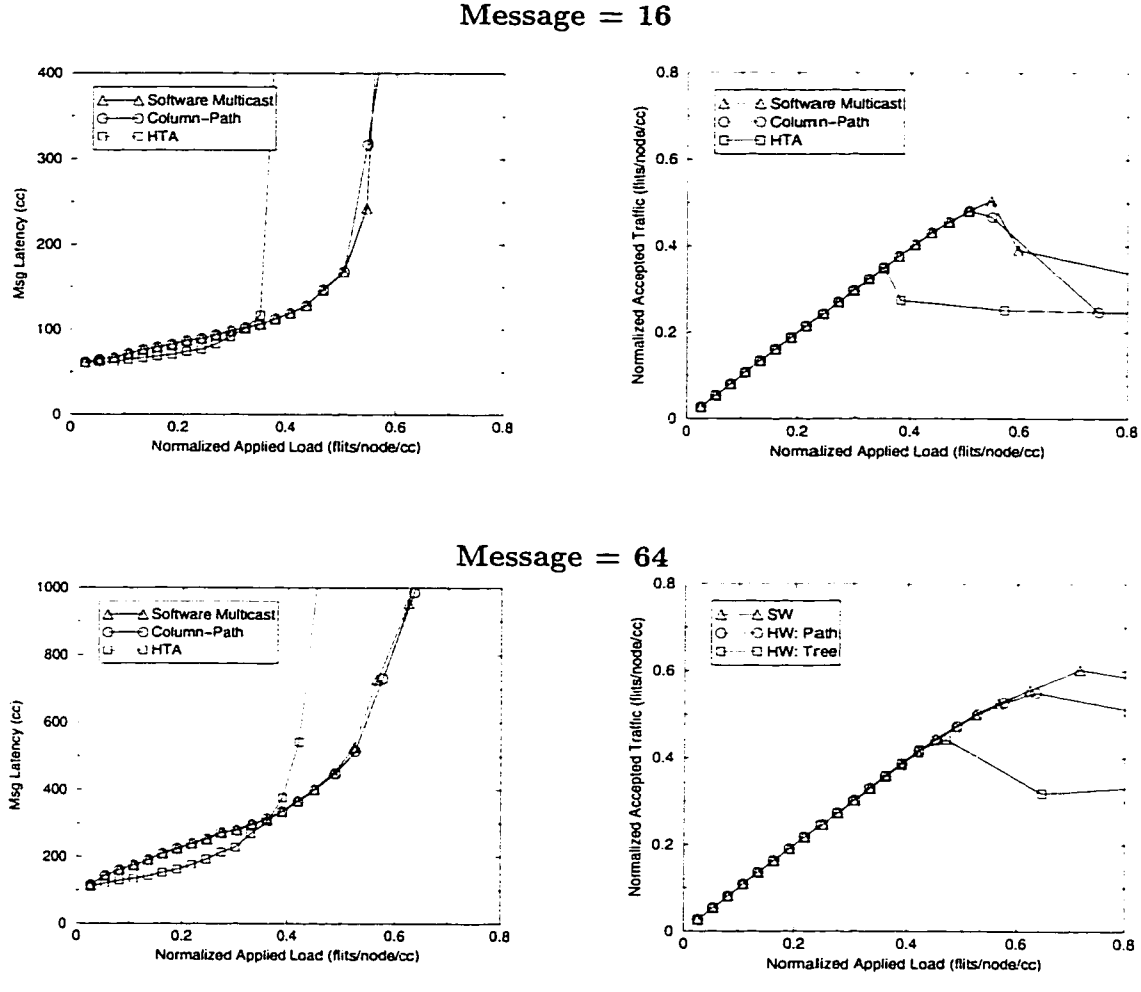


Figure 5.11: All unicast communication for a 16-ary 2-cube network.

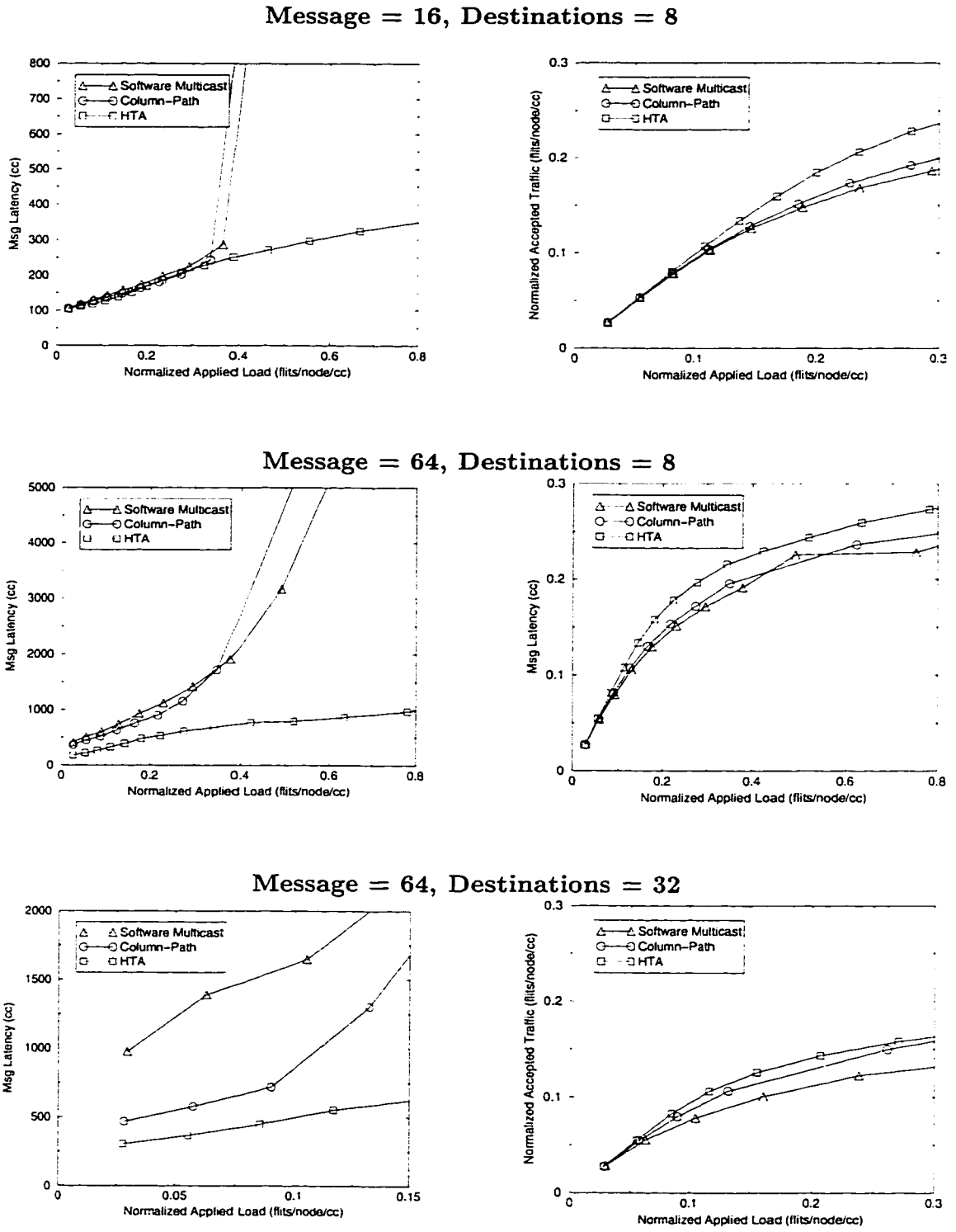


Figure 5.12: 50% unicast and 50% multicast communication for an 16-ary 2-cube network.

of time. However, this method wastes valuable channel bandwidth and causes contention in the network if two multicast channels request the split channel.

Several machines already have some hardware support for multicast. The nCUBE-2 is a wormhole-switched hypercube which supports broadcast within each subcube [NCU90]. However, deadlock is possible if multiple multicasts exist [Ni95]. The NEC Cenju-3 supports broadcast within each continuous region, but deadlock is once again possible if multiple multicasts exist. Finally, the Thinking Machines Corporation (TMC) CM-5 supports one multicast at a time via the control network.

5.4 Summary of Contributions

In this chapter, we present an empirical evaluation of a fully adaptive minimal hardware tree-based routing algorithm (HTA) for multicast communication under virtual cut-through switching in k-ary n-cubes. HTA is compatible with existing unicast routing algorithms and uses deadlock detection and recovery. The success of this new multicast routing algorithm is due to its ability to keep the probability of message blocking at each intermediate node along a multicast message's path low.

The results of this experimental evaluation show that deadlock probability remains low except near saturation. The probabilities are comparable to other research results and vary between 0% and 15%, except near saturation where it can vary up to 30%. HTA performs very well and can out-perform both Software Multicast and Column-Path algorithms.

Chapter 6

CONCLUSION

Massively parallel processors are becoming increasing important and hold great computing potential for the future. However, as these systems increase in size, scalability becomes a major factor and the interconnection network often becomes the bottleneck. Understanding the relationship between different parameters in these networks and implementing these parameters effectively can result in increased network performance. This dissertation has investigated some of these parameters with the goal of improving interconnection network performance.

6.1 Research Summary

In this dissertation numerous issues have been addressed. These include the following:

- A discussion was given on the performance of deterministic and adaptive routers for k-ary n-cube networks including input/output selection policies, adaptive router performance with respect to buffer area, physical wire bandwidth, and effects of router delay on performance.

- A hybrid deterministic/adaptive router was shown to give comparable message latency to deterministic routing at low utilization and to adaptive routing at high utilization.

- A fully adaptive hardware tree-based multicast routing algorithm with deadlock detection and recovery was shown to give comparable to or better message latency than the Software Multicast and Column-Path algorithms.

6.2 Summary of Results

Many conclusions were obtained through the numerous experimental evaluations that were performed. These include:

Deterministic and Adaptive Routing:

- All input/output selection policies give similar results under most conditions as long as the selection was fair. The Round Robin policy was chosen in the research here because it is easy to implement, gave overall good performance, and is fair.

- When varying buffer size for a given number of VCs, the buffer size dominates performance for small number of VCs. For a greater number of VCs, buffer size no longer dominates.

- A tradeoff exists between the number of VCs used. Fewer number of VCs result in smaller message latencies at low utilization. Greater number of VCs result in higher saturation points.

- Two PCs per dimension gave higher saturation points and lower message latency. However, under constant bandwidth, one PC per dimension gave better performance. This effect was more pronounced in deterministic routers than in adaptive ones.

- A tradeoff exists between constant area adaptive and deterministic routers. Under relatively uniform traffic and small message and buffer sizes, deterministic routing results in lower message latency at low throughput, while adaptive routing gives higher

saturation points. For large message and buffer sizes, adaptive routing performs better at all throughput values for all types of traffic.

- Comparable performance between deterministic and adaptive routers was found only for small message sizes and for uniform random and complement traffic. Under these circumstances, a deterministic router with twice the buffer area as an adaptive router has about the same saturation point as the adaptive router.

Hybrid Routing:

- The hybrid router obtains a message latency comparable to that of the deterministic router at low traffic and a saturation point close to that of the adaptive router at high traffic.

- As message size increases under random uniform traffic, the performance advantage of the PHR decreases compared to the deterministic and adaptive pipelined routers. This is due to the facts that more messages, and therefore headers, are needed to achieve the same utilization with short message length and the PHR has a performance advantage for header flits, especially at low utilization.

- The performance of the PHR under all non-random traffic patterns is similar to that for random uniform traffic. Once again, the PHR performs best at low traffic, while the adaptive router performs slightly better at high traffic.

- Under all traffic patterns, the super-pipelined implementations for all routers achieve better overall performance gain than the pipelined implementations. This is due to the higher throughput that is achieved by deeper pipelines.

Multicast Routing (HTA):

- The probability of potential deadlocks using HTA is low except near saturation and is comparable to those found in [PW97, WP97, KLC94].
- The probability of potential deadlock is higher for a greater number of destinations per message, for larger message lengths, and for increasing network size.
- For all multicast communication and for small message sizes, HTA performs best at low utilization. At high utilization the Column-Path algorithm and HTA give comparable performance. For large message sizes, HTA always performs best.
- HTA's saturation point is always greater than the other two algorithms for all multicast communication.
- When only unicast communication is simulated, HTA performs best at low utilization, while the Column-Path and Software Multicast algorithms have higher saturation points.
- When traffic is composed of half unicast and half multicast traffic, HTA performs comparable to or better than the other algorithms for all messages sizes and at low utilization. At high utilization, HTA performs best for large message sizes and Software Multicast performs best for small message sizes.
- As the radix of the network increases, HTA's performance remains similar to that obtained in lower radix networks. The main difference is that HTA now always performs best for multicast and mixed unicast and multicast traffic.

REFERENCES

[AC95] K. Aoyama and A. Chien. The cost of adaptivity and virtual lanes in wormhole router. *J. of VLSI Design*, 2(4), 1995.

[ACC+90] R. Alverson, D. Callahan, D. Cummings, B. Koblenz, A. Portfield, and B. Smith. The Tera computer system. In *1990 Int'l. Conf. Supercomputing*, pages 1–6. ACM Press, 1990.

[AG89] G. Almasi and A. Gottlieb. *Highly Parallel Computing.* The Benjamin Cummings Publishing Company, Inc, 1989.

[Bar64] P. Baran. On distributed communication networks. *IEEE Trans. on Commun. Systems.* CS-12:1–9, Mar 1964.

[BCR94] R. Boppana, S. Chalasani, and C. Raghavendra. On multicast wormhole routing in multicomputer networks. In *Symp. on Parallel and Distributed Processing*, Oct 1994.

[BCR98] R. Boppana, S. Chalasani, and C. Raghavendra. Resource deadlocks and performance of wormhole multicast routing algorithms. *IEEE Trans. on Parallel and Distributed Systems*, 9(6), Jun 1998.

[BGPS92] P. Berman, L. Gravano, G. Pifarre, and J. Sanz. Adaptive deadlock and livelock free routing with all minimal paths in torus networks. In *Proc. of the Symp. on Parallel Algorithms and Architectures*, pages 3–12, 1992.

[BND87] G. Byrd, R. Nakano, and B. Delagi. A dynamic cut-through communication protocol with multicast. Technical Report STAN-CS-87-1148, Stanford University, Aug 1987.

[Bou94] G. Boughton. Arctic routing chip. In *Workshop on Parallel Computer Routing and Communication*, pages 310–7, 1994.

[BSD89] G. Byrd, N. Saraiya, and B. Delagi. Multicast communication in multiprocessor systems. In *International Conference on Parallel Processing*, volume 1, pages 196–200, Aug 1989.

[Chi93] A. Chien. A cost and speed model for k-ary n-cube wormhole routers. In *IEEE Proc. of Hot Interconnects*, Aug 1993.

[CK92] A. A. Chien and J. H. Kim. Planar-adaptive routing: low cost adaptive networks for multiprocessors. *Int. Symp. on Computer Architecture*, pages 268–277, May 1992.

[CM287] Connection Machine Model CM-2 Technical Summary. Technical Report HA87–4, Thinking Machines Corporation, 1987.

[CN95] C. Chiang and L. Ni. Network partitioning and unicast-based multicast on multistage networks. Technical Report MSU-CPS-ACS-102, Michigan State University, Mar 1995.

[DAC89] W. Dally and et al A. Chien. The J-Machine: a fine-grain concurrent computer. In *Proc. of the IFIP Congress*, pages 1147–1153, Aug 1989.

[Dal89] W. J. Dally. The J-machine: system support for actors. In Hewitt and Agha, editors, *Actors: Knowledge Based Concurrent Computing*. MIT Press, 1989.

[Dal92] W. J. Dally. Virtual-channel flow control. *IEEE Trans. on Computers*, 3(2):194–205, Mar 1992.

[DDH+94] W. Dally, L. Dennison, D. Harris, K. Kan, and T. Xanthopoulos. Architecture and implementation of the Reliable Router. In *IEEE Proc. of Hot Interconnects II*, Aug 1994.

[DL94] J. Duato and P. Lopez. Performance evaluation of adaptive routing algorithms for k-ary n-cubes. In *Parallel Computer Routing and Communication*, pages 45–59, 1994.

[DLSY96] J. Duato, P. Lopez, F. Silla, and S. Yalamanchili. A high performance router architecture for interconnection networks. In *Int. Conf. on Parallel Processing*, August 1996.

[DLY97] J. Duato, P. Lopez, and S. Yalamanchili. Deadlock- and livelock-free routing protocols for wave switching. In *Int. Parallel Processing Symp.*, April 1997.

[DS86] W. J. Dally and C. L. Seitz. The torus routing chip. *J. Dist. Computing*, 1(3):187–196, 1986.

[DS87a] W. Dally and P. Song. Design of a self-timed VLSI multicomputer communicaton controller. In *Proc. of the Int. Conf. on Computer Design*, pages 230–40, 1987.

[DS87b] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. on Computers*, C-36(5):547–553, May 1987.

[Dua91] J. Duato. Deadlock-free adaptive routing algorithms for multicomputers: Evaluation of a new algorithm. In *Proc. of the 3rd IEEE Symp. on Parallel and Distributed Processing*, Dec 1991.

[Dua93] J. Duato. A new theory of deadlock-free adaptive routing in wormhole networks. In *Symposium on Parallel and Distributed Processing*, pages 64–71, 1993.

[Dua95] J. Duato. A theory of deadlock-free adaptive multicast routing in wormhole networks. *IEEE Trans. on Parallel and Distributed Systems*, 6(9):976–87, Sep 1995.

[DYD97] B. Dao, S. Yalamanchili, and J. Duato. Architectural support for reducing communication overhead in multiprocessor interconnection networks. In *High Performance Computer Architecture*, pages 343–52, 1997.

[DYN97] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks: An Engineering Approach*. The Institute of Electrical and Electronics Engineers, Inc., 1997.

[ES93] et al. E. Spertus. Evaluation of mechanisms for fine-grained parallel programs in the J-Machine and the CM-5. *Computer Architecture News*, 2(21), May 1993.

[Fla87] C. Flaig. VLSI mesh routing systems. Master's thesis, California Institute of Tehnology, May 1987.

[FR97] A. Folkestad and C. Roche. Deadlock probability in unrestricted wormhole routing networks. In *IEEE International Conference on Communications*, Jun. 1997.

[GN92] C. L. Glass and L. M. Ni. The turn model for adaptive routing. In *Int. Symp. on Computer Architecture*, pages 278–287, May 1992.

[Gop85] I. Gopal. Prevention of store-and-forward deadlock in computer networks. *ieeetc*, 33(12):1258–64, Dec 1985.

[Hil85] W. Hillis. *The Connection Machine*. MIT Press, 1985.

[Hwa93] Kai Hwang. *Advanced Computer Architecture*. McGraw-Hill Book Co., 1993.

[Int91] Intel Corp. *Paragon XP/S Product Overview*, 1991.

[KA93] J. Kubiatowicz and A. Agarwal. Anatomy of a message in the alewife multiprocessor. *7th ACM Int. Conf. on Supercomputing*, 1993.

[KK79] P. Kermani and L. Kleinrock. Virtual cut-through: a new computer communication switching technique. *Computer Networks*, 3:267 - 286, 1979.

[KLC94] J. Kim, Z. Liu, and A. Chien. Compressionless routing. In *Proc. of ISCA*, Apr 1994.

[KP96] R. Kesavan and D. Panda. Minimizing node contention in multiple multicast on wormhole k-ary n-cube networks. In *Int. Conf. on Parallel Processing*, 1996.

[KS91] S. Konstantinidou and L. Snyder. Chaos router: architecture and performance. *Int. Symp. on Computer Architecture*, pages 212–221, 1991.

[KS93] R. Kessler and J. Schwarzmeier. CRAY T3D: a new dimension for cray research. *Compcon*, pages 176–82, 1993.

[L+92] C. E. Leiserson et al. The network architecture of the Connection Machine CM-5. In *Symp. on Parallel Algorithms and Architectures*, 1992.

[Lag94] Annette Lagman. *Modelling, Analysis and Evaluation of Adaptive Routing Strategies.* PhD thesis, Colorado State University, Computer Science Department, November 1994.

[LD94] Z. Liu and J. Duato. Adaptive unicast and multicast in 3d mesh networks. In *27th Annual Hawaii International Conference on System Sciences*, pages 173–82, Jan 1994.

[LH91] D. H. Linder and J. C. Harden. An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes. *IEEE Trans. on Computers*, 40(1):2–12, Jan 1991.

[LLG+92] D. Lenoski, J. Laudon, K. Gharachorloo, W. Weber, A. Gupta, J. Hennessy, M. Horowitz, and M. Lam. The Stanford DASH multiprocessor. *IEEE Computer*, 25(3):63–79, Mar 1992.

[LMDP97] P. Lopez, J. Martinez, J. Duato, and F. Petrini. On the reduction of deadlock frequency by limiting message injection in wormhole networks. In *Parallel Computing, Routing, and Communication Workshop*, Jun. 1997.

[LME93] X. Lin, P. K. McKinley, and A. Esfahanian. Adaptive multicast wormhole routing in 2d mesh multicomputers. In *Proc. PARLE'93*, pages 228–241, 1993.

[LMN91] X. Lin, P. McKinley, and L. Ni. Performance evaluation of multicast wormhole routing in 2D mesh multicomputers. In *Int. Conf. on Parallel Processing*, pages 435–42, 1991.

[LN91] X. Lin and L. M. Ni. Deadlock-free multicast wormhole routing in multicomputer networks. In *Int. Symp. on Computer Architecture*, pages 116–125, 1991.

[MDT96] M. Malumbres, J. Duato, and J. Torrellas. An efficient implementation of tree-based multicast routing for distributed shared-memory multiprocessors. In *8th IEEE Symp. on Parallel and Distributed Processing*, pages 186–9, Oct 1996.

[MLDP97] J. Martinez, P. Lopez, J. Duato, and T. Pinkston. Software-based deadlock recovery technique for true fully adaptive routing in wormhole networks. In *International Conference on Parallel Processing*, Aug. 1997.

[MS86] C. Moler and D. Scott. Communication utilities for iPSC. Technical Report iPSC Technical report No. 2, Intel Scientific Computers, 1986.

[MXEN94] P. McKinley, H. Xu, A. Esfahanian, and L. Ni. Unicast-based multicast communication in wormhole-routed networks. *IEEE Trans. on Parallel and Distributed Systems*, 5:1252–65, 1994.

[NCU90] NCUBE Company. *NCUBE 6400 Processor Manual*, 1990.

[Ni95] L. Ni. Should scalable parallel computers support efficient hardware multicast. In *Int. Conf. on Parallel Processing*, 1995.

[NM93] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. *IEEE Computer*, pages 62–76, 1993.

[NWD93] M. Noakes, D. Wallach, and W. Dally. The J-Machine multicomputer: An architectural evaluation. In *20th International Symposium on Computer Architecture*, May 1993.

[PBGB93] G. Papadopoulos, G. Boughton, R. Greiner, and M. Beckerle. *T: Integrating building blocks for parallel computing. In *Supercomputing '93*, Nov 1993.

[PDLM97] F. Petrini, J. Duato, P. Lopez, and J. Martinez. LIFE: a limited injection, fully adaptive, recovery-based routing algorithm. In *Fourth Inter. Conf. on High Performance Computing*, Dec. 1997.

[PSP94] D. K. Panda, S. Singal, and P. Prabhakaran. Multidestination message passing mechanism conforming to base wormhole routing scheme. In *Proc. First Parallel Routing and Communication Workshop*, Seattle, Washington, 1994.

[PW97] T. Pinkston and S. Warnakulasuriya. On deadlocks in interconnection networks. In *International Symp. on Computer Architecture*, pages 38–49, Jun. 1997.

[RMC97] D. Robinson, P. McKinley, and B. Cheng. Path-based multicast communication in wormhole-routed unidirectional torus networks. *J. of Parallel and Distributed Computing*, 45:104–21, 1997.

[SAF+88] C. Seitz, W. Athas, C. Flaig, A. Martin, J. Seizovic, C. Steele, and W. Su. The architecture and programming of the ametek series 2010 multicomputer. In *Third Conference on Hypercube Computers*, pages 33–6, 1988.

[Sco96] S. Scott. Synchronization and communication in the T3E multiprocessor. In *Symposium on Architectural Support for Programming Languages and Operating Systems*, Oct 1996.

[SS93] C. Seitz and W. Su. A family of routing and communication chips based on the mosaic. In *Proc. of the University of Washington Symp. on Integrated Systems*, 1993.

[ST96] S. Scott and G. Thorson. The Cray T3E networks: adaptive routing in a high performance 3d torus. In *Hot Interconnects IV*, Aug 1996.

[Tuc84] A. Tucker. *Applied Combinatorics*. John Wiley & Sons, second edition, 1984.

[VP95a] Anjan V. and T. Pinkston. An efficient, fully adaptive deadlock recovery scheme: DISHA. *Computer Architecture News*, 23(2), May 1995.

[VP95b] Anjan K. V. and T. Pinkston. An efficient, fully adaptive deadlock recovery scheme: DISHA. In *Int. Symp. on Computer Architecture*, pages 201–10, 1995.

[WP97] S. Warnakulasuriya and T. Pinkston. Characterization of deadlocks in interconnection networks. In *Int. Parallel Processing Symp.*, Apr. 1997.

[XGN94] H. Xu, Y. Gui, and L. Ni. Optimal software multicast in wormhole-routed multistage networks. In *Supercomputing*, pages 703–12, Nov 1994.